Turbulence in SPH

Pascal Ulrich Förster

Munich 2014

Turbulence in SPH

Bachelor's thesis

at the

Ludwig–Maximilians–University Munich

handed in by

Pascal Ulrich Förster

(Matr. No.: 8052487)

born December 5, 1987 in Düsseldorf, Germany

supervised by

PD Dr. Klaus Dolag

and

Dr. Alexander Beck

Munich, September 30, 2014

Evaluator: PD Dr. Klaus Dolag

Date of the oral exam: September 30, 2014

Turbulenz in SPH

Bachelor-Arbeit

an der

Ludwig–Maximilians–Universität München

eingereicht von

Pascal Ulrich Förster

(Matr.-Nr.: 8052487)

geboren am 5. December 1987 in Düsseldorf, Deutschland

betreut von
PD Dr. Klaus Dolag

und
Dr. Alexander Beck

München, den 30. September 2014

Gutachter: PD Dr. Klaus Dolag

Tag der mündlichen Prüfung: 30. September 2014

Table of Contents

1. Introduction

Noli turbare circulos meos!
— Archimedes

Some philosophers argue that there is a part of human nature which seeks to find order in the most tumultuous of affairs. Is it surprising, then, that physics, the natural science with the greatest aspiration to a fundamental knowledge of our universe, is also the most prone to dissecting and reducing impossibly complex problems into ever smaller bits, until the substructure lines up in front of the scientist's eye and is presented to him in an orderly fashion, sorted for example by fundamental force, by subatomic particle, and so on?

The initial quote attributed to Archimedes of Syracuse moments before his death – he was defending his geometric figures drawn in the sand against the destructive footsteps of an ignorant soldier – expresses this appreciation of order, this perception of beauty in all things geometrical and symmetric, in all phenomena orderly and reducible. We want to explore them down to their core and understand their workings from the smallest instance up to their largest composition.

Turbulence evades that grasp, though, as the sheer numbers of its participants makes it unfeasible to analyze the phenomenon on a microscopic, individual-particle level. Even only one mole of gas has approximately 10^{24} particles, and usually we will be dealing with several orders of magnitude more than that. Nonetheless, turbulence can be treated physically, or else this would be a Bachelor's thesis in philosophy rather than in physics. But the approach is fundamentally different and all-encompassing, since even the most unordered of turbulence can be handled statistically, not from the bottom to the top, but on a huge, macroscopic or even astronomical scale. And whatever the size or location of those turbulent eddies and fluctuations, be they in the pipes of our sewers or in the gas between our stars, they follow the same universal set of rules. In all their individual tangle, in all the tumult and all their invincible complexity, they still adhere to that all-comprising order, and thus are again in the physicist's grasp. In a way, then, the physics of turbulence is one of the greatest triumphs of science over what appeares to be a world of chaos.

There is a beauty to that.

1.1. Astrophysical motivation

Turbulence is Nature's way of transferring energy from larger to smaller scales by finding increasingly intricate ways to "store" or better, say, maintain a certain kinetic energy despite the limitations of outer conditions. Although being woefully neglected in many undergraduate physics courses on fluid mechanics, turbulence is an integral part of that topic and occurs in as many and different situations as seem possible. As a phenomenon mainly defined by the ratio of inertial and viscous forces and not by the actual magnitude of the forces involved, it ranges from the smallest eddies of a liquid streaming through a minuscle pipe over the better part of terrestial atmospheric circulation to the turbulent behavior of the intracluster medium (ICM), which is the hot gas in between the galaxies of the largest gravitationally bound systems of the known universe. Some astrophysical examples of turbulence are listed below.

Stars

The surface and outer layer of stars like our own sun is called the *convection zone* and is dominated by energy transport via convection, as opposed to e.g. the radiation zone below it. The imprint of turbulent process on the surface of the Sun can be nicely seen in figure 1.1. The maximum, external lengthscale for this kind of turbulence is given by the radius of the star in question, which can range from about $0.5\,R_\odot$ up to $1700\,R_\odot$, with $R_\odot = 6.955 \cdot 10^5\,\mathrm{km}$ the solar radius.

Interstellar medium

The matter between the star systems in a galaxy is called the *interstellar medium* (ISM). It is composed of different phases and temperatures, and is interspersed with stars and other massive objects that interact with it. In combination with the overall rotation of the galaxy, this makes the ISM turbulent. The external lengthscale for that turbulence is the outer dimension of the galaxy, which for our Milky Way is about $100,000$ lightyears (ly) or about $30\,\mathrm{kpc}$. For comparison: our two orbiting dwarf galaxies, the Large and the Small Magellanic Cloud, "only" measure approximately $25,000\,\mathrm{ly}$ and $10,000\,\mathrm{ly}$ (or about $7.7\,\mathrm{kpc}$ and $3.1\,\mathrm{kpc}$), respectively. Andromeda, which is depicted in figure 1.2, is the closest spiral galaxy to us. Besides being about the same size as our Milky Way, it is believed to have very similar characteristics and to be on a collision course with us.

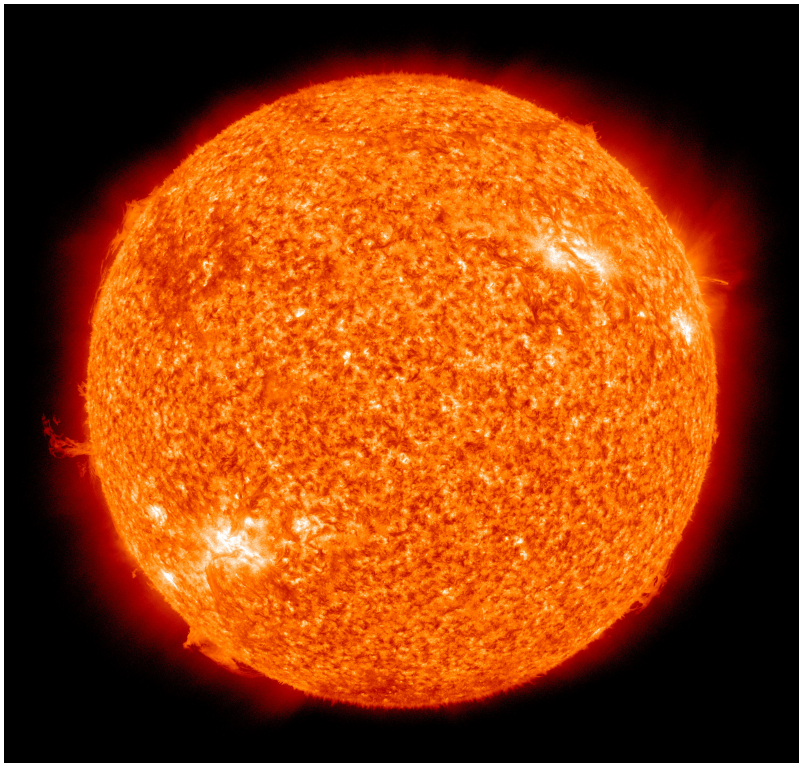

Figure 1.1.: False-color image of the Sun observed in the extreme ultraviolet region of the spectrum taken by the Atmospheric Imaging Assembly (AIA 304) of NASA's Solar Dynamics Observatory (SDO). Image courtesy of NASA/SDO and the AIA, EVE, and HMI science teams.

Figure 1.2.: The Andromeda Galaxy as seen through a small telescope. Image credit: Jacob Bers (bersonicastronomy.com).

Intracluster medium

The vast expanses between galaxies of a cluster is host to the largest known form of turbulence, which is occuring in the *intracluster medium* (ICM), a superheated and thin plasma mainly consisting of ionised hydrogen and helium with a mean free path of about one lightyear. The ICM interconnects all the galaxies of a cluster with each other. As clusters of galaxies are some of the largest gravitationally bound structures in the known universe (and in this regard only surpassed by superclusters), the maximum lengthscale of the turbulence, in this case the size of the cluster, will be mind-boggling 2 to 10 Mpc big. Figure 1.3 shows the massive Perseus Cluster, containing thousands of galaxies.

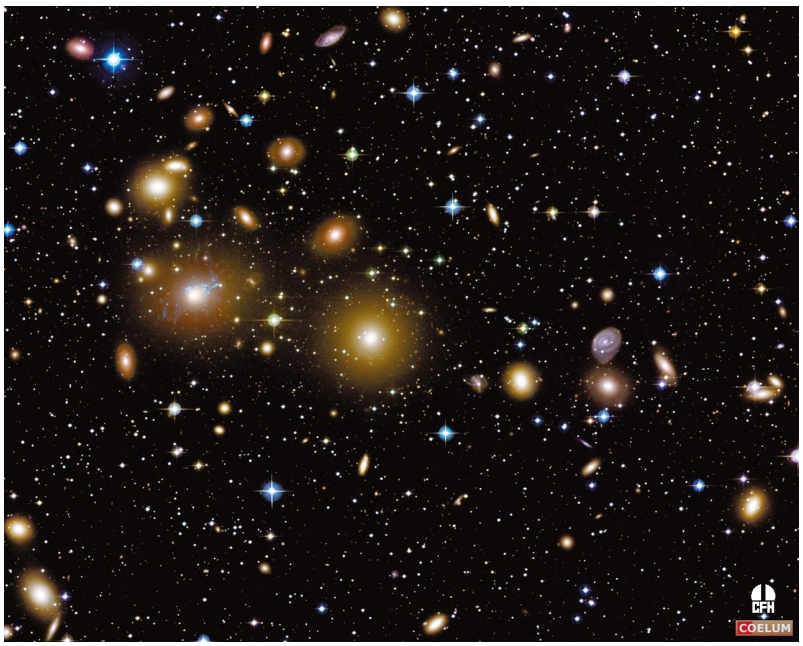

Figure 1.3.: The Perseus Cluster of Galaxies, one of the closest galaxy clusters. It is part of the Pisces-Perseus supercluster with more than 1000 galaxies. The view of this picture covers about 7.5 million light-years. Image credit: Jean-Charles Cuillandre (CFHT) and Giovanni Anselmi (Coelum Astronomia), Hawaiian Starlight.

1.2. Outline

The thesis at hand has been prompted by recent publications about the "natural" limitations of smoothed particle hydrodynamics (SPH) with regard to the simulation of turbulence. These findings center around unwanted side effects of artificial viscosity, which needs to be introduced into the otherwise perfectly Lagrangian scheme to implement dissipation. Most prominent in that regard is the paper by

Bauer and Springel (2012) about deficits in the subsonic turbulent regime, which dissuades further use of SPH codes in the regime and suggests the use of moving-mesh codes like Arepo. Although a well-considered answer to that specific paper already exists in form of Price (2012a), the idea had been sown to assess our current SPH code Gadget-3 equipped with state-of-the-art schemes and compare it to matching Arepo runs.

The structure of this thesis is as follows: In chapter 2 will we introduce the basic equations of hydrodynamics necessary for our further understanding of turbulence, then move on to the theory of turbulence itself in chapter 3. Both chapters are based on the comprehensive work of Landau and Lifshitz (1991). Chapter 4 discusses the basic concepts of smoothed particle hydrodynamics simulations, while chapter 5 presents the specifics of our simulations and chapter 6 their results. Finally, the last chapter 7 consists of a short summary, an analysis of our efforts so far and of some future prospects. The thesis also features an extensive appendix starting from page 73 with most of the code and all the configuration and parameter files.

2. A brief sketch of hydrodynamics

First of all, we need to clarify that since the hydrodynamical observations are macroscopic in nature, all effects observed and described are not so much properties of individual particles making up the fluid as properties of the continuum they form. This requires us to maintain, at any point, volume elements large enough with respect to the individual molecules so that microscopic events are negligible. Whenever we are speaking of "infinitesimally small volume elements", this will mean nothing else than the volume element at hand is sufficiently small with respect to the surroundings and typical lengthscales but still big enough in the aforementioned sense.

2.1. The continuity equation

Any fluid can be described by functions of the velocity distribution $\vec{v}(x, y, z, t)$ and two arbitrary thermodynamical quantities[1], a common example being pressure $p(x, y, z, t)$ and density $\rho(x, y, z, t)$. All of these quantities are functions of the coordinates in the sense that they describe the properties of the flow at that point in space and time in the laboratory system and not the properties of a specific particle or volume element of the fluid; they are *not* comoving.

The continuity equation describes the conservation of mass and is a concept shared with other field theories. To obtain it, we examine the different possibilities in the change of mass for a finite volume V_0 that has an overall mass of $\int_{V_0} \rho \mathrm{d}V$. The vector surface element $\mathrm{d}\vec{f}$ of this volume shall be perpendicular to its surface, outward oriented and its norm of the value of the surface it represents. With regard to units it makes sense that the flow of mass and ergo fluid per time shall then be $\rho \vec{v}\, \mathrm{d}\vec{f}$. Hence we gain the expression

$$\oint_{\partial V_0} \rho \vec{v}\, \mathrm{d}\vec{f}$$

with ∂V_0 the closed surface of the volume. On the other hand, the obvious approach

[1] This is due to the fact that monoatomic fluids with ideal properties pose a two-dimensional problem with respect to the state space and require the same number of parameters D to be described by an equation of state. The description of the dynamic component is handled by the velocity distribution.

to the change in mass is its time derivative

$$-\frac{\partial}{\partial t}\int_{V_0}\rho\,\mathrm{d}V\ .$$

Since we want to equate both expressions, we transform the first via the divergence theorem (also known as Gauss's theorem) as follows

$$\oint_{\partial V_0}\rho\,\vec{v}\,\mathrm{d}\vec{f}=\int_{V_0}\mathrm{div}\,(\rho\vec{v})\,\mathrm{d}V \tag{2.1}$$

and combine them both into the equation

$$\int_{V_0}\left[\frac{\partial\rho}{\partial t}+\mathrm{div}\,(\rho\vec{v})\right]\mathrm{d}V=0\ .$$

For this integral to be zero for any possible finite volume V_0, the integrand needs to be equal to zero, too, which gives us the *continuity equation* in either form

$$\frac{\partial\rho}{\partial t}+\mathrm{div}\,(\rho\vec{v})=0\qquad\text{or}\qquad\frac{\partial\rho}{\partial t}\ +\ \rho\,\mathrm{div}\,\vec{v}\ +\ \vec{v}\,\mathrm{grad}\,\rho=0\ . \tag{2.2}$$

2.2. Euler's equation of hydrodynamics

Now that we have gained insight into the conservation of mass and have deduced the proper equation describing it, we need to understand more about the motion of our fluid. The equation describing it is known as Euler's equation. We start with the force that the surrounding fluid exerts onto an arbitrary volume element V_0. That force can easily be quantified by the closed integral of the pressure p over the surface ∂V_0 of the fluid element, which again may be rewritten similarly to equation (2.1) according to Gauss' theorem:

$$-\oint_{\partial V_0}p\,\mathrm{d}\vec{f}=-\int_{V_0}\mathrm{grad}\,p\,\mathrm{d}V\ .$$

Then the force per volume element $\mathrm{d}V$ is given by $-\mathrm{grad}\,p$. Another approach to the force per element of the fluid is, of course, the classical "mass times acceleration"-approach derived from Newton's second law, which in this case takes the form of the product of the mass density ρ and the acceleration $\frac{\mathrm{d}\vec{v}}{\mathrm{d}t}$. Both expressions are equal, and thus

$$\rho\,\frac{\mathrm{d}\vec{v}}{\mathrm{d}t}=-\mathrm{grad}\,p\ . \tag{2.3}$$

Please take note that the derivative $\frac{\mathrm{d}}{\mathrm{d}t}$ is not the same as $\frac{\partial}{\partial t}$. While the latter occurred before in this thesis and constitutes the change of a physical property, e.g. the velocity $\vec{v}$, of the flow over time at a fixed point in space, the former gives us the time dependent change of that property of a *given fluid element* moving within our laboratory system. To be of further use to us we need to express it in the frame of reference of the laboratory system. In the case of the velocity $\mathrm{d}\vec{v}$, the first component is the change of the velocity at a fixed point in our system during the infinitesimally short time span $\mathrm{d}t$, and its second component is the difference in velocity between two fixed points in our flow with the distance $\mathrm{d}\vec{r}$ that is being traversed by a fluid element during the aforementioned time span $\mathrm{d}t$, namely $\mathrm{d}x\,\frac{\partial\vec{v}}{\partial x} + \mathrm{d}y\,\frac{\partial\vec{v}}{\partial y} + \mathrm{d}z\,\frac{\partial\vec{v}}{\partial z} = (\mathrm{d}\vec{r}\,\nabla)\,\vec{v}$. If combined, we get

$$\mathrm{d}\vec{v} = \frac{\partial\vec{v}}{\partial t}\,\mathrm{d}t + (\mathrm{d}\vec{r}\,\nabla)\,\vec{v}\,,$$

and after forming the substantial derivative through division by $\mathrm{d}t$

$$\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = \frac{\partial\vec{v}}{\partial t} + (\vec{v}\,\nabla)\,\vec{v}\,.$$

To obtain *Euler's equation of hydrodynamics* as the equation of motion of our fluid, we insert the above expression in (2.3) and write:

$$\frac{\partial\vec{v}}{\partial t} + (\vec{v}\,\nabla)\,\vec{v} = -\frac{1}{\rho}\,\mathrm{grad}\,p\,. \tag{2.4}$$

Changes to the above equation may be necessary when handling fluids with energy dissipation that can be caused by heat exchange between different volume elements or by the viscosity of the fluid, all of which we have ignored up until now. We will further investigate the topic of viscous fluids since our simulations rely on viscosity to form turbulence.

2.3. The motion of viscous fluids

In a viscous fluid, the momentum is not only transferred by the reversible motion of volume elements of the fluid as assumed above but also through an irreversible transfer from a point of higher velocity to one of lower velocity. When dealing with classical fluids this usually happens through friction.

Since friction as well as viscosity in general is an effect of interaction between two distinct volume elements of the fluid, we are then dealing with a problem best described by a tensor, in this case the viscous stress tensor σ', that is part of a more general stress tensor σ describing the portion of the transport of impulse in

the fluid not directly related to the motion of its mass.[2] The viscous effects only occur between different fluid elements moving at different speeds and, assuming not to great a difference in velocity, we expect the part of the impulse transport in question to be dependent on the first derivatives of the velocity and thus σ'_{ik} to be linear in $\frac{\partial v_i}{\partial x_k}$. Expressions independent of those derivatives are forbidden since they would not vanish for $\vec{v} = \text{const}$ and friction would occur even between volume elements of the same speed. A similar thought holds for the uniform rotation of the entire fluid where no friction will be present, either. In that case, the velocity is $\vec{v} = \vec{\omega} \times \vec{r}$ and the aforementioned derivatives must vanish.

Assuming the isotropy of the fluid and using the Einstein notation we determine the most general form a tensor of second order satisfying our demands can take, which is

$$\sigma'_{ik} = \eta \left(\frac{\partial v_i}{\partial x_k} + \frac{\partial v_k}{\partial x_i} - \frac{2}{3} \delta_{ik} \frac{\partial v_l}{\partial x_l} \right) + \zeta \, \delta_{ik} \frac{\partial v_l}{\partial x_l} \, .$$

The scalar non-negative coefficients η and ζ are commonly referred to as dynamic viscosities and describe the properties of the isotropic fluid sufficiently. The expression in brackets is constructed to equal zero for diagonal elements of the tensor, leaving only the last summand.

To obtain the new equation of motion for a viscous fluid, we add the term $\frac{\partial \sigma'_{ik}}{\partial x_k}$ to the right side of Euler's equation (2.4), which for that purpose needs to be written in its corresponding component notation. We get the most general equation of motion for a viscous fluid, that is

$$\rho \left(\frac{\partial v_i}{\partial t} + v_k \frac{\partial v_i}{\partial x_k} \right) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_k} \left\{ \eta \left(\frac{\partial v_i}{\partial x_k} + \frac{\partial v_k}{\partial x_i} - \frac{2}{3} \delta_{ik} \frac{\partial v_l}{\partial x_l} \right) \right\} + \frac{\partial}{\partial x_i} \left(\zeta \, \delta_{ik} \frac{\partial v_l}{\partial x_l} \right) .$$

If we further assume that the differences in viscosity throughout the fluid are negligible and η and ζ may then be assumed to be constant, both can be drawn out of the derivative and we get the *Navier-Stokes equation*

$$\rho \left[\frac{\partial \vec{v}}{\partial t} + (\vec{v} \, \nabla) \, \vec{v} \right] = -\text{grad} \, p + \eta \, \Delta \vec{v} + \left(\zeta + \frac{\eta}{3} \right) \text{grad div} \, \vec{v} \, .$$

The above equation holds for compressible or incompressible fluids, but with our later use in mind we will focus on incompressible fluids and are therefore able to significantly simplify it. Due to the incompressibility, the mass density $\rho = \text{const}$ and the continuity equation (2.2) is reduced to $\text{div} \, \vec{v} = 0$, this in turn leading to the

[2] Please cf. Landau and Lifshitz (1991) for further details on the tensor notation of Euler's equation of motion and for the deduction of the impulse current of an ideal fluid.

last term of our Navier-Stokes equation being zero, too. Thus we finally obtain

$$\frac{\partial \vec{v}}{\partial t} + (\vec{v}\,\nabla)\,\vec{v} = -\frac{1}{\rho}\,\mathrm{grad}\,p + \nu\,\Delta\vec{v} \tag{2.5}$$

with $\nu = \frac{\eta}{\rho}$ the kinematic viscosity. In the next chapter on turbulence we will make good use of this particular equation.

2.4. Similar flows and the Reynolds number

The Navier-Stokes equation teaches us that the solution to hydrodynamical problems is encoded in its constituents. Since we typically wish to gain knowledge about the velocity $\vec{v}$ and the pressure p of our flow, only the kinematic viscosity $\nu = \frac{\eta}{\rho}$ will be fed into the equation.[3] Because a single parameter is by far not enough to describe even a stationary problem adequately, we will explore other ways to characterize the problem.

Assuming an incompressible fluid we can easily find that the outward appearance of the flow around an arbitrary object is dependent on the viscosity of the fluid, the shape of the object and its movement through the fluid.[4] Additional effects may be caused by boundaries of some sort. If the shape of the object is known, it is sufficient to provide one of its dimensions in order to comprehensively describe it as long as the other dimensions are kept to linear scale. We shall call that (rather arbitrarily selected) dimension the typical length l. Furthermore, for the stationary flow the relative velocity of the fluid around that object is constant, which allows us to assign to it the scalar parameter u. Together with the kinematic viscosity ν as an intrinsic property of the material fluid we have now gained three scalar parameters to describe our problem. We emphasize at that point that we have not further specified the nature of the hydrodynamical problem at hand. Thus any possible configuration is covered by this approach, the caveat being that the fluid is viscous and incompressible and the problem stationary.

To eliminate absolute measures in our classification and establish a relation between the three parameters, we take a look at their units

$$[\nu] = \frac{\mathrm{m}^2}{\mathrm{s}}\,, \qquad [l] = \mathrm{m}\,, \qquad [u] = \frac{\mathrm{m}}{\mathrm{s}}$$

and then combine them in the only fashion as to obtain a dimensionless number,

[3] More accurately, besides the velocity we wish to gain knowledge about the ratio $\frac{p}{\rho}$ of the pressure $p\,(\vec{r}, t)$ and the constant density ρ, as found in the Navier-Stokes equation (2.5).

[4] As the relevant information is the relative motion between fluid and object, an equivalent description would be that of the movement of our fluid around the object.

the result of which is called the *Reynolds number*:

$$\mathrm{Re} = \frac{u\,l}{\nu}\,. \tag{2.6}$$

Any other dimensionless parameter can be expressed as a function of Re. As we will proceed to measure lengths in measures of l and velocities in measures of u, we express the velocity $\vec{v}$ as follows:

$$\vec{v} = u\,\vec{f}\left(\frac{\vec{r}}{l}, \mathrm{Re}\right)\,.$$

For different setups of the same type of problem, e.g. in different sizes, the dimensionless velocities $\frac{\vec{v}}{u}$ are only then described all by the same functions of $\frac{\vec{r}}{l}$ if the Reynolds numbers are identical. Flows of the same type and with the same Reynolds number are thus *similar* in the sense that they only differ in the scale of coordinates and velocities, but not in form.

Quite analog to the velocity we can construct a new notation for the pressure p as well. Rearranging the density ρ and the velocity u for the units to fit, we get

$$p = \rho u^2\, f\left(\frac{\vec{r}}{l}, \mathrm{Re}\right)\,.$$

This kind of construction can also be done for quantities that are not even functions of the coordinates but only of the Reynolds number itself. A good example here is the force $F = \rho u^2 l^2\, f(\mathrm{Re})$.

3. A short introduction to turbulence

In the previous chapter we have derived the fundamental equations of hydrodynamics in order to be able to tackle the phenomenon known as turbulence. Despite common usage of the word, the physical properties and necessities of the turbulent flow are very well-defined. Here, we will start with the analysis of stability criteria for stationary problems and proceed via the onset of turbulence to characterize the fully developed turbulent flow.

3.1. The stationary flow

Given stationary boundaries, any hydrodynamical problem should in principle have stationary solutions of the basic equations of hydrodynamics, independent of the exact shape or Reynolds number. Although this is the case, only those solutions that are stable under small perturbations may actually be observed, be it in nature or simulation. We can formalize the stability criterion by setting up the velocity field $\vec{v}$ as a linear combination of the unperturbed stationary velocity $\vec{v}_0(\vec{r})$ and the infinitesimally small perturbation $\vec{v}_1(\vec{r},t)$ and by implementing the pressure p similarly, so that

$$\vec{v} = \vec{v}_0 + \vec{v}_1 \qquad \text{and} \qquad p = p_0 + p_1$$

satisfy the Navier-Stokes equation (2.5) and the continuity equation (2.2) for incompressible viscous fluids which are given by

$$\frac{\partial \vec{v}}{\partial t} + (\vec{v}\,\nabla)\,\vec{v} = -\frac{\nabla p}{\rho} + \nu\,\Delta\vec{v} \qquad \text{and} \qquad \operatorname{div}\vec{v} = 0 \tag{3.1}$$

respectively. Inserting $\vec{v}$ and p into the above equation leads us straight to the monstrous equations of

$$\begin{aligned} \frac{\partial \vec{v}_0}{\partial t} + \frac{\partial \vec{v}_1}{\partial t} + (\vec{v}_0\,\nabla)\,\vec{v}_0 + (\vec{v}_0\,\nabla)\,\vec{v}_1 + (\vec{v}_1\,\nabla)\,\vec{v}_0 + (\vec{v}_1\,\nabla)\,\vec{v}_1 = \\ -\frac{\nabla p_0}{\rho} - \frac{\nabla p_1}{\rho} + \nu\,\Delta\vec{v}_0 + \nu\,\Delta\vec{v}_1 \qquad \text{and} \qquad \operatorname{div}\vec{v}_0 + \operatorname{div}\vec{v}_1 = 0\,. \end{aligned} \tag{3.2}$$

Here $\vec{v}_0$ and p_0 are the velocity and pressure of the unperturbed problem and in that respect naturally satisfy equations (3.1). We may therefore subtract the entire

equations[1]

$$\frac{\partial \vec{v}_0}{\partial t} + (\vec{v}_0 \nabla)\, \vec{v}_0 = -\frac{\nabla\, p_0}{\rho} + \nu\, \Delta \vec{v}_0 \qquad \text{and} \qquad \operatorname{div} \vec{v}_0 = 0$$

from their respective counterparts in (3.2). Taking into account that the perturbation is infinitesimally small we further omit all higher orders of $\vec{v}_1$ and are left with

$$\frac{\partial \vec{v}_1}{\partial t} + (\vec{v}_0 \nabla)\, \vec{v}_1 + (\vec{v}_1 \nabla)\, \vec{v}_0 = -\frac{\nabla\, p_1}{\rho} + \nu\, \Delta \vec{v}_1 \qquad \text{and} \qquad \operatorname{div} \vec{v}_1 = 0\,. \tag{3.3}$$

It also stands to reason that $\vec{v}_1$ must vanish adjacent to fixed surfaces.

The velocity $\vec{v}_1$ and the pressure p_1 are thus fixed by a system of homogeneous linear differential equations with time-independent coefficients, the general solution to which can be expressed as a sum of particular solutions with an $\mathrm{e}^{-i\omega t}$ time dependence. The actual values of ω are not arbitrary but rather determined by the equations (3.3) and the geometry and boundary conditions of the specific problem. Since ω tends to be complex, it can be represented as $\omega = \omega_1 + i\,\gamma_1$, where ω_1 and γ_1 are real. If there exists *any* positive ω_1 at all, the infinitesimally small perturbation by $\vec{v}_1$ and p_1 will grow without limit and the original pseudo-stationary flow represented by $\vec{v}_0$ and p_0 will not be observed. Only for all $\omega_1 < 0$ will the perturbations decline and the original setup be stable.

3.2. The critical Reynolds number and the onset of turbulence

The Reynolds number allows us to classify variations in fluid velocity, typical length-scale and viscosity of a setup, as discussed for u, l and ν in section 2.4. Experimental findings clearly show that for low enough Reynolds numbers the flow past an object is stable, but when reaching a certain Reynolds number $\mathrm{Re}_{\mathrm{crit}}$ called the *critical Reynolds number*, the flow becomes unstable against infinitesimally small perturbations and the stationary solution can no longer be observed. Experiments show us that $\mathrm{Re}_{\mathrm{crit}}$ usually takes a value between 10 and 100, but as mathematical examinations of such stability problems are non-trivial there is no theoretical explanation for those values yet.[2]

[1] The term $\frac{\partial \vec{v}_0}{\partial t}$ is already equal to zero for the stationary flow represented by $\vec{v}_0$. It has been left in place for the purpose of a clearer arrangement.

[2] This section 3.2 is mainly guided by the older version of Landau and Lifshitz' fluid mechanics Landau and Lifshitz (1959) as opposed to the newer edition used otherwise.

3.2.1. Turbulence at near-critical Reynolds numbers

We will now examine the onset of turbulence for Reynolds numbers very near to the critical Reynolds number. If $\mathrm{Re} < \mathrm{Re}_{\mathrm{crit}}$, all complex frequencies $\omega = \omega_1 + i\,\gamma_1$ in the solutions of equations (3.3) have negative imaginary parts γ_1. One of these imaginary parts will be equal to zero for $\mathrm{Re} = \mathrm{Re}_{\mathrm{crit}}$ and will continue to become $\gamma_1 > 0$ once the critical Reynolds number has been passed. Since we remain in close proximity to $\mathrm{Re}_{\mathrm{crit}}$, it is safe to assume $\gamma_1 \ll \omega_1$ and to model the perturbation velocity field as

$$\vec{v}_1 = A(t)\,\vec{f}(\vec{r}) \qquad \text{with} \qquad A(t) = \mathrm{const} \cdot \mathrm{e}^{\gamma_1 t}\mathrm{e}^{-i\omega_1 t}\,, \tag{3.4}$$

where $\vec{f}$ is a complex function of the location and $A(t)$ is the complex amplitude. The term $\mathrm{e}^{\gamma_1 t}$ grows quickly with time, therefore the validity of our solution, in which we assume sufficiently small $\vec{v}_1$, is limited to a very short period of time. It would also be most unexpected if the amplitude of our perturbation would continue to grow without limit, and we will now determine that limit.

Deriving the square of the absolute value of the amplitude $|A|^2 = A\,A^* = \mathrm{e}^{2\gamma_1 t}$ with respect to time for small t in accordance to equation (3.4) gives us

$$\frac{\mathrm{d}|A|^2}{\mathrm{d}t} = 2\gamma_1\,|A|^2\,,$$

but the underlying assumptions require us to expand that to higher orders when $|A|$ increases even within reasonable limits. That series expansion first leads us to the order of three, which will always contain a temporal dependence,[3] but since we care less for the actual development of $\vec{v}_1$ and more for the resulting limit of its amplitude, we eliminate it by averaging over time. This averaging takes place on scales larger than the periodical fluctuations $\frac{2\pi}{\omega_1}$ but much smaller than the time scale of significant increase in amplitude $\frac{1}{\gamma_1}$ due to $\omega_1 \gg \gamma_1$ and also disposes of all terms of the fourth order except the one proportional to $A^2\,A^{*2} = |A|^4$. With the Landau constant α accounting for the proportionality[4] we are left with

$$\overline{\frac{\mathrm{d}|A|^2}{\mathrm{d}t}} = 2\,\gamma_1\,|A|^2 - \alpha\,|A|^4$$

as the expansion up to the fourth order. Because the timescale of the average is insignificantly small compared to $\frac{1}{\gamma_1}$, the averaged amplitudes are no different from the unaveraged ones and we will denote and treat them the same way. Thus we

[3] When expanding in A and A^*, only summands with matching numbers of both A and A^* and thus even powers of $|A|$ have the potentially dominant exponential factor $\mathrm{e}^{-i\omega_1 t}$ cancelled out. The time derivative of summands of uneven powers will always contain such a dependence.

[4] For an arbitrarily small perturbation to cause the instability, α needs to be larger than zero.

obtain the solution

$$\frac{1}{|A|^2} = \frac{\alpha}{2\gamma_1} + \text{const} \cdot \mathrm{e}^{-2\gamma_1 t} \qquad \xrightarrow{t\to\infty} \qquad |A|^2_{\max} = \frac{2\gamma_1}{\alpha} \; . \tag{3.5}$$

In a last step we circle back to the Reynolds number as our classification of turbulence. To replace the usually unknown parameter γ_1 and the constant α, we expand γ_1 in orders of $\mathrm{Re}-\mathrm{Re}_{\mathrm{crit}}$. In the previously assumed proximity to the critical Reynolds number, the imaginary part of the complex frequency ω is approximately

$$\gamma_1 = \text{const} \cdot (\mathrm{Re} - \mathrm{Re}_{\mathrm{crit}}) \; .$$

This linearization satisfies our initial requirements that $\gamma_1 = 0$ for $\mathrm{Re} = \mathrm{Re}_{\mathrm{crit}}$. We insert it into equation (3.5) and finally get the relation

$$|A|_{\max} \sim (\mathrm{Re} - \mathrm{Re}_{\mathrm{crit}})^{\frac{1}{2}} \; . \tag{3.6}$$

3.2.2. Turbulence at over-critical Reynolds numbers

The previous considerations of this section hold true for turbulence with Reynolds numbers near $\mathrm{R}_{\mathrm{crit}}$, and we can regard the flow as the superposition of the stationary flow $\vec{v}_0\,(\vec{r})$ and the periodic flow $\vec{v}_1\,(\vec{r},t)$ whose amplitude at changing Reynolds is determined by the above equation (3.6). Similarly to equation (3.4) we write

$$\vec{v}_1 = \vec{f}_1(\vec{r})\,\mathrm{e}^{-i(\omega_1 t+\beta_1)} \; ,$$

introducing the initial phase β_1 that is only determined by random fluctuations of the initial conditions. This randomly set phase is a degree of freedom that is not covered by the equations of hydrodynamics in chapter 2 and their initial conditions. Thus the turbulent flow has a degree of freedom where the stationary flow has none. If the degree of perturbation gets too high, however, a separation of the stationary flow $\vec{v}_0$ and the periodic flow $\vec{v}_1$ is no longer justified and we only observe the periodic flow with a phase of $\varphi_1 = \omega_1 t + \beta_1$ and a period of 2π. The Fourier expansion of this turbulent flow at higher Reynolds numbers leads to

$$\vec{v} = \sum_{p\in\mathbb{Z}} \vec{A}_p(\vec{r})\,\mathrm{e}^{i\varphi_1 p} \tag{3.7}$$

where not only the fundamental frequency ω_1 is featured but also its integer multiples.

This periodic flow in turn becomes unstable if the Reynolds number is further increased just enough. We may then again separate $\vec{v}$ into a now periodic basic flow $\vec{v}_0\,(\vec{r},t)$ with the frequency ω_1 and a perturbation $\vec{v}_2\,(\vec{r},t)$ and insert it into

equations (3.1). In analogy to (3.3) we drop the second order terms of $\vec{v}_2$ and obtain basically the same differential equations, with the difference that the coefficients are now time-dependent with a period of $\frac{2\pi}{\omega_1}$. With the frequencies $\omega = \omega_2 + i\gamma_2$ being determined by those equations we again construct

$$\vec{v}_2 = \vec{f}_2(\vec{r}, t) \cdot \mathrm{e}^{\gamma_2 t}\mathrm{e}^{-i\omega_2 t} \ .$$

The function f_2 contains the periodicity, and as before a positive imaginary part γ_2 of the perturbation frequency ω leads to the periodic flow's absolute instability against infinitesimally small perturbations. Since the same criteria apply as for our considerations in the previous section, we may cut straight to the result and conclude that the resulting flow will be quasi-periodic with two different frequencies ω_1 and ω_2 and have two degrees of freedom due to an additional randomly set phase where before there was one.

The very similar approaches to the occurrence of the first and second periodic flow $\vec{v}_1$ and $\vec{v}_2$ suggest that more instabilities and thus more levels of periodic behavior may follow, and indeed they do. Following ever more closely onto the last periodic flow than their progenitors did and occurring over ever shorter distances, any number of additional flows $\vec{v}_n$ may develop. This is called *turbulence*. The only limit usually imposed is when the lengthscale of thermal dissipation is reached and the kinetic energy is thus taken out of the system via friction and heat transfer at the lower end of the energy cascade.[5]

We may formalize this knowledge in the following equation that poses a generalized form of equation (3.7). For n different frequencies ω_j and thus n different phases $\varphi_j = \omega_j t + \beta_j$ we expand the velocity as follows:

$$\vec{v}\,(\vec{r}, t) = \sum_{p_1, p_2, \ldots, p_n} \vec{A}_{p_1,\ldots,p_n}(\vec{r}) \ \cdot \ \exp\left[-i \sum\nolimits_{j=1}^{n} p_j \varphi_j\right]$$

With increasing Reynolds numbers and thus increasing numbers of fundemental frequencies ω_j and initial phases β_j, the number of degrees of freedom increases likewise. If Re tends to infinity, so does the number of degrees of freedom n.

The velocity $\vec{v}$ is, through its dependence on t, a function of the phases φ_j and thus periodic in 2π. This makes states with the only difference being additional integer multiples of 2π in φ physically indistinguishable from each other and practically limits the relevant range of all the phases to $0 \leq \varphi_j < 2\pi$. Since any two frequencies ω_1 and ω_2 are in general incommensurable, the corresponding reduced phases will come arbitrarily close to any possible values for the phases φ_1 and φ_2 simultaneously, given a long enough time span. This holds for all frequencies and phases, of course, and is a quasi-periodic property of the turbulent flow.

[5] This will be discussed in greater detail in section 3.3.1 ff. of this thesis.

3.3. Fully developed turbulence

At sufficiently high Reynolds numbers, the flow is characterized by unordered and irregular changes of the velocity both over time in any given point of the flow, and between any two points at a given time. This is commonly referred to as *fully developed turbulence*. In the previous section we have seen that the turbulent flow has a very large number of degrees of freedom, each of them corresponding to a randomly set initial phase β_j. Although in theory the complete knowledge of all the phases in addition to the other properties of the setup[6] would allow for a completely deterministic description and prediction of the turbulent flow, this is not only impractical but even physically meaningless. Instead, it is possible to choose a statistical approach similar to the one in classical thermodynamics. Because the fluid comes arbitrarily close to any possible states represented by the combined phase $\varphi_j = \omega_j t + \beta_j$, given enough time the exact initial conditions with regard to β_j will cease to matter, much as the many different initial locations and impulses of the atoms and molecules of a solid body will, in the end, not determine its overall state. We will now further elaborate on this statistical behavior.

3.3.1. The turbulent cascade

The velocity $\vec{v}$ fluctuates around some mean velocity $\vec{u}$ that we obtain through averaging over long intervals of time at every point in the flow. This mean velocity has lost its turbulent character and transitions smoothly from one point to another, though the difference between both velocities $\vec{v}' = \vec{v} - \vec{u}$ still varies irregularly as is expected from turbulence. It is a superposition of motions on different lengthscales, the first of which form on the largest scales and are comparable in size with the typical length l of the overall setup described in section 2.4. Although subsequently smaller eddies will appear and turbulent elements with a wide range of sizes are present at high Reynolds numbers, the largest motions still dominate the flow as their velocities are of the same order of magnitude as the change of the mean velocity Δu over the distance l, if not just as high. It stands to reason that the largest eddies will not be exactly as large as the outer dimensions of the flow l, but a bit smaller, as are the corresponding velocities.[7] Smaller eddies can be regarded as additional turbulent detail superimposed onto those larger structures, and they contain a considerably smaller fraction of the overall kinetic energy.

In our previous considerations we have used the Reynolds number as a means

[6] Other properties typically are the velocity distribution $\vec{v}(\vec{r}, t)$ and two thermodynamical quantities like the pressure $p(\vec{r}, t)$ and the density $\rho(\vec{r}, t)$, as previously described in section 2.1.

[7] Please note that the change in mean velocity Δu is not to be confused with the mean velocity $\vec{u}$ itself, since the former is mainly influenced by larger turbulence induced changes in velocity and is independent of arbitrary choices of the reference system, whereas the latter is not.

of classification for the whole hydrodynamical setup or flow as defined in section 2.4. Analogously to equation (2.6) we shall now define the Reynolds numbers of individual eddies as

$$\mathrm{Re}_\lambda \sim \frac{v_\lambda \, \lambda}{\nu} \, ,$$

where λ is the typical length of the eddy, v_λ its typical velocity and ν the viscosity of the fluid. Since λ and v_λ are comparable to l and Δu, large Reynolds numbers of the whole turbulent flow will result in large Re_λ of the major turbulent elements. This in turn suggests that the viscosity plays no role in the motion on larger scales, and that the dissipation of the kinetic energy must take place elsewhere, namely on scales small enough for the Reynolds number to be $\mathrm{Re}_\lambda \sim 1$. We deduce that the energy fed to the turbulence on large scales (e.g. through turbulent driving[8]) cascades down to the smallest scales almost without loss to get dissipated there.

As the viscosity ν is of no consequence for eddies much larger than the typical lengthscale of dissipation λ_0, changing ν to any reasonable value while keeping the general setup fixed should not affect any of the other quantities. What's more, we can even gauge the total dissipated energy just by using those quantities on the largest scales. The unit of the average energy dissipated per unit weight and time is

$$[\varepsilon] = \frac{\mathrm{J}}{\mathrm{kg} \cdot \mathrm{s}} = \frac{\mathrm{m}^2}{\mathrm{s}^3} \, ,$$

and with the typical length again $\sim l$ and the velocity $\sim \Delta u$, there is only one combination of the two to get the same dimensions as ε, which is

$$\varepsilon \sim \frac{(\Delta u)^3}{l} \, . \tag{3.8}$$

Therefore, despite its energy being dissipated at λ_0 the finer structures of a turbulent flow are not relevant to the magnitude of its dissipation.

3.3.2. Local Kolmogorov turbulence

We move on to lengthscales significantly smaller than the outer dimensions l but still larger than the dissipation lengthscale λ_0. In this so-called *inertial subrange*, the relevant physical quantities are the density of the fluid ρ, the size of the turbulent elements λ and the dissipated energy ε, since the latter determines the available energy on all scales of the turbulent cascade. Both l and Δu as well as ν have no influence here since $\lambda_0 \ll \lambda \ll l$. Furthermore, to avoid possible effects of boundary conditions we are choosing areas of our flow far enough (with respect to λ) away from any solid surfaces and thus can safely assume homogeneity and isotropy of the

[8] Turbulent driving, amongst other topics, will be discussed in the more applicatory section 4.3.

turbulence in relation to the underlying ground flow.

To obtain a measure of the typical velocity v_λ we combine the aforementioned relevant quantities so that the result has the dimension of a velocity. There is only one way to do that, which is

$$v_\lambda \sim (\varepsilon\,\lambda)^{\frac{1}{3}}\ . \tag{3.9}$$

This is *Kolmogorov and Obukhov's law of turbulence*. An important conclusion from this law is that the velocity variation v_λ is proportional to the cube-root of the associated lengthscale, which confirms our previous insight that the bigger the turbulent element, the greater its contribution to the overall velocity variation. The density ρ on the other hand does not have any effect on v_λ since it is the only quantity to feature the mass unit in its dimension, with $[\rho] = \frac{\mathrm{kg}}{\mathrm{m}^3}$. The length λ obviously does not contain any refernce to the mass, and in the average energy dissipated per unit weight and time ε the mass is already accounted for and thus taken out of the equation: $[\varepsilon] = \frac{\mathrm{J}}{\mathrm{kg{\cdot}s}} = \frac{\mathrm{m}^2}{\mathrm{s}^3}$.

Another much more famous notation of Kolmogorov's law in spectral form using the wave number $k \sim \frac{1}{\lambda}$ is

$$E(k)\ \sim\ \varepsilon^{\frac{2}{3}} k^{-\frac{5}{3}}\ , \tag{3.10}$$

where $E(k)$ is the kinetic energy per mass unit contained in a fluctuation of the magnitude k in the given range of $\mathrm{d}k$. It is derived through consideration of the involved units, as well: The unit of $E(k)$ is given by $[E] = \frac{\mathrm{m}^3}{\mathrm{s}^2}$, and the only combination of ε and k to attain it is given in equation (3.10). As with ε, the term does not contain the unit of weight, so a dependency on the density ρ is out of the question. The above equation can easily be connected to the notation of equation (3.9) through integration:

$$\int_k^\infty E(k)\,\mathrm{d}k\ \sim\ \frac{\varepsilon^{\frac{2}{3}}}{k^{\frac{2}{3}}} \sim (\varepsilon\lambda)^{\frac{2}{3}} \sim v_\lambda^2$$

The correlation to the square of the velocity v_λ is legitimate as it is the dominant factor in the term $E_{\mathrm{kin}} = \frac{1}{2} m v^2$ for the kinetic energy.

Besides this variation in velocity over certain distances at a fixed point in time, we are interested in the velocity variation v_τ at fixed coordinates during certain time spans τ that are small in comparison to the overall timescale of the flow $T \sim \frac{l}{u}$. Due to the fluid's mean velocity u, a volume element will have moved the distance of about τu in the meantime, which is equivalent to the distance λ as we assume the mean velocity to be constant for timespans $\tau \ll T$. With equation (3.9) we then obtain

$$v_\tau \sim (\varepsilon\tau u)^{\frac{1}{3}}\ .$$

To illustrate the self-similar nature of Kolmogorov turbulence, both v_λ and v_τ can be transformed using the relation for the energy dissipation (3.8) so that

$$\frac{v_\lambda}{\Delta u} \sim \left(\frac{\lambda}{l}\right)^{\frac{1}{3}} \qquad \text{and} \qquad \frac{v_\tau}{\Delta u} \sim \left(\frac{\tau}{T}\right)^{\frac{1}{3}} . \tag{3.11}$$

We see that the quantities characteristic for small scale turbulence are only distinguished by the scales they are measured on, but behave the same way.

3.3.3. Energy dissipation on small scales

Now we will take a quick look at the lower end of the turbulent cascade. At and below the characteristic lengthscale λ_0, which is often referred to also as the *internal scale*, the viscosity ν gains significant influence on the dynamics of the fluid. To put that influence into relation it is helpful to construct the local Reynolds number Re_λ in dependence on the overall Reynolds number[9] $\mathrm{Re} \sim \frac{\Delta u\, l}{\nu}$. Then using the self-similarity in Kolmogorov turbulence in (3.11) we evolve

$$\mathrm{Re}_\lambda \sim \frac{v_\lambda\, \lambda}{\nu} \sim \frac{\Delta u\, \lambda^{\frac{4}{3}}}{\nu\, l^{\frac{1}{3}}} \sim \frac{\Delta u\, l}{\nu} \cdot \frac{\lambda^{\frac{4}{3}}}{l^{\frac{4}{3}}} \sim \mathrm{Re} \cdot \left(\frac{\lambda}{l}\right)^{\frac{4}{3}} .$$

In combination with our criterion for the relevance of viscosity with regard to the inertial forces $\mathrm{Re}_{\lambda_0} \sim 1$ we are able to determine the lengthscale of dissipation λ_0 and the corresponding velocity v_{λ_0} to be

$$\lambda_0 \sim \frac{l}{\mathrm{Re}^{\frac{3}{4}}} \qquad \text{and} \qquad v_{\lambda_0} \sim \frac{\Delta u}{\mathrm{Re}^{\frac{1}{4}}} \tag{3.12}$$

respectively. Both decrease with increasing overall Reynolds number of the flow.

Finally, at lengthscales much smaller than λ_0 the flow becomes regular again since the local Reynolds number is sufficiently small to dissuade any turbulent behavior on that scale as the viscous forces truely dominate over the inertial forces. We can develop a measure for the velocity v_λ by assuming that $\frac{v_\lambda}{\lambda} \sim \frac{v_{\lambda_0}}{\lambda_0}$, which is reasonable for the non-turbulent flow, and with the help of equation (3.12) we get

$$v_\lambda \sim \frac{v_0}{\lambda_0}\lambda \sim \frac{\Delta u}{l}\lambda\, \mathrm{Re}^{\frac{1}{2}} .$$

[9] The original definition of the overall Reynolds number was $\mathrm{Re} = \frac{u\, l}{\nu}$ in equation (2.6). As before in this section, though, the velocity of the large scale turbulence is not proportional and of the same order of magnitude of the actual mean velocity u, which depends on the choice of the frame of reference. Instead, it relates to the fluctuation in the mean velocity on distances comparable to the external scale l, which is exactly Δu.

3.3.4. The power spectrum

In the previous section we have outlined various properties of turbulence on different scales. A most comprehensive way to analyze all the data of an experiment or simulation, for example, with regard to its validity is to plot its kinetic energy $E(k)$ against the corresponding wave number $k \sim \frac{1}{\lambda}$. Such a power spectrum allows us to identify the different subranges and test their compliance with our theoretical expectations. For fully developed turbulence, it can be naturally divided into three different subranges:

- *The energy subrange*: The largest scales with $\lambda \sim l$. They contain most of the turbulent energy as described in section 3.3.1.
- *The inertial subrange*: The median scales with $l \gg \lambda \gg \lambda_0$. Since the inertial forces dominate this part of the spectrum, the energy is handed further and further down via Kolmogorov turbulence, as discussed in section 3.3.2.
- *The dissipation subrange*: The smallest lengthscales with $\lambda \lesssim \lambda_0$. Located at low end of the spectrum, this is where the dissipation of the turbulent energy occurs, as addressed in the previous section 3.3.3.

Figure 3.1 symbolically depicts these three subranges in an idealized power spectrum plot.

We have already gained insight into the specific power spectrum of the Kolmogorov part of our turbulence, which under ideal conditions is easy to identify in a log-log plot as a straight line due to $E(k) \sim k^{-\frac{5}{3}}$. We expect the inertial subrange of any turbulence to verge on this line as soon as the turbulence is fully developed. It is important to note that the namesake $-\frac{5}{3}$ slope is the result of one-dimensional considerations, though. To obtain the power law for three dimensions, we have to take into account that $E(k)$ in equation (3.10) is the kinetic energy per mass unit contained in turbulent motions over the one-dimensional distance $\lambda \sim \frac{1}{k}$. In three dimensions, $E(k)$ must relate to turbulence in the volume $\lambda^3 \sim \frac{1}{k^3}$, which leads us to the relation

$$E(k)_{\text{3D}} \sim k^{-\frac{11}{3}} \ , \tag{3.13}$$

the Kolmogorov power law for fully developed three-dimensional turbulence.

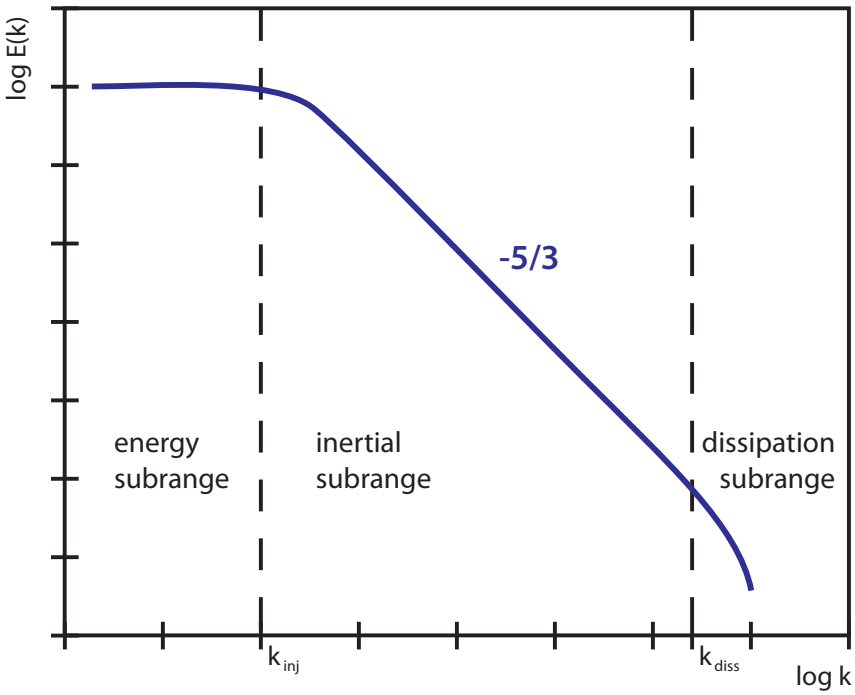

Figure 3.1.: Representation of the idealized velocity power spectrum and its different subranges. k_{inj} denotes the end of the injection (or energy) subrange, and k_{diss} the beginning of dissipation. Due to the logarithmic scaling of the axes, the eponymous $-\frac{5}{3}$-turbulence of the inertial subrange is easy to recognize as the straight declining section of the powerspectrum.

4. Essentials of turbulent SPH simulations

The previous chapters have helped us define the concept of turbulence. Since to this day no self-contained analytical theory of turbulence exists, a common approach to solving turbulent problems is simulation. This is especially true when setting up experiments is out of the question, which is the case for most astrophysical problems. With the theoretical groundwork done, we will now turn to different methods of simulating turbulence and analyzing the data. Although we will use smoothed particle hydrodynamics (SPH) as the principal simulation technique in this thesis, we will first briefly introduce two other major simulation techniques and only then expand on the topic of SPH. After that, we will give a short overview of the general setup of our simulations and of the data processing, and subsequently have a look at the methods of evaluating and visualizing the data.

4.1. Alternative simulation techniques

Our theoretical considerations have been based on the assumption that we may divide any fluid into volume elements small enough to seem infinitesimally small compared to the relevant lengthscales of the problem, but still large enough for microscopic effects to be negligible. The two different approaches to simulating the turbulent flow presented in this section implement those two properties: the overall volume is divided into a high number of volume elements, and exact microscopic behavior is left out of the simulation.[1] This makes the volume elements the smallest items to consider, and a major difference between the simulation techniques is how those volume elements are obtained and handled.

4.1.1. The Cartesian grid

The most straightforward approach to discretizing the continuous flow of a given setup is to put a fixed Cartesian grid onto it. The advantage is its simplicity in

[1] While the microscopic behavior is not usually explicitly considered, macroscopic properties as an effect of microscopic effects such as the temperature T, the pressure p, the density ρ and other state variables may very well be calculated.

design: Since the grid doesn't move, many computional operations elsewise needed just to maintain a useful discretization are unnecessary. The immobility of the mesh brings about its very own problems, though. First of all, such a simulation is not Galilei-invariant and is thus affected by possible bulk motions with large masses crossing in and out of individual cells at a high rate. Second, the spatial resolution is constant over time and space, so to obtain a high enough resolution for simulations that sport high density contrasts and dynamics over several orders of magnitude, the computional effort and cost increases at least with the third power of the number of grid cells per dimension N_{grid}. Both situations are frequent in astrophysical simulations, e.g. those of cosmological structure growth.[2] There is the possibility of artificially reducing the spatial resolution in areas of less interest and of increasing it where needed to accurately follow the physical processes involved. Albeit this adaptive mesh refinement (AMR) remedies the latter of the two problems described above, it brings about other problems, the most prominent one of them being the need for an effective automatic adaption routine predicting where higher or lower future resolutions will be required.

4.1.2. Moving-mesh simulations

A further improvement of the Cartesian grid approach is to devise a mesh that is adaptive in both shape and spatial resolution as well as being comoving with the local flow. The latter eliminates the problem of Galilei-invariance, since the relative motion between the mesh cells and the flow of mass is kept to a minimum, whereas the first is necessary to adapt the mesh to the ever-changing demands of e.g. a turbulent flow, increasing the effective resolution of a simulation while maintaining the same overall number of cells. Usually, areas of higher mass density constitute areas of greater importance. For example, in a simulation of loosely distributed gas converging someplace to form an accretion disc, the areas of higher mass density in and around the disc are dynamically more active and require a higher number of cells to be properly resolved. On the other hand, the outlying sectors that are gradually being drained of gas and whose density decreases accordingly usually feature much less dynamics and physical interaction and need not be resolved as accurately. A higher effective resolution can then be achieved by keeping the mass inside a cell as constant as possible while at the same time minimizing the flow of mass through the boundaries of any given cell as well, which will effectively result in the cells following the mass to areas of higher density and better resolving the processes there.

A key part of any moving-mesh code is the scheme used to set up and maintain a sensible mesh. In the case of Arepo, the moving-mesh code by Volker Springel

[2]For more information on the various advantages and disadvantages of different hydrodynamic codes, please cf. Tasker et al. (2008) and Wadsley et al. (2008).

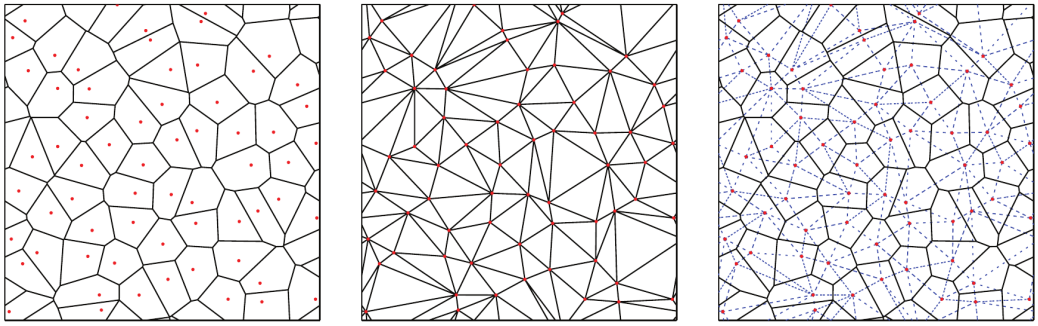

Figure 4.1.: Voronoi (left) and Delaunay (middle) tessellation of a two-dimensional periodic box. The Delaunay tessellation provides a complete set of perpendicular bisectors needed for the construction of the irregular moving-mesh.
Image credit: Springel (2010a)

used for some comparison runs in this thesis, a Voronoi tessellation is invoked. This basically constitutes a multitude of points, each of which is assigned the region of space around them that is closer to them than it is to any of the other points. Figure 4.1 shows how this tessellation can be achieved via the perpendicular bisectors of any two neighboring points in two dimensions. In the initial setup, those points are distributed across the volume of the simulation such that any cell contains the same mass irrespective of its shape and size, although the density $\rho = \frac{M}{V}$ may vary accordingly. The whole cell is represented by one point particle so that in the run of the simulation, the hydrodynamical calculations are performed on those particles rather than on volumes of the fluid.[3]

4.2. Smoothed Particle Hydrodynamics

The bulk of the simulations in this thesis were performed with the help of smoothed particle hydrodynamics. This Lagrangian simulation scheme was first devised by Lucy (1977) and Gingold and Monaghan (1977), and has been utilized and refined by many numerical astrophysicists ever since. More specifically, we used GADGET-3, a widely used SPH code. We will review the key features and the resulting advantages and disadvantages of SPH below.

[3]This is just a very brief scetch of the design of a functional moving-mesh scheme. For detailed information on the matter please cf. Springel (2010a), the original AREPO proposal paper.

4.2.1. The basic concept of SPH

In contrast to the grid or mesh based simulation schemes described above, smoothed particle hydrodynamics is not based on a clear split of the continuous fluid into seperate cells. It is fairly clear, though, that a discretization is necessary for numerical processing because the resolution of our simulation, no matter its complexity, will be finite. So instead of clean-cut divisions we use smoothed particles to represent our fluid and move with the flow. Those particles have fixed mass and are each represented by a central point from which a so-called *smoothing kernel* W extends a weighted mass function that specifies how much of the particle mass m_j is present at which distance $|\vec{r} - \vec{r}_j|$ from the central point of particle j. This simply allows us to compute the density $\rho(\vec{r})$ at any arbitrary point of our simulation via a kernel weighted sum

$$\rho(\vec{r}) = \sum_{j=1}^{N_{\mathrm{ngb}}} m_j \, W(|\vec{r} - \vec{r}_j|, h) \; , \tag{4.1}$$

N_{ngb} being the number of smoothing neighbors and h the smoothing length. The latter determines how far the kernel extends the "smoothed" influence of the particle at hand. A plausible choice for the smoothing kernel W is the three-dimensional Gaussian, given by $W = 1/(\sqrt{\pi}h)^3 \cdot \exp(-r^2/h^2)$; in practice its non-zero nature would be hindering, so close approximations with a finite non-zero range are used.[4]

A set number of smoothing neighbors N_{ngb} guarantees every point in space to have a sufficient overlap of weighted mass from different hydrodynamic particles. Due to possibly very different spacing of the particles in the face of a wide dynamic range, individual and variable smoothing lengths h_j for every particle are usually necessary. The individual smoothing lengths will then adapt according to the following equation

$$\frac{4\pi}{3} h_j^3 \, \rho_j = N_{\mathrm{ngb}} \, \overline{m} \; , \tag{4.2}$$

with $\overline{m}$ the average mass of a particle.

Since the particle mass m_j of every single particle is permanently fixed, the effective resolution follows the mass, as the mass density ρ of a local volume can only increase if the number of local particles does, too. Another benefit of $m_j = \mathrm{const}$ is the exact conservation of the total mass regardless of the simulation. Even more, from the density sum (4.1) we can easily derive the Lagrangian

$$L_{\mathrm{sph}} = \sum_j m_j \left[\frac{1}{2} v_j^2 - u_j(\rho_j, s_j) \right] \; ,$$

[4] For more information on different kernels and a good and extensive review of SPH, please cf. Price (2012b).

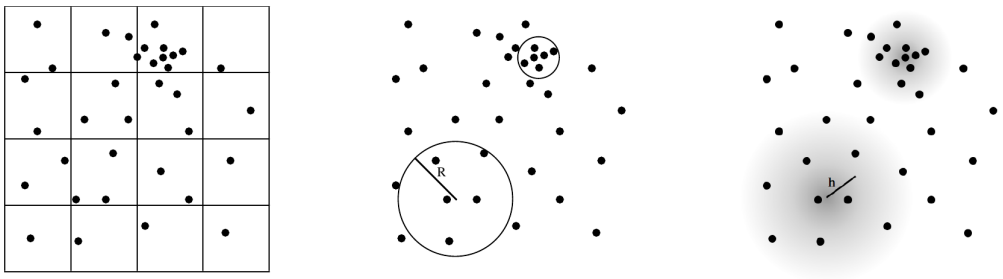

Figure 4.2.: Different methods to compute a continuous density from point mass particles are: the particle-mesh method (left), constructing a local volume according to the local density of particles (middle), and the approach adopted in smoothed particle hydrodynamics (right).
Image credit: Price (2012b)

with v_j the particle velocity and u_j the specific internal energy as a function of the density ρ_j and the entropy s_j of each particle. Using the Euler-Lagrange equations

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \vec{v}}\right) = \frac{\partial L}{\partial \vec{r}} \tag{4.3}$$

and the first law of thermodynamics, we obtain the equations of motions for each particle i as

$$\frac{\mathrm{d}\vec{v}_i}{\mathrm{d}t} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W(|\vec{r}_i - \vec{r}_j|, h) \ ,$$

which is the comoving version of Euler's equation (2.3) in a discretized form. This is the simplest form of the equations of motion as proposed by Monaghan (1992), since we have assumed a uniform and constant smoothing length h. For variable and different smoothing lengths h_i and h_j, corrections of the form

$$g_i = \left(1 + \frac{h_i}{3\rho_i}\frac{\partial \rho_i}{\partial h_i}\right)^{-1}$$

to the $\frac{p_i}{\rho_i^2}$ terms (for j respectively) are needed. The result is a more accurate equation of motion

$$\frac{\mathrm{d}\vec{v}_i}{\mathrm{d}t} = -\sum_j m_j \left(g_i \frac{p_i}{\rho_i^2} \nabla_i W_{ij}(h_i) + g_j \frac{p_j}{\rho_j^2} \nabla_j W_{ij}(h_j)\right) \ ,$$

where $W(|\vec{r}_i - \vec{r}_j|, h)$ has been abbreviated as $W_{ij}(h)$. A more detailed description can be found in Springel and Hernquist (2002) and Springel (2010b).

Some more key features of SPH as a result of the above equations include an exact,

time-independent solution to the continuity equation, the perfectly done advection, the absence of intrinsic dissipation, the exact and simultaneous conservation of both linear and angular momentum as well as energy and entropy, and a guaranteed minimum energy of the particles. The implications of these features are important to a fully-fledged understanding of smoothed particle hydrodynamics.[5] We will further elaborate on the lack of intrinsic dissipation and on the need for an artificial viscosity in the following section.

4.2.2. Artificial viscosity

Since SPH is completely Lagrangian, dissipation has to be implemented "artificially". This *artificial viscosity* should not be mistaken for some arbitrary, unphysical addition to bring the simulation closer to our expectations. On the contrary, the need for artificial viscosity is a direct result of our use of the Euler-Lagrange equations (4.3), where we have implicitly assumed that the quantities in the Lagrangian are differentiable and continuous. That is clearly not the case when shocks and other discontinuities occur, and a special treatment is needed, though not only from a numerical point of view, but also from a strictly physical one.[6]

The artificial viscosity term is usually implemented as a viscous force:

$$\left.\frac{\mathrm{d}\vec{v}_i}{\mathrm{d}t}\right|_{\mathrm{visc}} = -\sum_j m_j \Pi_{ij} \nabla_i \overline{W}_{ij} \ ,$$

with, v_i the velocity of particle i, $\overline{W}_{ij}$ the arithmetic average of the smoothing kernels W_i and W_j and Π_{ij} the vicous tensor[7] as defined by

$$\Pi_{ij} = -\frac{\alpha}{2}\frac{(c_i + c_j - 3w_{ij}) \cdot w_{ij}}{\rho_{ij}} \ . \tag{4.4}$$

Here, c_i and c_j are the sound speed of the particles i and j, and w_{ij} is the amount of relative velocity between them in the direction towards particle j. To make sure the artificial viscosity only operates when the pair of particles is approaching each other and thus the entropy produced by it is positive definite, it is set up to be

$$w_{ji} = \begin{cases} \vec{v}_{ij} \cdot \frac{\vec{r}_{ij}}{|\vec{r}_{ij}|} & \text{if} \quad \vec{v}_{ij} \cdot \vec{r}_{ij} < 0 \\ 0 & \text{otherwise} \ . \end{cases}$$

The introduction of artificial viscosity addresses many problems of dissipation-free

[5] For a more detailed analysis of the key features please cf. Price (2012c).
[6] For a detailed review of the artificial viscosity in SPH please cf. Price (2012b).
[7] Please cf. section 2.3 for our previous elaborations on the viscous stress tensor and Landau and Lifshitz (1991) for more information on the subject of viscous tensors in general.

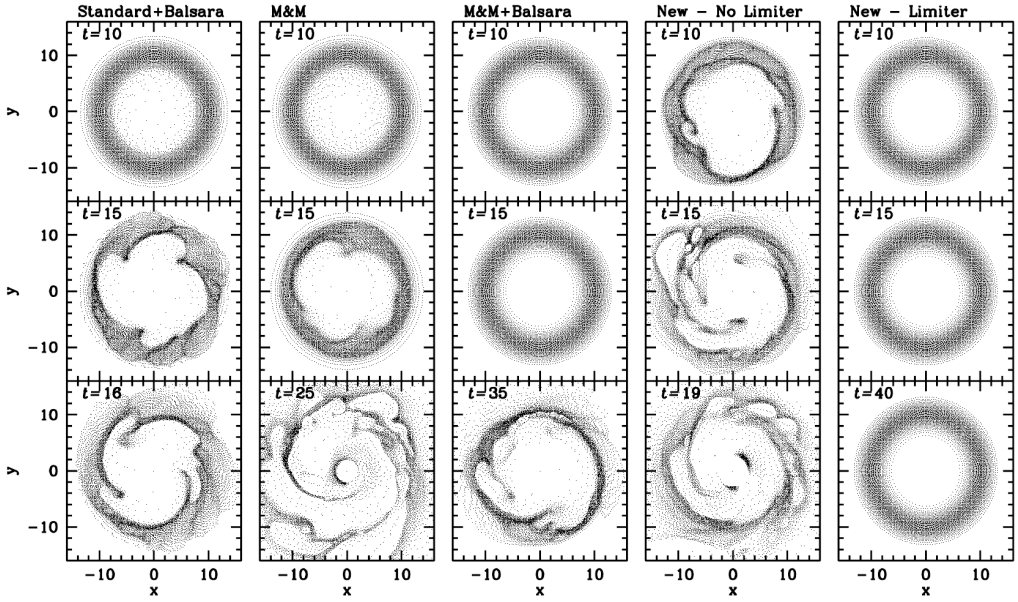

Figure 4.3.: The Keplerian ring test, a standard test against a code's treatment of fluid instabilities, here applied to various viscosity schemes. From left to right: Standard SPH with Balsara (1995) switch, the Morris and Monaghan (1997) method without and with Balsara switch, and the Cullen and Dehnen (2010) scheme without and with their new limiter. Image credit: Cullen and Dehnen (2010)

SPH, however some new problems are introduced. The most prominent one is that the constant artificial viscosity parameter α can lead to questionable viscosity away from shocks, dissolving structure on scales much larger than the effective resolution and inhibiting the generation of turbulence from fluid instabilities. This is well illustrated by the Keplerian ring test in figure 4.3.

Therefore, an important modification is the so-called Balsara switch.[8] It is used to turn down the artificial viscosity in regions of strong shear and thereby prevents the unwanted transport of angular momentum induced by a constant artificial viscosity parameter α. A convenient criterion for that regulation is the vorticity of the flow, as it is a measure for its rotation. The switch is constructed to not affect the artificial viscosity when the particles move fairly straight ahead, meaning the divergence $|\nabla \cdot \vec{v}\,| \approx \max$, but to gradually reduce it the greater the vorticity $|\nabla \times \vec{v}\,|_i$ gets. For particle i, these considerations lead us to

$$f_i = \frac{|\nabla \cdot \vec{v}\,|_i}{|\nabla \cdot \vec{v}\,|_i \;+\; |\nabla \times \vec{v}\,|_i} \,.$$

As the viscous tensor Π_{ij} always concerns two particles, the above expression is

[8]For the original proposal, please cf. Balsara (1995).

calculated for both particles and the median $\frac{f_i+f_j}{2}$ is taken and gets multiplicatively adjoined to the α in equation (4.4). The Balsara switch is a standard feature of our SPH code.

Another way to improve the performance of SPH with regard to its artificial viscosity is to assign an individual artificial viscosity parameter α_i to every particle. Its time dependent evolution is defined by

$$\frac{\mathrm{d}\alpha_i}{\mathrm{d}t} = -\frac{\alpha_i - \alpha_{\min}}{\tau} + S_i \ ,$$

where $\alpha_{\min}$ is the minimum value of α_i, τ is the decay time scale that determines how many sound crossing times it takes for the viscosity to decay in smooth regions of the flow, and S_i is a source term that swiftly increases α_i when the particle approaches a shock. The fixed α in equation (4.4) is then replaced by the median $\frac{\alpha_i+\alpha_j}{2}$ of both involved particles' time-dependent parameters. This scheme was first proposed by Morris and Monaghan (1997), and prominently introduced into the GADGET-2 code by Dolag et al. (2005). Another prominent switch of similar intent is the scheme proposed by Cullen and Dehnen (2010), possibly constituting the state of the art in the field as they not only aim to reduce the artificial viscosity away from shocks but to eliminate it. We ourselves have used the brand-new, not yet public artificial viscosity scheme by Alexander Beck.[9]

A deliberate analysis of various kinds of discontinuities in SPH and the right ways to treat them may be found in Price (2008).

4.3. Setting up initial conditions

Usually, the correct setup of the initial conditions (ICs) would deserve a whole chapter of its own. We will stick to the basics, though, and highlight the crucial parts of the process to give an overall idea of the methods and problems involved.

4.3.1. The need for a box

A simulation, irrespective of its scope, will only ever have a finite amount of computing power and thus span a finite amount of space. That confined space is usually referred to as the *box* and needs to be well defined, since the kind of box and especially its boundary conditions will significantly determine the outcome of the simulation. Three important types of boundary conditions are listed below as well as illustrated in figure 4.4:

[9]Dr. Alexander Beck, Universitätssternwarte München, Munich, Germany.

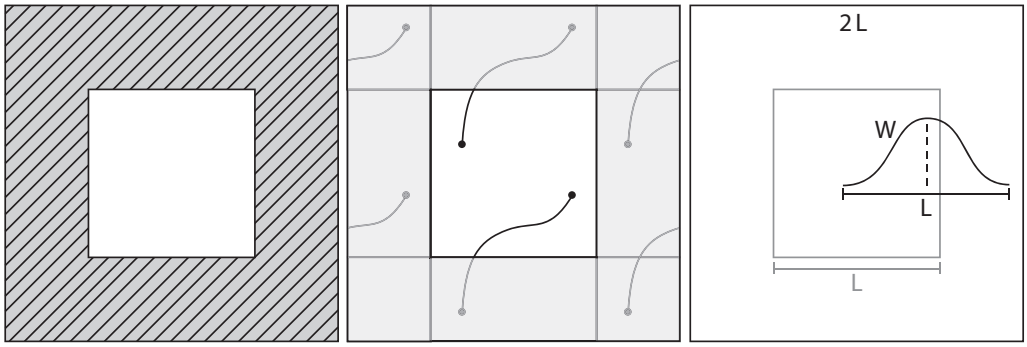

Figure 4.4.: Visualization of different boundary conditions of a box in two dimensions: A box with fixed boundaries (left), a periodic box (middle) and a box with zero-padding (right) and kernel W.

- *Fixed boundaries*: The box has solid boundaries that render it quite similar to an actual, material box. Because the walls of such a box reflect incoming particles, momentum is not conserved, though energy is.

  A typical application is classical simulations of hydrodynamic behavior where the form of the box is part of the problem and boundary effects such as wave reflection at the surface are desired.

- *Periodic boundaries*: The box acts as if it had no physical boundaries. Instead, if a particle passes out on one "end" of the box, it just "reappears" on the other side, retaining all its original properties including pulse and energy. Perhaps a more accurate interpretation of the periodic boundary conditions is that the box effectively gets mapped all around itself, but without additional computional expense.[10] So, not only do particles travel from one side to the other, but also do physical effects like gravitational fields extend past the (imperceptible) boundary into the next instance and thus into itself.

  Cosmological simulations usually use periodic boxes, namely for two reasons: firstly, because boundary effects are not desired at all, and secondly because it is usually assumed that there will be other, adjoining structures all around which are in this case emulated by the box itself without further effort.

- *Zero-padding*: The box ensures that the simulated object or region is isolated and is not subjected to external influences. To that end, what would have been the contents of the box in case of one of the above setups is placed inside

[10] In the first layer in three dimensions there would be twenty-six identical boxes, in the second layer ninety-eight boxes, and ever so on in all directions. Please note that by identical we mean that they are not only set up in the same fashion, but that those boxes exactly mirror everything in the "original" box. As a matter of fact, those boxes are indistinguishable as they are one and the same, and thus need only be represented by a single one.

a much bigger periodic box. This buffer is not empty but rather presents an "uneventful" extension of the original box. It features the same basic properties properties (e.g. the gas density, temperature, pressure etc.) to prevent unwanted dynamics in the border region, but none of the events or objects like shock fronts, massive central objects or other anomalies that initiate the physical processes we actually want to simulate.

This padding basically keeps the kernel from reaching the limits of the box while evaluating the core region and thus isolating the contents of the inner "box" from any interactions with itself. To that avail, the kernel is sized as to still fit the original box with its smaller dimensions, and explicit evaluation of the padding region is strongly dissuaded. Usually, the padding box is of twice the size per dimension, which results in serious increases of demand for memory and computional power. The increase in memory consumption alone to store a zero-padded box instead of a simple periodic box is proportional to 2^{dim}, which means that in three dimensions $\frac{7}{8}$ of the box will practically go to waste.

Simulations of isolated objects like that of a forming disk make good use of zero-padded boxes, since only the one object and not its possible interactions with some environment is of interest.

4.3.2. Units, scales and basic conditions

Traditionally, the SI system of units is not much used in astrophysics. It is both traditional and in some cases rather convenient to use the Gaussian CGS system, as e.g. the $4\pi\varepsilon_0$ in Maxwell's equations are replaced by the (dimensionless) number 1. That being said, the simulation code usually will not perform any calculations in standard CGS units, as the scales in astrophysical problems are huge in comparison with the standard units and all numerical values would need significantly more memory besides being near-unreadable. Instead, so-called *internal units* are defined, placing the value of 1 on a reasonable scale. It is necessary to only define a minimum set of internal units, because the rest can easily be constructed of them. They are given in their numerical, unitless CGS values to the simulation code, and all other input into the simulation will be in these custom units.

Besides the system of units, other specifics of the initial conditions need to be defined, too. This includes the size of the box, of course, but also such parameters as the hydrogen mass fraction H_{frac}, the temperature T, the Boltzmann constant k_B, the total mass and many more.[11]

[11]For a full review of the initial parameters, please cf. `make_data.pro` in appendix A.1.

4.3.3. Particle distribution

A large amount of code lines go into creating the initial distribution of the particles. The hexagonally close packed particle distribution (HCP) needs to be implemented accordingly, since it is naturally favored for its dense character rendering specific rearrangement motions of the particles as a consequence of the initial setup obsolete. A simple cubic distribution, for example, infuses the particles with potential energy caused by the unequal distances between close neighbors that renders the setup instable and effectively increases the kinetic energy inadvertently.

Still, HCP is not the most dense distribution achievable for a finite volume. A possibility to gain an even more relaxed initial distribution is to run the box full of HCP distributed particles with SPH and no external forces and let them "wriggle" on their own account due to the still inherent energy, then take that energy out successively until a maximally relaxed distribution is reached. The result is often referred to as a *glass file* due to its amorphous, glass-like structure. For many applications a hexagonally close packed distribution will be sufficient, though.

4.3.4. The velocity field

In the case of a turbulent setup, some thought needs to go into creating a feasible velocity field. This is best done in k-space, because the motions on various length-scales correspond to a wave number $k \sim \frac{1}{\lambda}$ regardless of their actual position in the box. As has been discussed before in section 3.3, turbulence is best characterized by the spectrum of the power contained in its impossibly complex motions on different scales rather than by any set of specific motions. Only the latter would be best described in real space, while k-space allows for the statistical distribution of velocity according to the appropriate power laws.

Setting up a complex grid

First, a velocity grid in three dimensions needs to be created that has a total of $(N_{\mathrm{Grid}})^3$ cells. It is then populated with all modes available to a box of size L. The smallest mode is the box mode k_{min}, it corresponds to motions on the largest scales in the box, and the largest mode is the Nyquist mode k_{max}, which is determined by the smallest length that can be resolved with any given spatial resolution N_{Grid} according to the Nyquist-Shannon sampling theorem.[12] They are given by

$$k_{\mathrm{min}} = \frac{2\pi}{L} \qquad \text{and} \qquad k_{\mathrm{max}} = \frac{\pi \cdot N_{\mathrm{Grid}}}{L} \; .$$

[12] For a thorough discussion of various aspects of digital signal processing in cosmology, including but not limited to this one, please cf. Jasche et al. (2009).

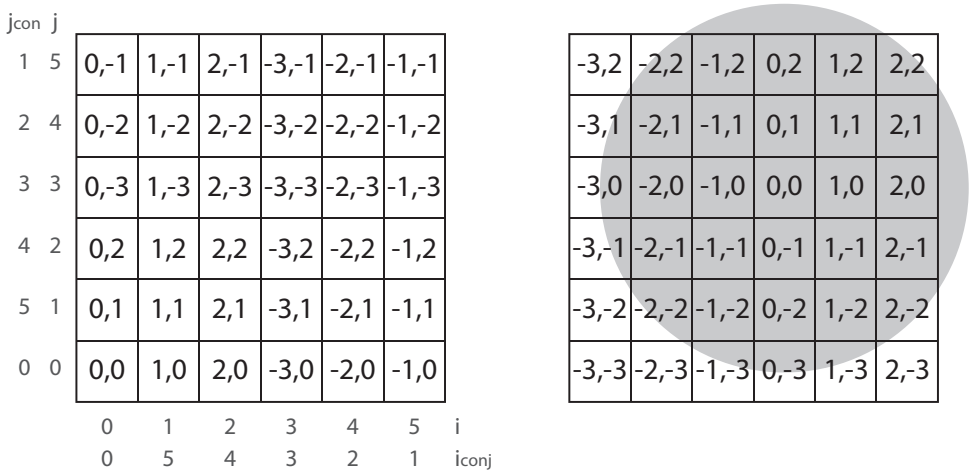

Figure 4.5.: Reduced schematic of the setup of a complex grid with $N_{\text{grid}} = 6$. Only two dimensions are depicted, with $k_z = 0$. The pair of numbers in each grid cell represent the values of k_x and k_y as multiples of k_{min}. Left: allocation of k_x and k_y according to the grid index and its conjugated counterpart. For details cf. appendix A.1. Right: reorganized grid according to k_x and k_y, so that the sphere in k-space (cf. equation (4.5)) is visible as the grey circle. In this representation, k_{mag} are only calculated for cells whose center overlaps with the circle.

All modes in between are integer multiples of k_{min}. Every grid cell has three of these one-dimensional modes referred to as k_x, k_y and k_z assigned to it in a rather elaborate fashion both to cover all relevant cases and to cut back on computational expenses through the use of symmetries. From them a cell-specific three-dimensional wave number k_{mag} is calculated so that

$$k_{\text{mag}} = \sqrt{k_x^2 + k_y^2 + k_z^2} \leq k_{\text{max}} \, . \tag{4.5}$$

Please take note that the processes described both in the current as well as the following paragraph are (at least in theory) performed for every single cell of the grid $[i, j, k]$, with $i, j, k = \{0, 1, \ldots, N_{\text{Grid}} - 1\}$. The whole scheme is depicted in figure 4.5 for a small example, for the full details please consult the source code in appendix A.1.

Computing the velocity in k-space

Next, the equivalent of the velocity field in k-space is created using the above grid. To later obtain a physically meaningful velocity field, it is of paramount importance to set up this complex field to be Hermitian, for only then will its Fast Fourier Transform (FFT) be real.

The combined wave number k_{mag} of every specific cell is used to generate the matched power $P(k)$ according to the predictions of Kolmogorov turbulence in three dimensions,[13] so that the norm

$$\int_{k_{\min}}^{k_{\max}} \mathrm{d}k \; P_0 \; 4\pi \; k^2 \, k^{-11/3} = \left[6\pi \; k^{-2/3}\right]_{k_{\min}}^{k_{\max}} \stackrel{!}{=} 1$$

is fulfilled.

This $P(k)$ in turn is used to scale a pair of pseudo-random values per cell in accord to the desired power. Actual random numbers are by concept not obtainable via computional methods, but random number generators mostly do a good job of providing reasonably unforseeable substitutes. Out of convenience, we will refer to them as "random numbers" from now on. Since we need those values to follow a Gaussian distribution, the so-called *Box-Muller transform* is invoked to generate sets of two independent standard normal distributed random numbers C_{rl} and C_{im} from two independent uniformly distributed random numbers U_1 and U_2, the latter of which are readily available via internal procedures of IDL and other programming languages. The Gaussian random numbers are defined as

$$C_{\mathrm{rl}} = A \cdot \cos\varphi \qquad \text{and} \qquad C_{\mathrm{im}} = A \cdot \sin\varphi \; ,$$

where A is the amplitude and φ is the phase. Both amplitude and phase are derived from the uniformly distributed random numbers U_1 and U_2 according to

$$A = \sqrt{-\ln\left(U_1\right) \cdot P(k)} \qquad \text{and} \qquad \varphi = 2\pi \cdot U_2 \; .$$

It has been proven by Box and Muller (1958) that the resulting numbers C_{rl} and C_{im} are independent of each other.

Also, on a side note regarding our later application of this scheme, this is where the range of modes we want to seed in our box can easily be selected. It may for example be of greater interest to let the energies cascade down thorough the different lengthscales than to seed the full spectrum. In that case, we would simply set the amplitude A to zero for all $k_{\mathrm{mag}} > k_{\mathrm{seed,max}}$, effectively depriving them of any power.

[13] In previous chapters we have referred to $P(k)$ as $E(k)$, the energy per mass unit contained in turbulent motions on the scale $\lambda \sim \frac{1}{k}$. Please cf. equations (3.10) and (3.13) and the respective section 3.3 for more information. In simulations, it is customary to simply call this quantity the *power* denoted $P(k)$.

Transformation from k- to real space

Now, as may have been guessed by their denotation, C_{rl} and C_{im} happen to be the real and the imaginary part of the velocity field in k-space, respectively. Since there is a pair of those random values for every cell in the complex grid, we may as well write

$$C_{[i,j,k]} = C_{\mathrm{rl}[i,j,k]} + i\, C_{\mathrm{im}[i,j,k]}$$

and transform them via FFT to the real space. A subsequent check of the result is in order too, since if the symmetries in k-space are just slightly off, the resulting velocity field in real space will be unphysical. Then we may resize the amplitude of the entire velocity field to agree with the desired amount of turbulent velocity as defined by the fraction of turbulent energy with regard to the total thermal energy in the box.

4.4. About GADGET

GADGET, short for **GA**laxies with **D**ark matter and **G**as int**E**rac**T**, is a massively parallel TreeSPH code. The first version was originally written by Volker Springel as part of his PhD project at the Max-Planck-Institute for Astrophysics in Garching, Germany. The second version constituted an almost complete rewrite of the original version, improving on many features and adding some more, like the TreePM scheme. The current version GADGET-3 again sports significant improvements in accuracy and functionality, but the code is not public yet. For the original papers of Version 1 and 2 please cf. Springel et al. (2001) and Springel (2005), respectively.

We have used GADGET to perform all the SPH simulations in this thesis. Since the code is very versatile, it can be adapted to quite different demands. By itself it is capable of N-body/SPH simulations and TreeSPH[14], and it has been continuously extended by numerous contributors over the years. Fields of application range from star formation, to colliding galaxies, to the formation of the large-scale structure of the universe.

[14]The TreeSPH scheme was first described by Hernquist and Katz (1989). It unites smoothed particle hydrodynamics with the hierarchical treatment of gravity.

4.5. Binning to the grid

After running the simulation, the particle distribution will in general be irregular and not follow the lines of any ordinary grid. Therefore, we need to sample the SPH data back onto a grid for evaluation, which is also referred to as *binning the data*. A selection of different methods to do this will be presented below.

4.5.1. Standard SPH binning method

The straightforward approach is to superpose the whole box with a uniform grid of high resolution and to sample the value of some desired quantity Λ at the center of every cell, henceforth denoted as $\vec{r}$. Since Λ has been calculated by e.g. GADGET for every SPH particle j and can be regarded as a property of its central point, we can use an approach similar to the one described in section 3.3.3, where we calculated the mass density $\rho(\vec{r})$ at an arbitrary point of our simulation. To this end, the quantity Λ is introduced on both sides of equation (4.1), so to say: On the left side, $\Lambda(\vec{r})$ relates to its yet unknown value at the sampling point $\vec{r}$, while on the right side, the various Λ_j correspond to the already calculated values for all particles j with the central points $\vec{r}_j$ whose smoothing kernel W makes them overlap with $\vec{r}$. Both sides of the equation are also set into relation with the mass density ρ: The left side is divided by the combined density $\rho(\vec{r})$ at the sample point, which so vanishes, while on the right side every summand is divided by the average density ρ_j of the respective particle. Thus we get

$$\Lambda(\vec{r}) = \sum_{j=1}^{N_{\mathrm{ngb}}} m_j \, \frac{\Lambda_j}{\rho_j} \, W(|\vec{r} - \vec{r}_j|, h_j) \; . \tag{4.6}$$

To achieve near-perfect results, all the cells need to be smaller than the smallest particle, which means the resolution of the grid will be especially high for simulations covering a wide dynamical range. Also, low grid resolutions will compromise the output of otherwise correctly calculated conserved quantities like the total mass.

4.5.2. Modified SPH binning methods

There are several approaches to improve the binning with regard to the high memory requirements of the standard SPH method. The aim of these methods is to yield an accuracy comparable to that of the standard method but at a lower formal resolution of the grid.

The first approach increases the number of sampling points from one per grid cell to a higher number N_{smp}. A typical choice is $N_{\mathrm{smp}} = 9$, so that the additional sampling points form a cube around the central point $\vec{r}$ and are well-spaced around

the volume of the cell. The quantity Λ is then determined as follows:

$$\Lambda(\vec{r}) = \frac{1}{N_{\mathrm{smp}}} \left[\sum_{j=1} m_j \frac{\Lambda_j}{\rho_j} W(|\vec{r} - \vec{r}_j|, h_j) + \right.$$
$$\sum_{j=1} m_j \frac{\Lambda_j}{\rho_j} W(|\vec{r} + \tfrac{a}{\sqrt{3}}(1,1,1) - \vec{r}_j|, h_j) +$$
$$\left. \sum_{j=1} m_j \frac{\Lambda_j}{\rho_j} W(|\vec{r} + \tfrac{a}{\sqrt{3}}(-1,1,1) - \vec{r}_j|, h_j) \; + \; \ldots \right]$$

While the memory requirements remain the same as those of the standard method, the computational costs increase about linearly with N_{smp}. For large variations of Λ on the lengthscale of the grid cells, the results will be better with this method.

The second modified SPH binning method, called broadened SPH, adds the width of one grid cell to the smoothing length and thus prevents particles from falling through the grid:

$$\Lambda(\vec{r}) = \sum_{j=1}^{N_{\mathrm{ngb}}} m_j \frac{\Lambda_j}{\rho_j} W(|\vec{r} - \vec{r}_j|,\, h_j + l_{\mathrm{cell}}) \; ,$$

with l_{cell} the length of a cell. Computional costs can be decreased due to more widely spaced grids, but at the cost of considerable smoothing.

4.5.3. TSC and other frequently used window functions

There are several traditional window functions used to assign the irregular simulation data to a regular grid. They determine which area around the center of a grid cell is sampled in order to assign a value for a certain quantity to that cell, and how exactly every particle sampled for this cell is weighted in the process. The most commonly used window functions are the nearest grid point (NGP), the cloud-in-cell (CIC) and the triangular-shaped-cloud (TSC) method. We will only briefly describe them here, since they probably should not be used anymore except for performance or comparison reasons. Figure 4.6 depicts their window functions $W(\vec{x}) = \Pi_i W(x_i)$ both in real as well as in Fourier space.

- *NGP*: The nearest grid point method simply samples all particles within the width of a cell with full weight to that cell. Its window function reads as follows:

$$W(x_i) = \begin{cases} 1, & |x_i| < 0.5 \; , \\ 0, & \text{else} \; . \end{cases}$$

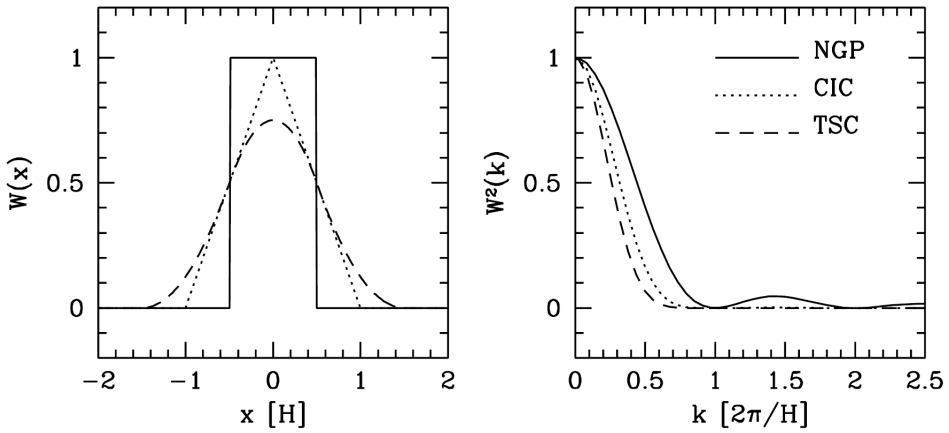

Figure 4.6.: The three common window functions NGP, CIC and TSC (left), and the square of their counterparts in Fourier space(right). Image credit: Cui et al. (2008)

- *CIC*: The cloud-in-cell method samples particles in a range of twice the cell length. The weighting decreases linearly with the distance of the particle to the cell's center. Its window function is:

$$W(x_i) = \begin{cases} 1 - |x_i|, & |x_i| < 1 \ , \\ 0, & \text{else} \ . \end{cases}$$

- *TSC*: The triangular-shaped cloud method is similar to CIC, only that the form of the window function is smooth and remindinds of a Gaussian. The sampling range extends over thrice the grid cell length. The slightly more complicated window function is:

$$W(x_i) = \begin{cases} 0.75 - x_i^2, & |x_i| < 0.5 \ , \\ \dfrac{(1.5 - |x_i|)^2}{2}, & 0.5 < |x_i| < 1.5 \ , \\ 0, & \text{else} \ . \end{cases}$$

An important feature of these three functions is their compact support in real space, which allows for computionally efficient binning, since only a finite (and not too large) number of particles is sampled per grid cell.

4.5.4. The D20 sampling

The above binning methods all share some inherent flaws with regard to the sampling quality: Since they provide no top-hat support in Fourier space, the sampling will

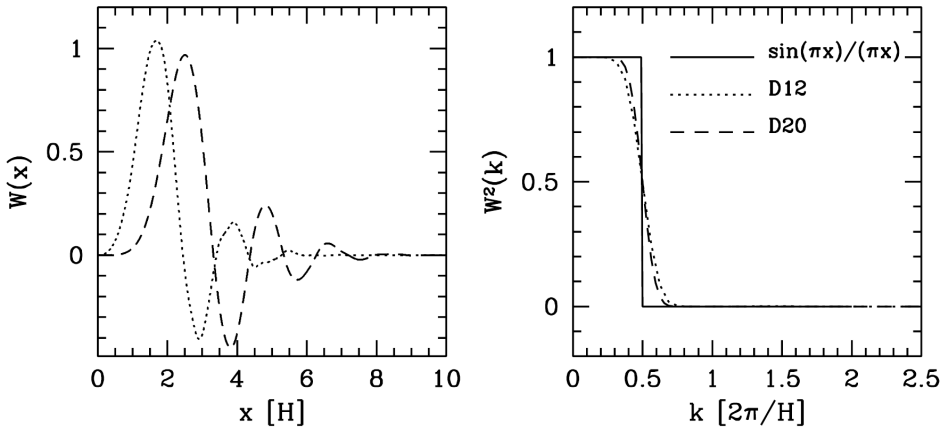

Figure 4.7.: The D12 and D20 window functions (left) and the square of their counterparts in Fourier space (right) as well as the top-hat window function in Fourier space corresponding to $W(x_i) = \sin(\pi x) \,/\, (\pi x)$ in real space are shown. Image credit: Cui et al. (2008)

be biased and the power spectrum will not be the true power spectrum but rather a version that is convoluted with the window function $W(x_i)$ in use.[15]

From a sampling point of view, the ideal window function would be $W(x_i) = \sin(\pi x) \,/\, (\pi x)$. It offers a perfect top-hat shape in Fourier space and would thus be aliasing-free. It has no compact support in real space, though, which makes it computationally unfeasible since a very large number of particles would be assigned to each and every grid cell.

To be practically free of the aforementioned sampling errors and especially the convolution issue but still have a good cost-benefit ratio, scaling functions of the Daubechies wavelet transformations as described by Daubechies (1992) can be used instead. The Daubechies 12 and 20 function (D12 and D20, respectively) both come very near to the optimum shape of a sampling function in Fourier space, providing a very good compromise between near-perfect top-hat-like support in Fourier space and compact support in real space. The former reduces sampling effects to a minimum, and the latter is necessary for computionally efficient mass assignment. Both D12 and D20 are depicted in figure 4.7, together with the Fourier transformed of $W(x_i) = \sin(\pi x) \,/\, (\pi x)$ for comparison. In this thesis, we will use the more accurate D20 function.

[15] For a detailed analysis of this alias effect and an elegant scheme to correct it afterwards, please cf. Jing (2005).

5. Our implementation of decaying turbulence

5.1. Motivation of this setup

In recent years several publications have given rise to the doubt about SPH's abilities to properly simulate turbulence. Most prominent in that regard is Bauer and Springel (2012), where the authors claim that smoothed particle hydrodynamics is not suitable for simulations of subsonic turbulence. Instead, the use of moving-mesh codes like Arepo is encouraged, as the diffusive nature of SPH due to its necessary artificial viscosity is blamed.[1]

One cannot help but notice that most recent comparison papers that seriously question SPH's abilities, like Bauer and Springel (2012), compare runs with the best settings of the alternative code to runs with basically the worst SPH settings. For example, simple viscosity limiters like the Balsara switch described in section 4.2.2 are not enabled, although their implementation into Gadget predates the newer codes by what must be a decade. This has motivated us to examine the performance of the not yet public but frequently used SPH code Gadget-3 in simulations of decaying turbulence in combination with up-to-date settings, and to compare it with matching Arepo runs. To that end we have performed Gadget-3 runs with three different degrees of turbulent energy $E_{\mathrm{turb}}/E_{\mathrm{therm}} = \{5\%, 10\%, 30\%\}$ at two different resolutions $N_{\mathrm{Part}} = \{128^3, 256^3\}$ each, and Christian Alig[2] was kind enough to provide us with the comparison runs done in Arepo at $N_{\mathrm{Part}} = 128^3$. Please cf. table 5.1 for a full listing of the different simulations.

5.2. Properties and realization of our simulation

In the following section we describe the details of our setup and how the simulations were carried out. We decided for a periodic box of *decaying turbulence*, which has the advantage of not needing any further power injection routine from "outside", as opposed to driven turbulence. To be able to witness the formation of full-fledged

[1] We have earlier discussed the need for an artificial viscosity in section 4.2.2.

[2] Dr. Christian Alig, Universitätssternwarte München, Munich, Germany.

	30%	10%	5%
128^3	*G_T30R128*	*G_T10R128*	*G_T05R128*
	vG_T30R128	*vG_T10R128*	*vG_T05R128*
	A_T30R128	*A_T10R128*	*A_T05R128*
256^3	*G_T30R256*	*G_T10R256*	*G_T05R256*

Table 5.1.: Overview of the GADGET-3 and AREPO simulations carried out in this thesis, sorted by turbulent energy fraction and resolution.

turbulence and the cascade of energy down through the lengthscales, we only seeded the largest 70 or so modes, similar to what Bauer and Springel (2012) aimed for with their driven turbulent box.

5.2.1. Setting up the turbulent box

The basic setup of our simulations consists of a 3000 kpc box filled with gas of the density $\rho \approx 1.5 \cdot 10^{-6} \frac{\mathrm{g}}{\mathrm{cm}^3}$ and at a temperature of 10^7 K, which in that regard mimics the intracluster medium. The temperature determines the thermal energy E_{therm}, and we have varied the turbulent energy, which is to say the energy invested into our initial turbulent velocity field, E_{turb} according to the ratio of the two. "turb30", for example, is short for $X_{\mathrm{turb}} = (E_{\mathrm{turb}}/E_{\mathrm{therm}}) = 30\%$.

We have created the initial conditions file `fibo_initial.ic` by starting IDL in the simulation's ICs directory on dorc and then compiling `make_data.pro` and executing `make_box`. The commands issued are:

```
1 IDL> .run make_data.pro
2 IDL> make_box
```

The settings of the box can be adjusted in `make_data.pro` before compilation. For the code of the procedure file please cf. appendix A.1.

5.2.2. Compiling GADGET

Because the simulation code GADGET-3 is extremely versatile, the options to compile it are numerous. Our simulation features turbulent gas of a density so low that self-gravitational effects are negligible, so gravitation may be turned off. After several tries, we decided for compilation settings that are part of the GADGET-*2014* scheme by Alexander Beck. This configuration proved to be the best choice, since they include a novel, as yet unpublished treatment of the artificial viscosity problem. One exception from the adherence to the scheme was the use of the *Wendland C6 kernel* (WC6) instead of the WC4.

To use GADGET for our simulation, the executable file `P-Gadget3` needs to be compiled from the source code in the way specified by the `Config.sh` file. This is done via the following command in Gadgets main directory:

```
1 /ptmp/.../P-Gadget3> Makefile
```

The routine `Makefile` is part of the GADGET code and uses `Config.sh` for the compile-time options. For our final settings please cf. appendix B.1.

5.2.3. Running the simulation

Crucial for a sucessful simulation are finely tuned execution parameters. Finding a sensible compromise between computational costs and accuracy is by itself not easy, but the real challenge lies in finding the parameters that lead to physically reasonable results. In our case, early settings featured much too high an artificial viscosity resulting in unphysically fast dissipation and gravitational softening of a kind to easily increase the computional time by a factor of 20 or more. In the end, again, the newest settings by Alexander Beck from GADGET-*2014* were the best choice, although some slight alterations due to the different kernel used at compilation were necessary.

Whereas the configuration file of the previous section contains information on the functions of GADGET-3 included in the executable, the `paramfile_fibo` hands the specifics of the simulation to the code. Those settings include rather straightforward items like the size of the box (3000 kpc), the internal units (u_{length}, u_{mass}, u_{velocity}) and the location of the initial conditions file, but also some rather delicate ones. Those are listed below:

- *Duration of the run*: The according parameters determine how long the simulation will run, and they are given in internal units. It is important to choose a time span long enough to cover all substantial developments in the box. This subject is closely related to the next item on this list, the time between snapshots. A good measure for such processes is the sound crossing timescale of the whole box, given by $t_{\text{sc}} = l_{\text{Box}}/c_s$. Our simulations run over course of approximately $1.5 \cdot t_{\text{sc}}$.
  Name: `TimeBegin` and `TimeEnd`.

- *Time between snapshots*: The parameter sets the time span between two consecutive output files and together with the duration of the run ultimately determines how many of these so-called *snapshots* will be written to disc. Please note that this output period is not equivalent to the time steps that GADGET actually does its calculations in. Those are much smaller and should be dynamically adjusted by the code itself. Also, the time between snapshots is a

parameter where the need for higher time resolution and the limits of acceptable memory consumption need to be reconciled. We have set the parameter such that 201 snapshots will be written.
Name: `TimeBetSnapshot` and `TimeOfFirstSnapshot`.

- *Number of neighbors*: The parameter designates the number of smoothing neighbors a particle should have. It is important since the individual smoothing length h_i of every particle is linked to it via equation (4.2). In our case, N_{ngb} is set to 295, and the maximum deviation allowed from this value is 0.01.
Name: `DesNumNgb` and `MaxNumNgbDeviation`

- *Softening lengths*: The numerous parameters determine the gravitational softening for various types of particles. We only use gas particles, but nonetheless they need to be well set. These are some of the settings that increased the runtime to unreasonable lengths, especially given that we use no gravitation. In the end, our simulations were performed with all parameters set to 1.
Name: `SofteningGas`, `~Halo`, `~Disk`, `~Bulge`, `~Stars`, `~Bndry` and maximum values for all of these: `SofteningGasMaxPhys` etc.

- *Artificial bulk viscosity constant*: Since we use time-dependent artificial viscosity as previously specified in `Config.sh` for compilation, we need to set the general artificial viscosity to a rather high value. In our case, this means a value of 3, whereas without the new scheme it would have been 1.
Name: `ArtBulkViscConst`

- *Viscosity settings*: The source scaling is deactivated, since we employ the novel scheme GADGET-*2014*. The viscosity decay length determines the reach of the persistence of the artificial viscosity, while $\alpha_{\mathrm{min}} = 0.025$ poses the lower limit away from shocks. The underlying scheme (with the source term activated) was originally proposed by Dolag et al. (2005) and has been briefly outlined in section 4.2.2. For our runs we have set the decay length to 4.
Name: `ViscositySourceScaling`, `~DecayLength` and `~AlphaMin`

- *Artificial conductivity*: These parameters are owed to our use of the artificial conductivity feature specified at compilation. The artificial conductivity constant is set to 1, and the lower limit to 0.
Name: `ArtCondConstant` and `ArtCondMin`

For the full parameter file please cf. appendix C.1.

It is possible to directly execute GADGET with these given parameters in the command line, but to utilize the massively parallel architecture of the code by using a larger number of processors, it is recommended to submit the simulation to the job management system of the dorc mashines via the following commands:

```
/ptmp/.../run3> qsub fibo_script_run3.sh
```

The script `fibo_script_run3.sh` contains the information on the technical aspects of the job, like the requested number of processors `ncpus=48` and allocated memory `mem=24` in GB. It can be found in appendix A.2.

The current status as well as the CPU time used so far by the simulation can be checked with `qstat`, and in combination with a so obtained job id `qdel` can be used to cancel individual runs.

5.2.4. Compiling Sph2Grid

Once the simulation has run through and all the snapshots have been written to disc, the data needs to be binned for further use.[3] For this task, we employ the routine Sph2Grid by Julius Donnert,[4] a program that by default uses a D20 kernel to sample the SPH data to a grid. Also particularly helpful is the option to compute the velocity power spectrum of every snapshot on the fly and store it together with the corresponding k values. That way, later we just need to extract the result for $P(k)$ from the grid files and plot it according to our wishes. Most other options are deactivated, since we will only analyze the velocity power spectrum.

The desired executable is called `Sph2Grid` or `Sph2Grid_D20` and is compiled out of the configuration file by going to the Sph2Grid main directory and typing the command:

```
/ptmp/.../Sph2Grid> ./Makefile
```

This automatically generates the specified version of it. `Makefile` is a routine original to Sph2Grid, but the corresponding `Config` file can be found in appendix B.2.

5.2.5. Binning the data

Of course, Sph2Grid needs some parameters, too, and they are stored in the file `sph2grid.par` in the routine's main directory. Most of the settings there are of a more technical nature, for example the location of the input and output file, the size of the box and, again, the internal Gadget units. Two of them are a bit more significant, though: The number of grid points per dimension (`GridPoints`) and the total number of bins (`Nbins`). Both are set to the same value as the spatial resolution of our SPH simulation per dimension, which is $(N_{\mathrm{Part}})^{1/3} = 128$ and 256, respectively. We will not further elaborate on that matter, but note that this is a question of sampling performance and all by itself not a trivial choice.

[3] Please cf. our previous discussion in section 4.5 for more details.
[4] Dr. Julius Donnert, Istituto di Radioastronomia, Bologna, Italy.

By itself, SPH2GRID only processes a single snapshot file per call. The command to do so is:

```
/ptmp/.../Sph2Grid> ./Sph2Grid sph2grid.par
```

For a few hundred of them, executing that by hand would be tedious and time-consuming, to say the least. Thus we have devised a handy script that allows for a more convenient workflow: The number of the first and of the last snapshot file to be binned have to be specified in the script itself, and a simple executive call of the script suffices to start the automated routine. Therefore, it recognizes the `snap_***` and the `grid_***` filenames in the `sph2grid.par` parameter file, replaces the number parts with the lowest specified number, calls SPH2GRID as specified above, increments the number by one and loops over the whole process until the highest specified snapshot has been binned. To call the script type:

```
/ptmp/.../Sph2Grid> ./script_Sph2Grid.sh
```

The script `script_Sph2Grid.sh` can be found in appendix A.3, the parameter file in appendix C.2.

5.2.6. Plotting the spectrum

Last but not least we arrive at the final step of our workflow, the extraction and plotting of the already calculated velocity power spectrum $P(k)$. To do that, we use an extensive IDL routine by the name of `powerspectrum2.pro`. It is based on the routine `powerspectrum.pro` by Julius Donnert, and has been greatly modified and extended to allow for several different output options. Same as the original version, the extended one needs to be compiled twice before execution:

```
IDL> .run powerspectrum2.pro
IDL> .run powerspectrum2.pro
```

The heart of the plotting routine is the `powerspectrum_subroutine` procedure. It does the actual reading, calculating (as far as necessary) and plotting. Both procedures `powerspectrum_serial` and `powerspectrum_multi` are just frontends designed for different purposes. They primarily determine format and number of the plots produced.

- *Serial functionality*: Numerous single JPEG files of rather low resolution are produced by `powerspectrum_serial`. We have created this feature especially for quick and dirty outputs while the simulations are still running. With the already available snapshot files binned and then processed this way, necessary assessments with regard to the usefulness of the run can be made. Also, a whole batch of the low-resolution images can easily be converted into a GIF file, visualizing the evolution of the spectrum over time.

Keywords: `fname` and `fend` need to be set to the name of the first and last grid file to be processed. Typical call:

```
IDL> powerspectrum_serial, fname='./grid_000', fend='./grid_200'
```

- *Multi-page functionality*: A number of different power spectrum plots is done on the same page. The output is as an EPS file and will have print quality. This front-end allows for highly customizable plots, and all the multi-plot figures and many of the single plots created for this thesis have been done with `powerspectrum_multi`.
  Keywords: `twofiles` or `threefiles` tells the routine the number of data sets (grid files) per sub-plot; if the keyword is undefined, only a single file is used. `ncols` and `nrows` specify the number of sub-plots in the horizontal and vertical expanse, respectively; if undefined, they default to 1. `nodetails` disables in-plot annotations and the like; this is useful for plots containing a large ammount of rather small sub-plots. `fname`, `f2name` and `f3name` specify the names of the first one, two or three files to be plotted; if they are not set, the routine will prompt the user to enter the right amount of file names. `block` allows it to read information other than the velocity power spectrum, to which it defaults; should be left unset, since we only computed that with Sph2Grid. Typical call:

  ```
  IDL> powerspectrum_multi, threefiles=1,ncols=2,nrows=3,nodetails=1
  ```

  The user will the be asked to enter the path to the first file and, depending on the ∼`files` settings, to the second and third one for every iteration and thus sub-plot. To end the input and properly close the file, please type `end` instead of the first filename of a new iteration. Also, the scheme has an automatic exit if two subsequent filenames

Fine-tuned adaptions to the plots like changes of color, linestyle or annotations need to be done in the code of `powerspectrum_subroutine` itself. The full source code of `powerspectrum2.pro` can be found in appendix A.4.

6. Results

In this section we present our findings and analyze the resulting data from the numerous simulations mainly with the help of power spectra. The results are sorted by degree of turbulence, starting with the highest. An overview of the naming scheme was given in table 5.1 on page 44.

6.1. Simulations with 30% turbulence

Of the different simulation setups we have run, those evolving from initial conditions with a ratio $X_{\mathrm{turb}} = E_{\mathrm{turb}}/E_{\mathrm{therm}} = 30\%$ of turbulent to thermal energy in the velocity field are the ones least prone to artificial viscosity problems.

6.1.1. GADGET-vs.-AREPO comparison

Low resolution runs

Both 128^3 particle simulations *G_T30R128* (GADGET) and *A_T30R128* (AREPO) start out so similar that in the first two snapshots presented in figure 6.2 they are practically indistinguishable. This does not come as a surprise, because the early development of the velocity power spectra is substantially influenced by the initial conditions, and both runs use exactly the same ICs file. They continue to behave very similarly up until about $0.5\,t_{\mathrm{sc}}$, with $t_{\mathrm{sc}} \approx 3.268 \cdot 10^{17}\,\mathrm{s}$ the sound crossing time of our 3000 kpc box.

From snapshots 075 through 150 in figure 6.3, *A_T30R128* stays at lower powers and sports a steeper, less Kolmogorov-like slope than *G_T30R128*. The difference becomes relevant at $k \approx 0.04\,\mathrm{kpc}^{-1}$, which is well below $k_{\langle \mathrm{sml} \rangle} \approx 0.065\,\mathrm{kpc}^{-1}$, the wave number of the average smoothing length $\langle h_{\mathrm{sml}} \rangle \approx 96.78\,\mathrm{kpc}$ of the lower resolution GADGET run. This means that the AREPO run contains less energy on smaller lengthscales at that point in the simulation.

Sometime around $1.2\,t_{\mathrm{sc}}$ (between snapshots 150 and 175), *A_T30R128* catches up to his GADGET counterpart and again produces near-identical results, up to their veering from the theoretically predicted Kolmogorov $-\frac{11}{3}$ slope. Only at the average smoothing length $k_{\langle \mathrm{sml} \rangle}$ do they separate, AREPO for now showing much more shot noise at the highest end of the spectrum. It is remarkable that while the tails of

line	name	abbrev.	definition
— —	box mode	k_{box}	$2\pi/L$
– · · · –	minimal seeding	$k_{\mathrm{seed,min}}$	$\pi/2 \cdot k_{\mathrm{box}}$
– · · · –	maximal seeding	$k_{\mathrm{seed,max}}$	$\pi \cdot k_{\mathrm{box}}$
– – –	average smoothing length (2×)	$k_{\langle \mathrm{sml} \rangle}$	$2\pi\, N_{\mathrm{Part}} / (\sum_j h_{\mathrm{sml},j})$
— —	Nyquist mode	k_{Nyquist}	$2\, N_{\mathrm{Grid}}/L$

Table 6.1.: Overview of the important k-values shown as vertical lines in all our major power spectrum plots. The Nyquist mode is shown for $N_{\mathrm{Grid}} = 256$ only. Listed according to their left-to-right appearance.

both GADGET runs (low and high resolution) vary less than an order of magnitude during the whole run of the simulation, the lower end ot AREPO's power spectrum continually increases, having gained three orders of magnitude by the time of the final snapshot at $1.5\, t_{\mathrm{sc}}$.

High resolution run

The high resolution GADGET run *G_T30R256* is, over the course of the $1.5\, t_{\mathrm{sc}}$ long run of the simulation, in general agreement with both lower resolution setups. It starts with less energy for values of k beyond the seeding range, since its higher spatial resolution N_{Grid} lowers that discretization noise significantly as the half-sized particles excert only smaller rearrangement motions. After less than $0.4\, t_{\mathrm{sc}}$ have passed, *G_T30R256* and the two lower resolution runs feature the same major slope of still developing Kolmogorov turublence.

The former holds true except near the resolution limit of both *G_T30R128* and *A_T30R128*, which can roughly be gauged by the average smoothing length $k^{128}_{\langle \mathrm{sml} \rangle}$ of the lower resolution GADGET run. In this region of the power spectrum, *G_T30R256* follows the straight line of a power law longer, even before a $-\frac{11}{3}$ like slope is established, then shows the same dip in energy aroud $k^{256}_{\langle \mathrm{sml} \rangle}$ as its lower resolution pendant. It is arguable that turbulence on scales below the average smoothing length is not sufficiently resolved to carry any physical meaning, though, so the so-called shot noise, characteristic for the final steep increase of the power spectrum at high values of k, could as well be cut off or carefully isolated and corrected.

More information on the minimum wave number $k_{\mathrm{seed,min}}$ and the maximum wave number $k_{\mathrm{seed,max}}$ of the seeding range[1] as well as the Nyquist mode k_{Nyquist} and other important wave numbers can be found in table 6.1.

[1] We have used a similar seeding range for our *decaying* turbulence as Bauer and Springel (2012) have for their *driven* turbulence.

Comparison plots of various snapshots

The plots in figure 6.1 show the first and the last snapshot of the previously discussed 30% runs, while the figures 6.2 and 6.3 also present the evolution of the box in between.

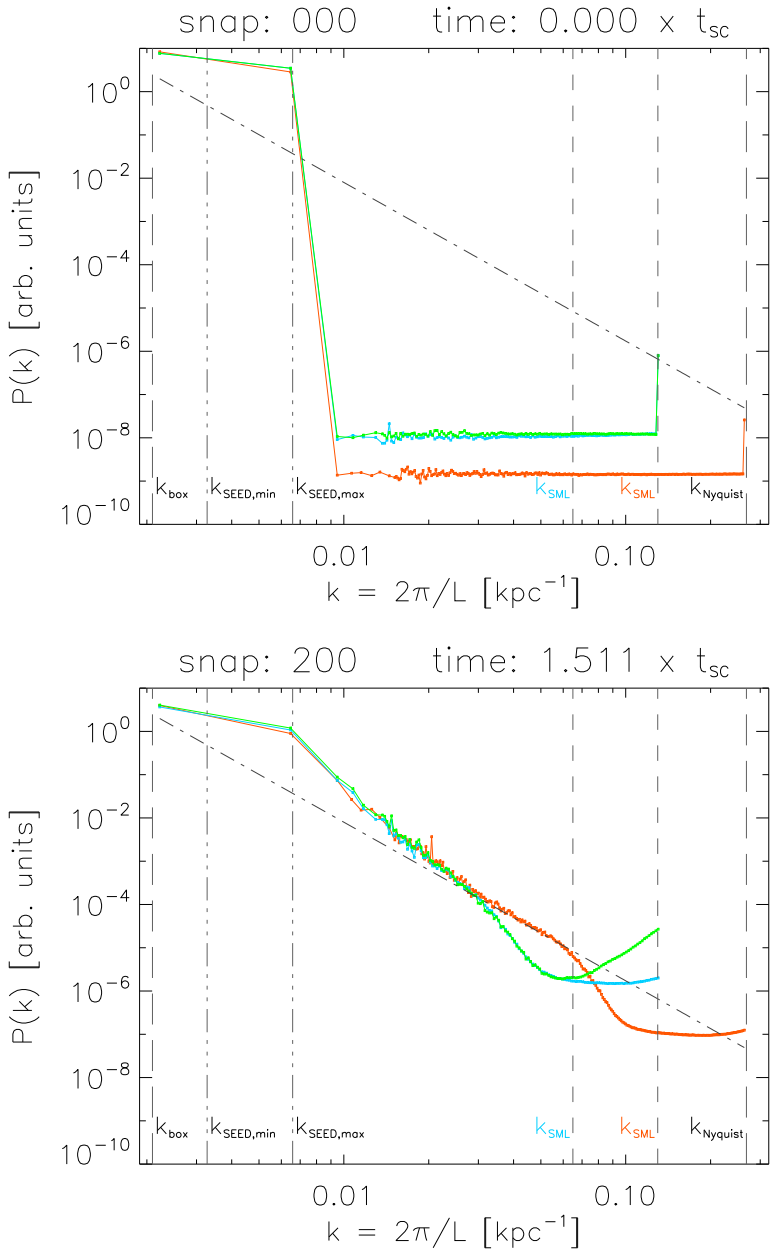

Figure 6.1.: Velocity power spectra of the first and the last snapshot for the GADGET-3 runs and the AREPO run with 30% turbulence each. The graphs shown are: *G_T30R128* (blue), *G_T30R256* (red) and *A_T30R128* (green) as well as the Kolmogorov $k^{-11/3}$ spectrum (dash-dotted line).

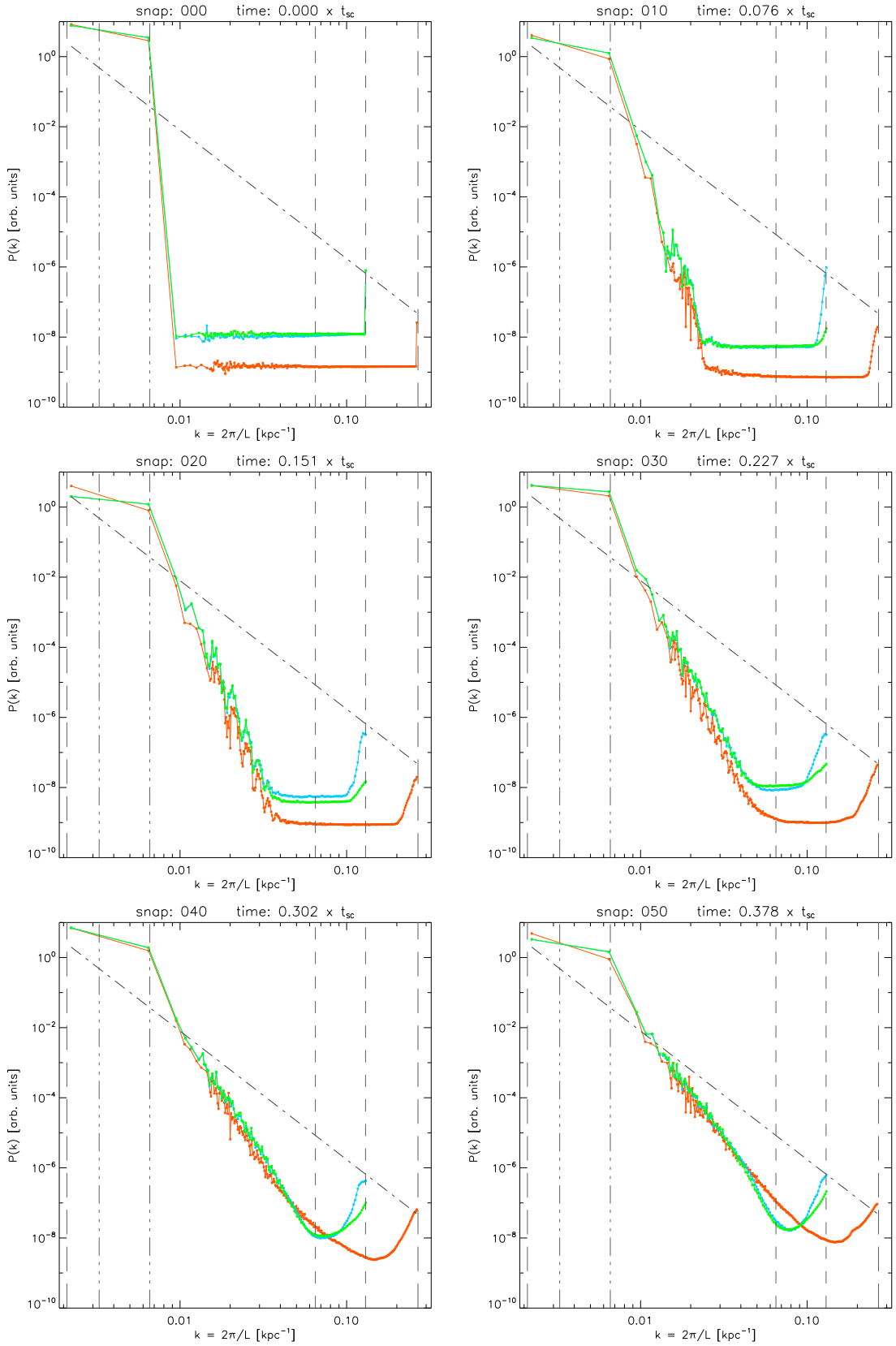

Figure 6.2.: Velocity power spectra of snapshots 000 through 050 for the GADGET-3 runs and the AREPO run with 30% turbulence each. The graphs shown are: *G_T30R128* (blue), *G_T30R256* (red) and *A_T30R128* (green) as well as the Kolmogorov $k^{-11/3}$ spectrum (dash-dotted line).

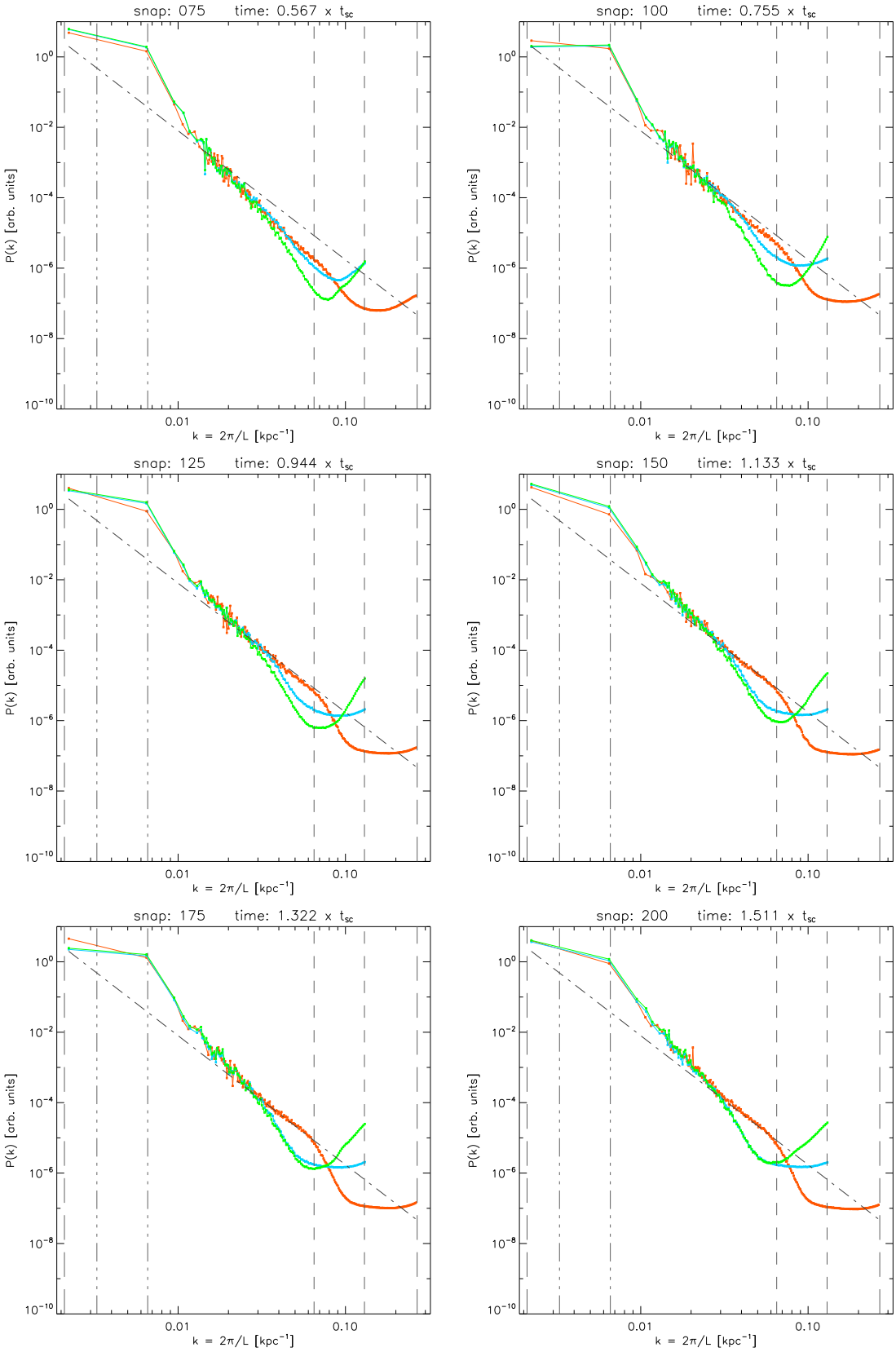

Figure 6.3.: Velocity power spectra of snapshots 075 through 200 for the GADGET-3 runs and the AREPO run with 30% turbulence each. The graphs shown are: *G_T30R128* (blue), *G_T30R256* (red) and *A_T30R128* (green) as well as the Kolmogorov $k^{-11/3}$ spectrum (dash-dotted line).

6.2. Simulations with 10% turbulence

This section presents our simulations based on $X_{\mathrm{turb}} = E_{\mathrm{turb}}/E_{\mathrm{therm}} = 10\%$ in the initial conditions. At this amount of turbulent energy, we may expect to see the first signs of possible problems with the artificial viscosity in our SPH runs, since with the serious decrease in kinetic energy the balance between inertial and viscous forces has shifted in favour of the latter.

6.2.1. GADGET-vs.-AREPO comparison

Low resolution runs

As opposed to the 30% turbulence runs of the previous section, *G_T10R128* and *A_T10R128* do not start out exaclty the same way. First of all, the discretization noise beyond the seeding range is about half an order of magnitude higher for the GADGET run, as can be seen in the first plot of figure 6.4 as well as figure 6.5. The latter shows further differences between both our low resolution runs: Where in snapshots 20 through 40 the AREPO run *A_T10R128* behaves similarly to its higher turublence version *A_T30R128* and slowly populates the smaller modes,[2] the low resolution GADGET run *G_T10R128* prominently increases the energy on smaller scales within $0.1\,t_{\mathrm{sc}}$ and assumes an almost Kolmogorov-like slope.

Somewhere between snapshots 050 and 075 (or at about $0.5\,t_{\mathrm{sc}}$), the two low resolution runs reach general concurrence again, and carry on to do so for the better part of the run. Only after approximately $1.3\,t_{\mathrm{sc}}$ do they diverge again, mainly because *A_T10R128* takes on a steeper than $-\frac{11}{3}$ slope and develops the characteristic minimum in $P(k)$ around the average smoothing length of its GADGET counterpart, while *G_T10R128* retains the Kolmogorov spectrum.

High resolution run

In the beginning, the high resolution a run *G_T10R256* behaves similar to *A_T10R128* with regard to the gradual population of the smaller modes (see figures 6.5, snapshots 000 through 040). Between snapshots 040 and 750, *G_T10R256* exceeds both lower resolution runs in its power and assumes a Kolmogorov slope at snapshot 100 (about $0.7\,t_{\mathrm{sc}}$. From thereon, *G_T10R256* behaves very similar to its lower resolution pendant *G_T10R128*, although the latter sports an offset power of about half an order of magnitude down compared to it.

[2] By "smaller modes" we mean modes of smaller dimension L in real space. This commutes with larger values $k = 2\,\pi/L$ in Fourier space and in the power spectrum plots.

Comparison plots of various snapshots

The plots in figure 6.4 show the first and the last snapshot of the previously discussed 10% runs, while the figures 6.5 and 6.6 also present the evolution of the box in between.

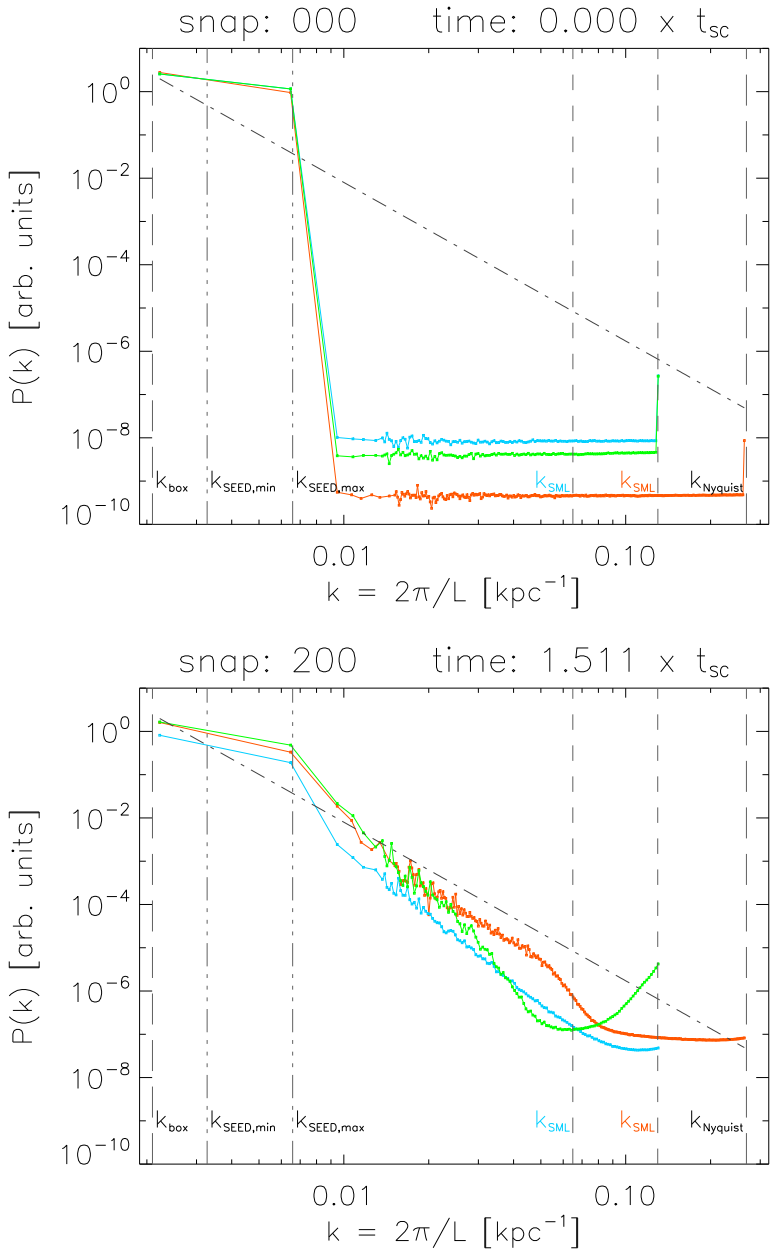

Figure 6.4.: Velocity power spectra of the first and the last snapshot for the GADGET-3 runs and the AREPO run with 10% turbulence each. The graphs shown are: *G_T10R128* (blue), *G_T10R256* (red) and *A_T10R128* (green) as well as the Kolmogorov $k^{-11/3}$ spectrum (dash-dotted line).

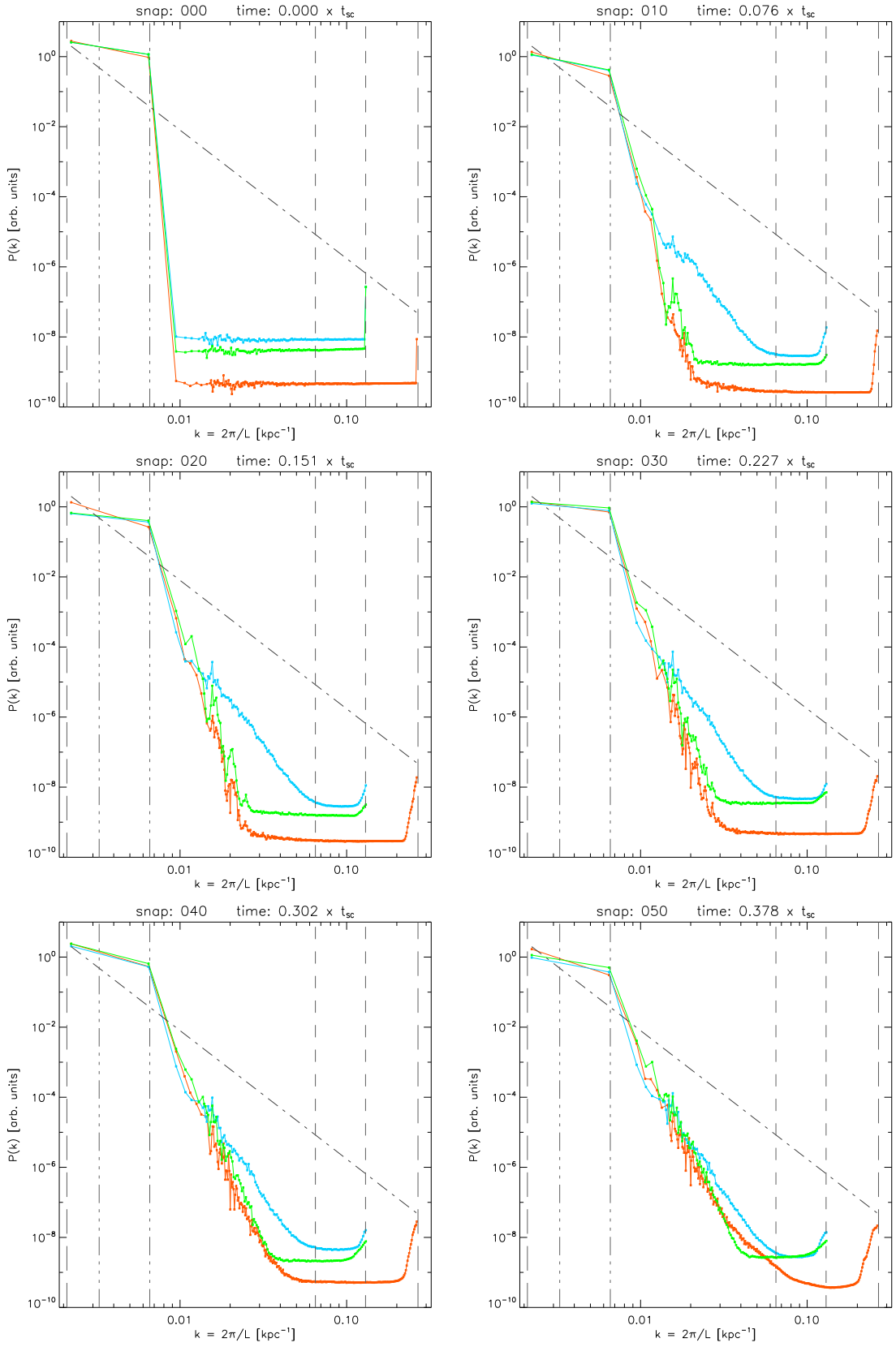

Figure 6.5.: Velocity power spectra of snapshots 000 through 050 for the GADGET-3 runs and the AREPO run with 10% turbulence each. The graphs shown are: *G_T10R128* (blue), *G_T10R256* (red) and *A_T10R128* (green) as well as the Kolmogorov $k^{-11/3}$ spectrum (dash-dotted line).

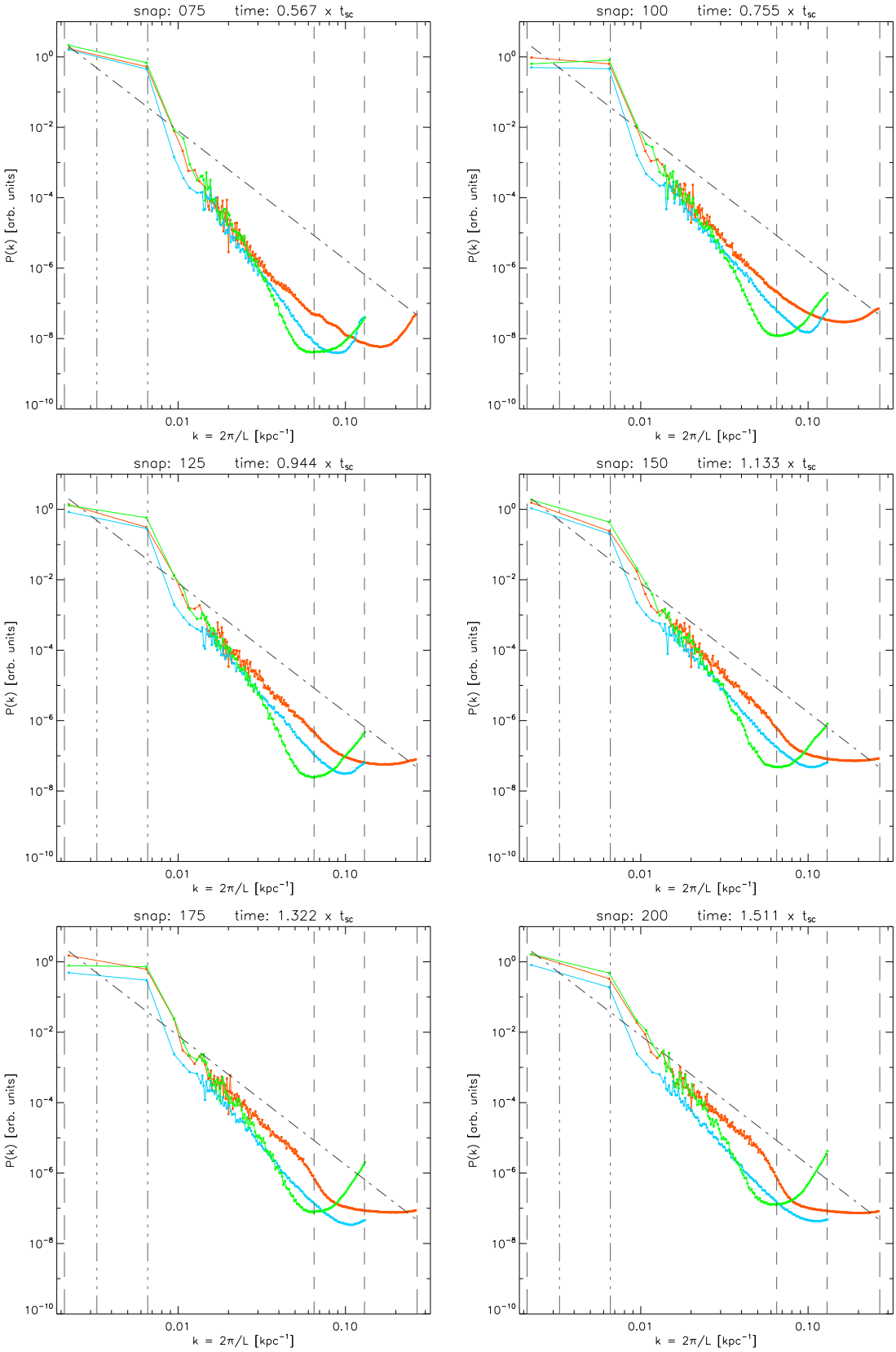

Figure 6.6.: Velocity power spectra of snapshots 075 through 200 for the GADGET-3 runs and the AREPO run with 10% turbulence each. The graphs shown are: *G_T10R128* (blue), *G_T10R256* (red) and *A_T10R128* (green) as well as the Kolmogorov $k^{-11/3}$ spectrum (dash-dotted line).

6.2.2. Long run GADGET-vs.-AREPO comparison

To further explore the long-time development of both GADGET-3 and AREPO at medium to low amounts of turbulent energy $X_{\mathrm{turb}} = 10\%$, we have extended the lower resolution runs *G_T10R128* and *A_T10R128* to the total runtime of $4.5\,t_{\mathrm{sc}}$. The results are shown in figure 6.7.

The most distinctive feature of the long run comparision plots is the overall disagreement between *G_T10R128* and *A_T10R128* over the magnitude of $P(k)$. The already significant differences in the seedings range between $k_{\mathrm{seed,min}}$ and $k_{\mathrm{seed,max}}$ compound themselves when transferred to smaller, individually less energetic modes, such that in the about end two orders of magnitude in power separate the moving-mesh from the SPH code over large parts of the spectrum.

Our plots show GADGET to dissipate visibly more energy over the whole length of the run than AREPO. This becomes evident in the seeding range, where the power of *G_T10R128* has sunk to levels more than half an order of magnitude lower than *A_T10R128*. As time progresses, the power spectrum of our SPH run "sags" ever more, displaying a significant loss of energy, probably to viscosity, over large parts of what should actually be the inertial subrange and thus by definition near viscosity-free.

Intersting enough, AREPO all the while continues its general upward trend in the power spectrum plots and only later achieves a Kolmogorov-like slope in the very center of the spectrum. Its characteristic dip in $P(k)$, which was located around the average SPH smoothing length at the end of our original comparision at snapshot 200 (about $1.5\,t_{\mathrm{sc}}$), moves up to larger modes and reduces the effective resolution of the moving-mesh code, though.

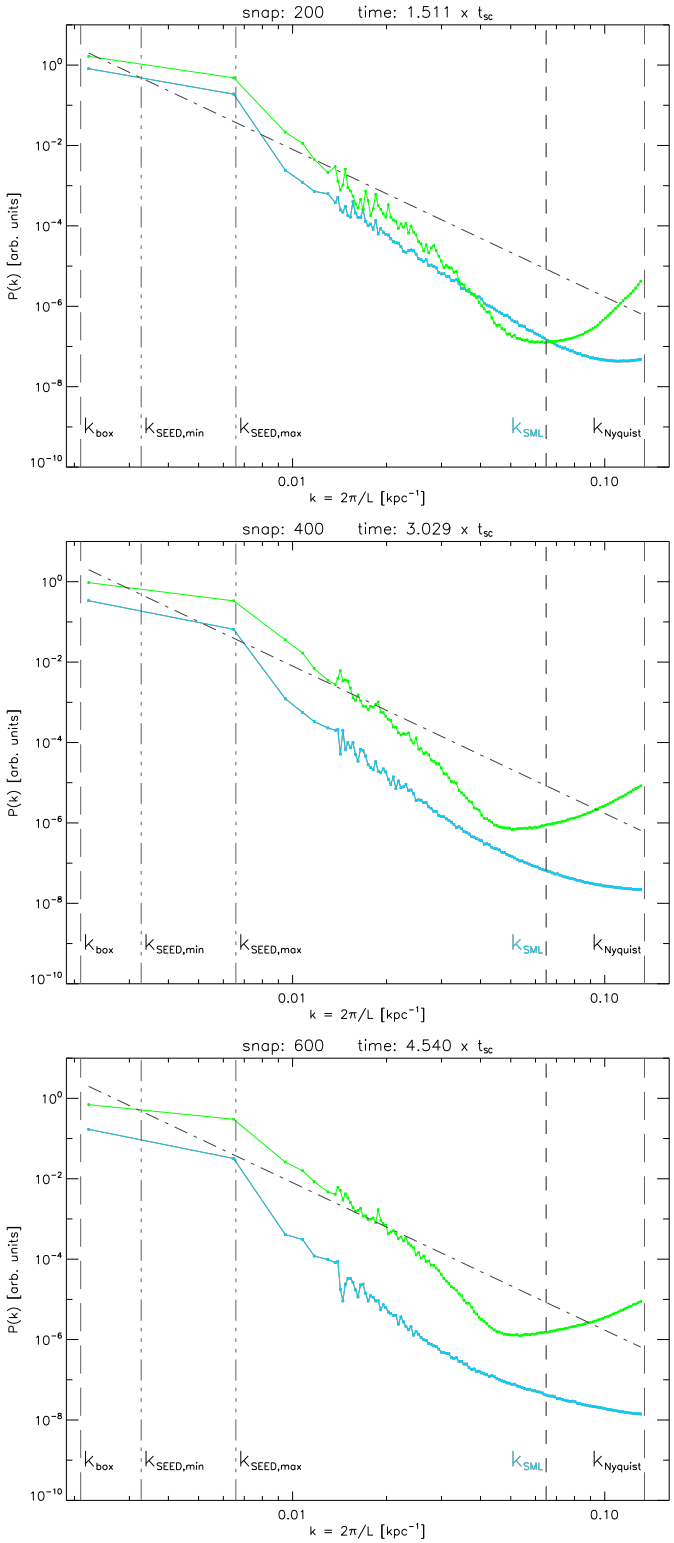

Figure 6.7.: Velocity power spectra of snapshots 200, 400 and 600 for the lower resolution GADGET-3 and AREPO runs with 10% turbulence each. The graphs shown are: *G_T10R128* (blue) and *A_T10R128* (green) as well as the Kolmogorov $k^{-11/3}$ spectrum (dash-dotted line).

6.3. Simulations with 5% turbulence

In this last section of the current chapter, we present the results for our simulations with the least turbulent energy, determined via $X_{\mathrm{turb}} = E_{\mathrm{turb}}/E_{\mathrm{therm}} = 05\%$ in the initial conditions. At this ratio between intertial and viscous forces, one may expect our SPH code GADGET to be at a disadvantage compared to the moving-mesh code AREPO. We will observe if that is the case.

6.3.1. GADGET-vs.-AREPO comparison

Low resolution runs

Quite similar to the low resolution runs in the previous section on 10% turbulence, the GADGET run *G_T05R128* takes off pretty fast and assumes its Kolmogorov-like slope before $0.1\,t_{\mathrm{sc}}$ have passed. Meanwhile, *A_T05R128* ever so slowly transfers power to the smaller modes, as is to be expected from developing turbulence. In contrast to the 10% runs, there is not initial offset in the discretization noise between both codes.

At snapshot 075 (or about $0.5\,t_{\mathrm{sc}}$), both runs converge at least in middle parts of the spectrum, but AREPO does not develop the $-\frac{11}{3}$ slope typical for Kolmogorov turublence at all, and does not so over the run of the whole simulation of about $1.5\,t_{\mathrm{sc}}$. Instead, the previously observed dip in power around the average smoothing length $k_{\langle \mathrm{sml} \rangle} \approx 0.065\,\mathrm{kpc}^{-1}$ of the GADGET run is even more pronounced than before. *G_T05R128* on the other hand maintains a very steady and quite Kolmogorov-like spectrum until the end.

High resolution run

The high resolution GADGET run *G_T05R256* starts out similar to its higher energy counterparts with much less energy outside the seeding range than the lower resolution runs *G_T05R128* and *A_T05R128*. It then slowly but steadily fills up the smaller modes of the spectrum, which is well-pronounced by the ever shrinking flat region in figure 6.9 basically constituted of discretization noise. At the time of snapshot 075 (about $0.5\,t_{\mathrm{sc}}$) the flat region has been completely replaced by a sloped spectrum that evolves into the expected Kolmogorov spectrum before a simulation time of $0.9\,t_{\mathrm{sc}}$ is reached. *G_T05R256* keeps this form until the end of our simulation and is in good accord with its lower resolution pendant *G_T05R128*.

Comparison plots of various snapshots

The plots in figure 6.8 show the first and the last snapshot of the previously discussed 10% runs, while the figures 6.9 and 6.10 also present the evolution of the box in between.

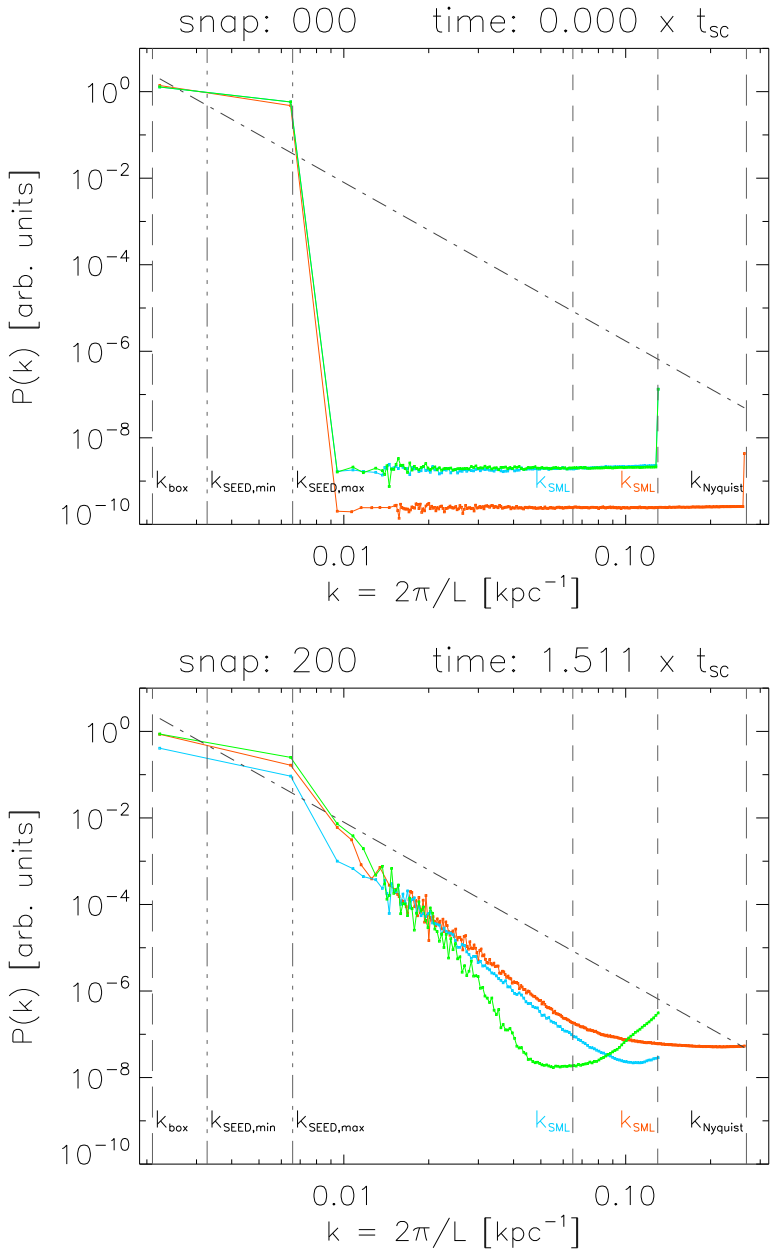

Figure 6.8.: Velocity power spectra of the first and the last snapshot for the GADGET-3 runs and the AREPO run with 10% turbulence each. The graphs shown are: *G_T05R128* (blue), *G_T05R256* (red) and *A_T05R128* (green) as well as the Kolmogorov $k^{-11/3}$ spectrum (dash-dotted line).

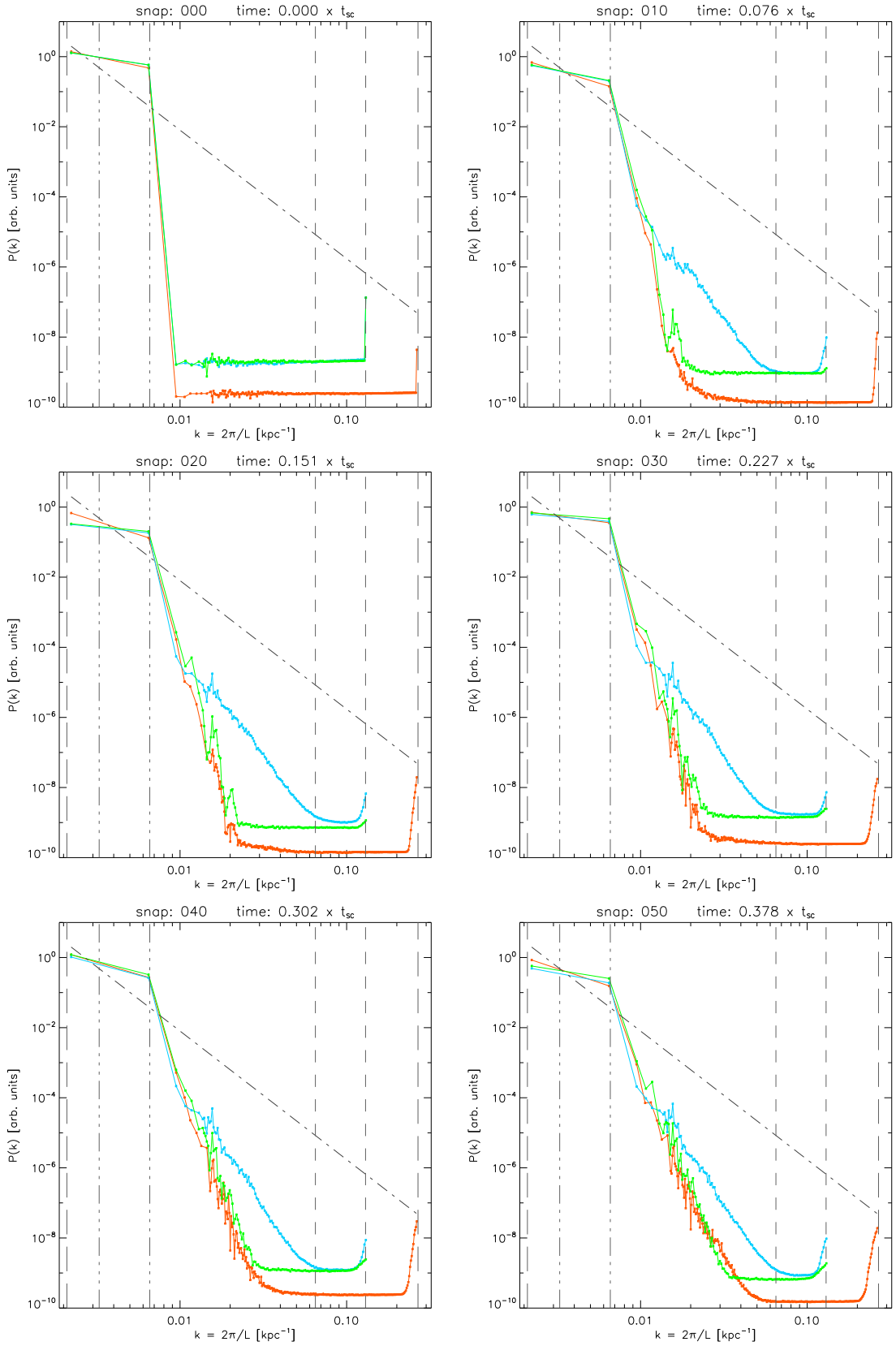

Figure 6.9.: Velocity power spectra of snapshots 000 through 050 for the GADGET-3 runs and the AREPO run with 5% turbulence each. The graphs shown are: *G_T05R128* (blue), *G_T05R256* (red) and *A_T05R128* (green) as well as the Kolmogorov $k^{-11/3}$ spectrum (dash-dotted line).

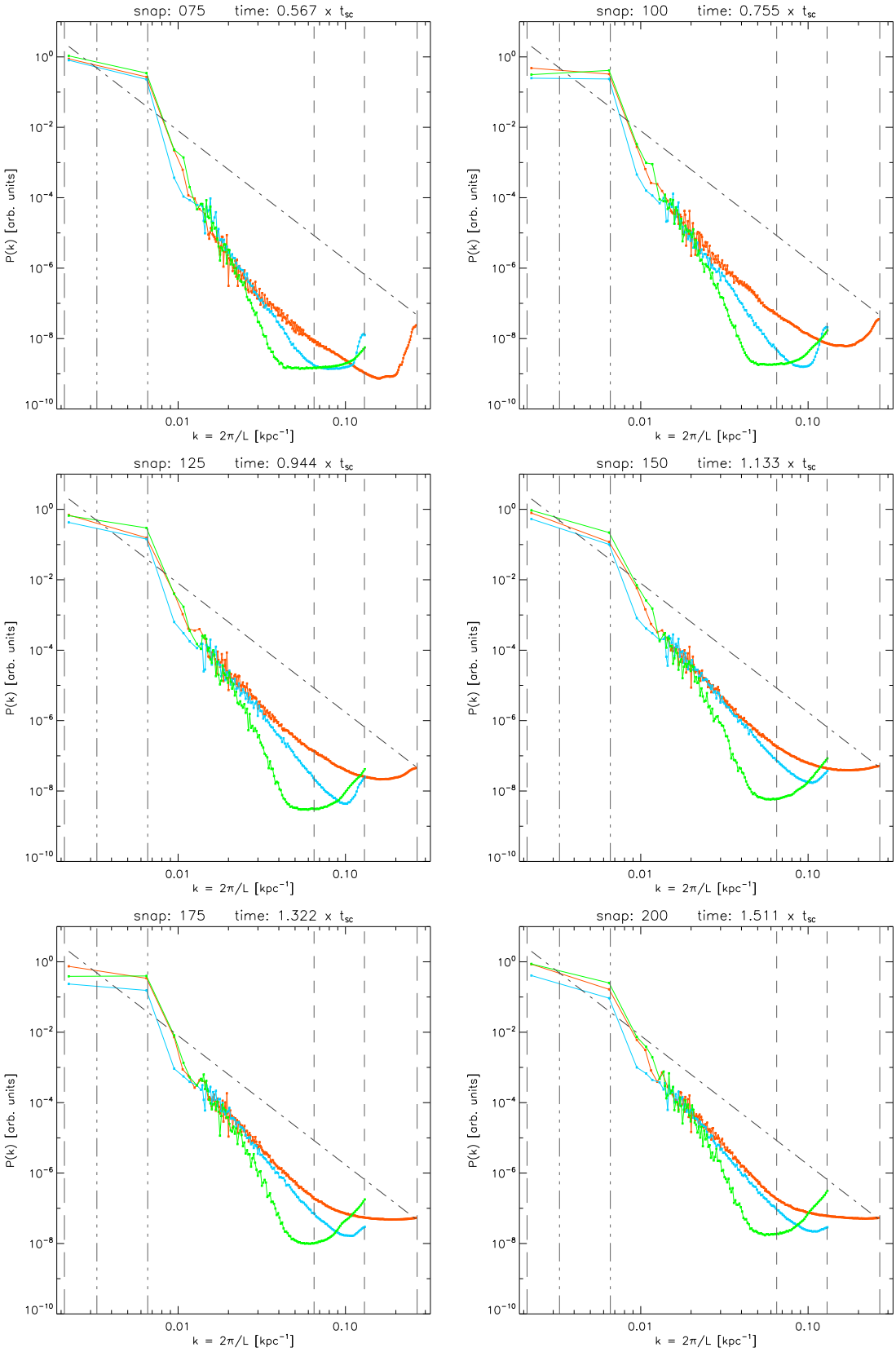

Figure 6.10.: Velocity power spectra of snapshots 075 through 200 for the GADGET-3 runs and the AREPO run with 5% turbulence each. The graphs shown are: *G_T05R128* (blue), *G_T05R256* (red) and *A_T05R128* (green) as well as the Kolmogorov $k^{-11/3}$ spectrum (dash-dotted line).

7. Discussion

In this final chapter we will condense our previous findings. The following discussion will compare the results with our expectations and point out likely reasons for new developments. Then we will outline some points of interest that go beyond this thesis and risk a glimpse into the possible future.

7.1. Summary

We have set out to examine the current version of GADGET-3 employing the not yet public GADGET-*2014* settings and schemes, and to compare it to the moving-mesh code AREPO. To that end we first have established the basic theory of hydrodynamics and turbulence in chapters 2 and 3. They lead to our understanding of fully developed turbulence, the main ideas of which are dissipation on small scales, a $-\frac{11}{3}$ Kolmogorov power spectrum for the inertial range in three dimensions and, in general, the turbulent cascade with its practically dissipation-free energy transport down along the whole range of the power spectrum.

The more applicatory chapters 4 and 5 have seen the introduction of essential simulation techniques and their adaption to our needs. Besides the basic concept of our employed simulation and analysis techniques as well as the description of our setups, potential strengths and weaknesses of smoothed particle hydrodynamics were discussed.

Finally, in the previous chapter, we moved on to our simulation results. We have found both GADGET-3 and AREPO to perform very similar in the 30% turbulence scenario, but to show significant differences in runs with decidedly less overall turbulent energy, such as $X_{\mathrm{turb}} = 10\%$ or even 5%. While low resolution SPH tends to have a generally lower power spectrum, it still manages a more or less Kolmogorov-like slope over the central part of the spectrum, thus agreeing with our theoretical expectations of turublence. AREPO has severe problems forming a sufficiently large intertial range and Kolmogorov turublence in the very low turublence setting, but suffers not from the probably constant energy drain GADGET experiences in the extended 10% run. The higher resolution SPH runs deliveres the most physically and consistently sound results.

7.2. Conclusions

The results of our simulations in chapter 6 have lead us to several conclusions about the ability of SPH codes in general and of GADGET-3 with state-of-the-art viscosity schemes in specific to simulate turbulence in the subsonic regime.

Expected resolution limits

The resolution limit of GADGET and AREPO is, at the very best, expected to be the average smoothing length $\langle h_{\mathrm{sml}} \rangle$ of an SPH run with the same number of particles, since at that scale dissipation occurs. This makes sense from a physical point of view, as the discretized volume elements of the fluid are treated as particles, and in motions well below their average outer dimensions, friction, attraction and internal effects would dominate. Also, from the more practical perspective of simulations, motions below the typical length of particles will en masse just occur within those particles and be neither for SPH nor for moving-mesh codes resolved by equations of motion that handle those point particles as smallest (and only) entities.

The role of artificial viscosity

At the same resolution and relatively high ammounts of turbulent energy (30%), GADGET and AREPO perform almost identically, with a slight advantage for GADGET, as the SPH scheme stays closer to the "ideal" theoretical Kolmogorov spectrum of developed turbulence. Quite clearly, a GADGET run of significantly increased resolution outperforms both of them, which was expected.

For noticeably less turbulent energy (10%), a lower resolution run of GADGET transfers power to the smaller modes much faster than the AREPO run of the same or the GADGET run of a higher resolution does. The cause for that is not clear yet and should be subject to further examination. Quite fascinatingly, in the later course of the simulation not only the higher resolution but also the lower resolution GADGET run fare much better in forming the characteristic $-\frac{11}{3}$ Kolmogorov slope of three-dimensional fully developed turbulence. The moving-mesh code AREPO loses a lot of power on the smaller scales well above the average smoothing length of the corresponding SPH run. This is even more distinct for the least turbulent simulations (5%).

The generally good performance of GADGET is due to its new artificial viscosity scheme by Alexander Beck. Since SPH is in its original version perfectly Lagrangian and thus non-dissipative, the only reason why it could loose energy on smaller scales would be because of the later on introduced artificial viscosity. Contrary to that, our long run comparison seems to point towards some low but permanent artificial viscosity present in our version of GADGET that over the course of several sound

crossing times steadily robs the spectrum of its power and effectively impedes turbulent motions on different scales. As the (shorter) higher resolution GADGET runs show a different behavior, this might in part be a resolution issue and is worth investigating.

In case of the very low turbulence setting (5%), the new scheme employed in our state-of-the-art simulations reduces the artificial viscosity away from shocks to such a low amount, that dissipation occurs well below the effective resolution. Thus the maximum range of turbulence for the particular resolution is achieved, while AREPO is prevented from forming turbulence by its natural mixing.

We find that especially GADGET-*2014* is reliable when it comes to fully developed turbulence, and that for comparatively low resolutions like $N_{\mathrm{Part}} = 128^3$ caution during the formation period of the Kolmogorov spectrum is in order.

Additional remarks

The omnipresent increase in power at the low end tail of the power spectrum both in our SPH and Gadget simulations is due to the so-called *shot-noise* effect. It stems from the discretization of an originally continuous fluid problem and is one of several possible sampling effects to occur. Since by using the D20 binning method we eliminated most of the other effects, the shot-noise is the most prominent distortion to our results.[1] Its significance is limited to the usually not well resolved lower end of the spectrum beyond the average smoothing length, though, so it is often left in place or just cut off.

[1] The D20-proposal paper by Cui et al. (2008) also briefly addresses other sampling effect besides the convolution with different window functions.

7.3. Future prospects

It will be interesting to see how the new artificial viscosity and conductivity schemes comprised in GADGET-*2014* handle fluid instabilities. Several standad tests like one-dimensional shock tubes, Kelvin-Helmholtz instabilities and the Keplerian ring test remain to be demonstrated, but if the performance so far is any indication it will clear those hurdles with grace.

Future comparisons with moving-mesh codes may have to take greater care with the binning mechanisms used on the AREPO snapshots. Due to the high compatibility between GADGET-3 and AREPO output, we have implicitly assumed that we may use our binning scheme on the moving-mesh code's results, as well. It is quite possible, though, that the cell structure needs to be taken into better account.

Also, future investigations could try to isolate the power actually contained in the turbulent cascade from the energy of the seeding range and thus further examine the cascading process, especially with regard to possible influences of dissipative terms in areas outside the low end dissipation subrange of the spectrum. This would probably require some advanced deconvolution methods.

Acknowledgments

This is the place to sincerely thank all the great people who helped and supported me over the course of my bachelor's thesis:

PD Dr. Klaus Dolag and *Dr. Alexander Beck*, for their great support and supervision. They did a formidable job, and between them I could not imagine it to be done any better.

Prof. Dr. Harald Lesch, for his undergraduate courses in astrophysics and his inspirational nature that made me remember why I wanted to become an astrophysicist in the first place.

Julius Donnert, for putting up with my endless questions and providing the great routines I worked on and with.

Madhura Killedar, for having to deal with my English quirks, and for answers big and small.

Christian Alig, for his work on the Arepo comparision runs.

David Schlachtberger, *Johann Weber* and *Rhea-Silvia Remus* for helping with all the minor questions and mayor concerns that keep coming up when one are new at something.

The rest of the Cast team and all the other great people at USM, for the warm welcome and wonderful time. I'd be happy to join them again, and soon.

My wonderful girlfriend, Alina, who had to put up with an even more scatterbrained me than usual, and my parents, who graciously made this all happen.

And last but not least you, the reader, for perusing this somewhat humongous bachelor's thesis: Thank you!

Munich, September 30, 2014
Pascal Ulrich Förster

A. Appendix: Code repository

A.1. Creating the initial conditions

The IDL procedure `make_data.pro` by Julius Donnert singelhandedly creates the initial conditions of the simulation as previously described in section 4.3. We have modified it mainly in regard to the specific setup of our simulation *First_Box* (FiBo), and a prominent alteration of that kind is that we only seed the largest 70 or so modes instead of the full spectrum.

`make_data.pro`

```
; This file contains routines to produce a gadget snapshot
; of a periodic box with a Kolmogorov velocity power spectrum
; from a glass file. The turbulent velocity contains the fraction
; X_turb of the thermal energy in the box. Temperature and density
; are set to values reasonable for a galaxy cluster atmosphere.
; Note: This is not a real GADGET snapshot as hsml is set to a
; reasonable constant. Upon running GADGET is going to recalculate it.
;
; Some routines from Klaus' library are needed:
; * write_head.pro
; * add_block.pro
;
; Note: you have to compile this twice

; make initial conditions for a turbulent box
pro make_box, npart, debug=debug

  if not keyword_set(npart) then $
    npart = 128L^3

  npart = ulong(npart)

  print, "Npart = ", npart

; Gadget units & chemistry
    m_unit = 1.989d43        ; [10^10 Msol]
    l_unit = 3.085678d21     ; [kpc]
    v_unit = 100000D         ; [km/s]

    H_frac = 0.76
  umol = 4.0/(5.0*H_frac+3.0)
    mp = 1.6726231e-24       ; proton mass cgs
    k_boltz = 1.3806580e-16 ; [cgs]
```

```
34
35 ; input values
36     fout = './snap_000'      ; output filename
37     X_turb = 0.3             ; E_turb / E_therm
38
39     boxsize = 3000D          ; [kpc]
40
41     T = 1d7                  ; [K]
42     rho = 1d-27/(m_unit/l_unit^3.)   ; [GADGET]
43     mass = rho * boxsize^3
44
45 ; make positions
46     pos = make_positions_hcp(boxsize, boxsize, boxsize, npart)
47
48 ; make data structures
49     head = make_head()
50   vel  = make_array(3,npart,/float)
51   id   = make_array(npart,/uint)
52   u  = make_array(npart,/float)
53   hsml = make_array(npart,/float)
54   dens = make_array(npart,/float)
55
56     head.npart = [1,0,0,0,0,0] * npart
57   head.massarr = [1,0,0,0,0,0] * mass/npart
58   head.time = 1
59   head.redshift = 1./head.time -1
60   head.flag_sfr = 0
61   head.flag_feedback = 0
62   head.parttotal = head.npart
63   head.flag_cooling = 0
64   head.num_files = 1
65   head.boxsize = boxsize
66   head.omega0 = 0.3
67   head.omegalambda = 0.7
68   head.hubbleparam = 0.7
69
70 ; IDs
71     id = ulindgen(npart)+1
72
73 ; internal energy
74     u[*] = u2t(T,/inv)
75
76 ; thermal energy
77     mass_cgs = mass * m_unit
78   Etherm = mass_cgs/umol/mp * 3./2. *k_boltz * T
79
80     print, 'Thermal Energy in Box [cgs] = ',Etherm
81
82 ; hsml
83     hsml[*] = 75.51 ; what GADGET finds on average for this glass
84
85 ; make velocities (the hard part)
86   Etherm /= m_unit *v_unit^2          ; cgs to gadget
```

```
87      v2 = npart*2.0*Etherm/mass*X_turb       ; total v^2 of particles
88
89      ngrid = long(npart^(1./3.))
90
91      vgrid = make_vel(ngrid, boxsize, v2, debug=debug)
92
93      ; Ncells = Npart
94      cellsize = boxsize / ngrid
95
96      vel[0,*] = idlNGP(pos/cellsize, vgrid[0,*,*,*])
97      vel[1,*] = idlNGP(pos/cellsize, vgrid[1,*,*,*])
98      vel[2,*] = idlNGP(pos/cellsize, vgrid[2,*,*,*])
99
100 ; density
101     dens[*] = rho
102
103 ; output
104     print, 'Writing : '+fout
105
106     write_head, fout, head
107   add_block, fout, float(pos), 'POS'
108   add_block, fout, float(vel), 'VEL'
109   add_block, fout, ulong(id), 'ID'
110   add_block, fout, float(u), 'U'
111   add_block, fout, float(hsml), 'HSML'
112   add_block, fout, float(dens), 'RHO'
113
114 end
115
116 ; make velocity grid (this is where the magic happens :-)
117 function make_vel, ngrid, boxsize, amp, debug=debug
118
119     seed = 14041981
120
121   kmin = 2*!pi/(boxsize)        ; box mode
122   kmax = !pi*ngrid/boxsize      ; Nyquist mode
123
124   vel = make_array(3,ngrid, ngrid, ngrid,/float, val=0)
125     kmag = make_array(ngrid,ngrid,ngrid,/float, val=0)
126     cdata = make_array(3,ngrid,ngrid,ngrid,/complex, val=0)
127     cdata_rl = make_array(ngrid,ngrid,ngrid, val=0,/double)
128     cdata_im = make_array(ngrid,ngrid,ngrid, val=0,/double)
129     iconj = 0
130     jconj = 0
131     kconj = 0
132
133   for axes=0,2 do begin
134     for i=0,ngrid-1 do $
135     for j=0,ngrid-1 do $
136     for k=0,ngrid/2 do begin
137 ; Generate k value  first (thank you Volker)
138             ; Define conjugated indizes of the grid
139       if i ne 0 then iconj = ngrid - i $
```

```
140             else iconj = 0
141       if j ne 0 then jconj = ngrid - j $
142             else jconj = 0
143       if k ne 0 then kconj = ngrid - k $
144             else kconj = 0
145
146             ; Define grid
147       if i LE ngrid/2. then kx = i * kmin $
148                else kx = -iconj * kmin
149
150       if j LE ngrid/2. then ky = j * kmin $
151                else ky = -jconj * kmin
152
153       if k LE ngrid/2. then kz = k * kmin $
154                  else kz = -kconj * kmin
155
156       kmag[i,j,k]  = sqrt(kx^2 + ky^2 + kz^2)
157
158
159       if kmag[i,j,k] GT kmax then $
160         continue     ; Only do a sphere in k space
161
162             if i+j+k eq 0 then $
163                 continue     ; no DC current
164           ; if (i eq ngrid/2) or (j eq ngrid/2) or (k eq ngrid/2) then
    $
165           ;      continue     ; no DC current
166
167             if i gt ngrid/2 then $
168                 continue     ; these are done via symmetry
169
170 ; Power spectrum P(k)
171       Pk = kmin* kolmog_3D(kmag[i,j,k],kmax,kmin)
172
173 ; Generate normal distributed random numbers with dispersion Pk
174 ; using Box Mueller method
175       A =  sqrt( -alog(randomu(seed,/double)) * Pk )
176       phase = 2.*!pi*randomu(seed,/double)
177
178 ; Cutting off all except the ~70 largest modes (Bauer&Springel2012)
179                         IF kmag[i,j,k] LT (6.25/4)*kmin THEN $
180                            A = 0
181                         IF kmag[i,j,k] GT (12.57/4)*kmin THEN $
182                            A = 0
183
184 ; Set power so we get a real vel after inverse FFT
185             if i gt 0 then begin     ; grid is hermitian in i>ngrid/2
186                 cdata_rl[i,j,k] = A * cos(phase)
187           cdata_im[i,j,k] = A * sin(phase)
188
189                 cdata_rl[iconj,jconj,kconj] = cdata_rl[i,j,k]
190           cdata_im[iconj,jconj,kconj] = -1*cdata_im[i,j,k]
191             end else begin  ; i = 0 needs special treatment
```

```
192                 if j eq 0 then begin ; first row
193
194                     if k gt ngrid/2. then $
195                         continue
196
197                     cdata_rl[i,j,k] = A * cos(phase)
198                 cdata_im[i,j,k] = A * sin(phase)
199
200                     cdata_rl[i,j,kconj] = cdata_rl[i,j,k]
201                 cdata_im[i,j,kconj] = -1*cdata_im[i,j,k]
202                   end else begin  ; j != 0 here
203                     if j gt ngrid/2. then $     ; rest of the plane
204                         continue
205
206                     cdata_rl[i,j,k] = A * cos(phase)
207                 cdata_im[i,j,k] = A * sin(phase)
208
209                     cdata_rl[i,jconj,kconj] = cdata_rl[i,j,k]
210                 cdata_im[i,jconj,kconj] = -1*cdata_im[i,j,k]
211                   end
212               end
213     end
214
215         cdata[axes,*,*,*] = COMPLEX(cdata_rl,cdata_im)
216         data = reform(FFT(cdata[0,*,*,*], /inverse, /double))/Ngrid^3
217
218 ; check if we got the symmetries correct
219         for i=0,ngrid^3-1 do $
220             if abs(imaginary(data[i])) gt 1e-7*abs(real_part(data[i]))
    then $
221                 stop
222
223 ; set velocities
224     vel[axes,*,*,*] =  real_part(data)
225
226   end
227
228 ; norm to total amplitude because kolmog_3D is wrong
229         norm = sqrt(total(vel[0,*,*,*]^2 +vel[1,*,*,*]^2 +vel
    [2,*,*,*]^2 ))
230
231         vel *=  sqrt(amp) / norm
232         cdata *=  sqrt(amp) / norm  ; Parseval's theorem -> ngrid^3
233
234 ; testing
235     if keyword_set(debug) then begin
236         ; total energy of data
237         v2_k = total(abs(FFT(vel[0,*,*,*]))^2. $
238                     +abs(FFT(vel[1,*,*,*]))^2. $
239                     +abs(FFT(vel[2,*,*,*]))^2.) * ngrid^3
240       print, 'v^2  in k space : '+strn(v2_k)
241
242         ; total energy in real space
```

```
243       v2 = total(vel[0,*,*,*]^2 +vel[1,*,*,*]^2 +vel[2,*,*,*]^2 )
244       print, 'v^2 in real space: '+strn(v2)
245       print, 'Requested v^2  : '+strn(amp)
246 
247         ; 3D Spectrum from data
248         lcData = cData[0,*,*,*]^2+cData[1,*,*,*]^2+cData[2,*,*,*]^2
249 
250         plot, kmag,sqrt(lcdata *4*!pi * kmin^2 )  $
251              ,xrange=[2e-3, 0.2],psym=3,yrange=[1e-8,1e3],/ylog,/xlog $
252              ,ytitle='sqrt(P(k) 4!7p!X k!Dmin!N!U2!N) [km/s]',xstyle=1 $
253              ,xtitle='k=2!7p!X/L [1/kpc]'
254       oplot, [1.,1]*kmin, [1e-10,1e10],col=16711680
255       oplot, [1.,1]*kmax, [1e-10,1e10],col=16711680
256 
257         ; 3D spectrum analytically
258       k = kmin + findgen(10000)/9999. * (kmax-kmin)
259       Pk =  kolmog_3D(k, kmax, kmin) * 8
260       oplot, k, sqrt(Pk * 4 * !pi * kmin^2), col=65280
261 
262         ; Spectrum from vel, this leaks into large k, because of
    numerical error
263         kData = make_array(3,ngrid,ngrid,ngrid)
264       kData[0,*,*,*] = abs(FFT(vel[0,*,*,*],/double)) * ngrid^3
265       kData[1,*,*,*] = abs(FFT(vel[1,*,*,*],/double)) * ngrid^3
266       kData[2,*,*,*] = abs(FFT(vel[2,*,*,*],/double)) * ngrid^3
267 
268         lkData = kData[0,*,*,*]^2 + kData[1,*,*,*]^2 + kData[2,*,*,*]^2
269 
270         oplot, kmag, sqrt(lkdata * 4 * !pi * kmin^2)  ,col=16711680,
    psym=3
271 
272         stop
273     end
274 
275   return, vel
276 end
277 
278 function kolmog_3D, k, kmax, kmin
279 ; here we require 1 = int^kmax_kmin dk P_0 * 4 pi k^2 k^(-11/3)
280     norm = (6*!pi * ( kmin^(-2./3.) - kmax^(-2./3.) ) )^(-1)
281   return, norm * k^(-11.0/3.0)
282 end
283 
284 ; sample Grid at Pos via NGP the IDL way
285 function idlNGP, pos, ingrid
286 
287     ngrid = n_elements(ingrid)^(1./3.)
288 
289     grid = reform(ingrid,ngrid,ngrid,ngrid )
290 
291     npos = n_elements(pos)/3
292 
293     u = reform(pos[0,*],npos)
```

```
294     v = reform(pos[1,*],npos)
295     w = reform(pos[2,*],npos)
296 
297     i = floor(u)
298     j = floor(v)
299     k = floor(w)
300 
301     return, grid[i,j,k]
302 end
303 
304 ; make a gadget header
305 function make_head
306   head = {         npart           : lonarr(6),  $
307       massarr      : dblarr(6),  $
308       time       :   double(1),  $
309       redshift     : double(1),  $
310       flag_sfr     : long(0),  $
311                          flag_feedback          : long(0),  $
312       parttotal    :   lonarr(6),  $
313       flag_cooling           : long(0),  $
314       num_files    : long(1),  $
315       boxsize      : double(1),  $
316       omega0       : double(0.3),$
317       omegalambda    : double(0.7),$
318       hubbleparam    : double(0.7),$
319                          flag_stellarage          :        long(0),$
320                          flag_metals              :        long(0), $
321                          npartTotalHighWord       :        lonarr(6), $
322             Labels                     :       bytarr(2,15),$
323                          la                       :        bytarr(256-6*4
    - 6*8 - 8 - 8 - 2*4 - 6*4 - 2*4 - 4*8 - 2*4 - 6*4 - 2*15)}
324 
325   return, head
326 
327 end
328 
329 ; convert internal energy to temperature for GADGET
330 FUNCTION u2t, u,  rad=rad, inv=inv, xH=xH, uvel=uvel, gamma=gamma
331 
332     if not keyword_set(gamma) then $
333         gamma = 5./3.
334 
335   IF NOT keyword_set(xH) THEN $
336     xH = 0.76
337 
338   IF NOT keyword_set(uvel) THEN $
339     uvel=1e5
340 
341   bk=1.380658d-16          ;k_boltzmann in cgs
342   prtn=1.672623d-24    ;m_proton in g
343 
344   yhelium = ( 1. - xH ) / ( 4 * xH )
345 
```

```
346   mean_mol_weight = (1. + 4. * yhelium) / (1. + 3. * yhelium + 1)
347
348   IF keyword_set(inv) THEN BEGIN
349     T = U
350     u = T /( (gamma-1) * uvel^2 * prtn * mean_mol_weight / bk)
351     return, u
352   END ELSE BEGIN
353     T = u * (gamma-1) * uvel^2 * prtn * mean_mol_weight / bk
354     return, T
355   END
356 END
357
358 ; optimal hcp particle distribution in a periodic
359 ; box, with ntot being a rough upper bound of particles to
360 ; distribute. return pos and ntot
361 ; ntot values like X^3 recommended for cubic boxes
362 function make_positions_hcp, Lx, Ly, Lz, ntot, debug=debug
363
364     if n_params() lt 3 then begin
365         print, 'Usage : pos = make_positions_hcp(Lx, Ly, Lz, ntot,
    debug=debug)'
366         print, '  Lx    : Boxsize in x direction'
367         print, '  Ly    : Boxsize in y direction'
368         print, '  Lz    : Boxsize in z direction'
369         print, '  ntot  : Max total number of particles'
370         print, '  debug : Show periodicity diagnostics'
371         return, -1
372     end
373
374     ; ntot is better divisible by two
375     if ntot mod 2 ne 0 then $
376         ntot--
377
378     ; find by combining spacings with Ntot
379     r = (sqrt(2.0D)*Lx*Ly*Lz/8.0D /ntot)^(1.0D/3D)
380
381     ; spacings
382     dx = 2.0D*r
383     dy = sqrt(3.0D)*r
384     dz = sqrt(6.0D)*2.0D/3.0D*r
385
386     ; particle numbers
387   np = make_array(3, val=0D)
388   np[*] = long(ntot^(1./3.))
389
390     ; enforce periodicity
391     dx += (lx-dx*np[0])/np[0]
392     dy += (ly-dy*np[1])/np[1]
393     dz += (lz-dz*np[2])/np[2]
394
395     ; diagnostics
396     print, 'Particle numbers : '
397     print, '      Nx = '+strn(np[0])
```

```
398     print, '      Ny = '+strn(np[1])
399     print, '      Nz = '+strn(np[2])
400     print, ' Total  = '+strn(np[0]*np[1]*np[2])
401     print, ' Wanted = '+strn(ntot)
402     print, ' Delta  = '+strn(double(ntot)-np[0]*np[1]*np[2])
403
404     ntot = np[0]*np[1]*np[2]
405
406     ; particle positions
407     x = make_array(np[0],np[1],np[2],/double)
408     y = make_array(np[0],np[1],np[2],/double)
409     z = make_array(np[0],np[1],np[2],/double)
410
411     idxarr = double(lindgen(np[0]))
412
413     ; A(0) plane
414     ; 0st row
415     x[*,0,0] = r + idxarr*dx
416     y[*,0,0] = r
417     z[*,0,0] = r
418
419     ; 1st row
420     x[*,1,0] = idxarr*dx
421     y[*,1,0] = r + dy
422     z[*,1,0] = r
423
424     ; A-plane
425     ; even rows
426     for i=2L,np[1]-1,2 do begin
427         x[*,i,0] = x[*,0,0]
428         y[*,i,0] = y[*,0,0] + i*dy
429         z[*,i,0] = z[*,0,0]
430     end
431
432     ; odd rows
433     for i=3L,np[1]-1,2 do begin
434         x[*,i,0] = x[*,1,0]
435         y[*,i,0] = y[*,1,0] + (i-1)*dy
436         z[*,i,0] = z[*,1,0]
437     end
438
439     ;B(1)-plane
440     ; 0rst row
441     x[*,0,1] = r + idxarr*dx
442     y[*,0,1] = 0
443     z[*,0,1] = r + dz
444
445     ; 1st row
446     x[*,1,1] = idxarr*dx
447     y[*,1,1] = dy
448     z[*,1,1] = r + dz
449
450     ; B-plane
```

```
451     ; even rows
452     for i=2L, np[1]-1, 2 do begin
453         x[*,i,1] = x[*,0,1]
454         y[*,i,1] = y[*,0,1] + i*dy
455         z[*,i,1] = z[*,0,1]
456     end
457 
458     ; odd rows
459     for i=3L, np[1]-1, 2 do begin
460         x[*,i,1] = x[*,1,1]
461         y[*,i,1] = y[*,1,1] + (i-1)*dy
462         z[*,i,1] = z[*,1,1]
463     end
464 
465     ; all planes
466     ; even planes
467     for i=2L, np[2]-1, 2 do begin
468         x[*,*,i] = x[*,*,0]
469         y[*,*,i] = y[*,*,0]
470         z[*,*,i] = z[*,*,0] + i*dz
471     end
472 
473     ; odd planes
474     for i=3L, np[2]-1, 2 do begin
475         x[*,*,i] = x[*,*,1]
476         y[*,*,i] = y[*,*,1]
477         z[*,*,i] = z[*,*,1] + (i-1)*dz
478     end
479 
480     ; make particle arrays
481     pos = make_array(3, ntot,/double)
482 
483     bin = ulong(0)
484     for i=0,np[0]-1 do $
485     for j=0,np[1]-1 do $
486     for k=0,np[2]-1 do begin
487         pos[0,bin] = x[i,j,k]
488         pos[1,bin] = y[i,j,k]
489         pos[2,bin] = z[i,j,k]
490         bin++
491     end
492 
493     ; randomize
494     seed = 14041981
495     for k=0, 2*ntot do begin
496         i = round(randomu(seed)*(ntot-1))
497         j = round(randomu(seed)*(ntot-1))
498 
499         tmp = pos[*,i]
500         pos[*,i] = pos[*,j]
501         pos[*,j] = tmp
502     end
503 
```

```
504     if keyword_set(debug) then begin
505         old_pmulti = !p.multi
506         !p.multi[1:2] = [3,2]
507 
508         ; check spacings
509         plot, x, y, psym=4, /iso, xrange=[0, Lx], yrange=[0, Ly], $
510             xtitle='x', ytitle='y'
511         oplot, x[0,0,0]+[0,dx], y[0,0,0]*[1.,1.], col=color(1)
512 
513         plot, x, z, psym=4, /iso, xrange=[0, Lx], yrange=[0, Lx], $
514             xtitle='x', ytitle='z'
515         oplot, x[0,0,0]*[1,1], z[0,0,0]+[0,dz], col=color(1)
516 
517         plot, y, z, psym=4, /iso, xrange=[0, Ly], yrange=[0, Lz], $
518             xtitle='y', ytitle='z'
519         oplot, y[0,0,0]+[0,dy], z[0,0,0]*[1.,1.], col=color(1)
520 
521         ; check periodicity
522         bad = where(x gt Lx or x lt 0, nbadx)
523         bad = where(y gt Ly or y lt 0, nbady)
524         bad = where(z gt Lz or z lt 0, nbadz)
525         print, 'Points outside the box  :'+strn(nbadx+nbady+nbadz)
526 
527         err = make_array(3, /double)
528         err[0] = Lx - dx*np[0]
529         err[1] = Ly - dy*np[1]
530         err[2] = Lz - dz*np[2]
531 
532         print, "Error in sampling periodicity : "
533         print, "    "+strn(err[0],len=7)+" particle spacings in x"
534         print, "    "+strn(err[1],len=7)+" particle spacings in y"
535         print, "    "+strn(err[2],len=7)+" particle spacings in z"
536 
537         x[*,*,*] += 2*dx
538         bad = where(x gt Lx)
539         x[bad] -= Lx
540 
541         y[*,*,*] += 2*dy
542         bad = where(y gt Ly)
543         y[bad] -= Ly
544 
545         z[*,*,*] += 2*dz
546         bad = where(z gt Lz)
547         z[bad] -= Lz
548 
549         plot, x, y, psym=4, /iso, xrange=[0, Lx], yrange=[0, Ly], $
550             xtitle='x', ytitle='y'
551         oplot, 2*dx * [1,1] , [0.,Ly], col=color(1)
552         oplot, [0,Lx] , 2*dy*[1,1], col=color(1)
553 
554         plot, x, z, psym=4, /iso, xrange=[0, Lx], yrange=[0, Lx], $
555             xtitle='x', ytitle='z'
556         oplot, [0,Lx] , 2*dz*[1,1], col=color(1)
```

```
557         oplot, 2*dx * [1,1] , [0.,Lz], col=color(1)
558
559         plot, y, z, psym=4, /iso, xrange=[0, Ly], yrange=[0, Lz], $
560             xtitle='y', ytitle='z'
561         oplot, 2*dy * [1,1] , [0.,Lz], col=color(1)
562         oplot, [0,Ly] , 2*dz*[1,1], col=color(1)
563
564         stop
565
566         !p.multi = old_pmulti
567
568         return, -1
569     end
570
571     return, float(pos)
572 end
```

A.2. Submitting the simulation for computation

The short script below has been used to run the simulation on the *dorc* mashines at USM. The specifications include the number of cores and the amount of memory (in GB) used as well as the path of the GADGET executable and the parameter file of the run.

`fibo_script_run3.sh`

```
1 #!/bin/bash
2
3 #PBS -N FiBo_res128_turb30_v2
4 ##PBS -o out.log
5 ##PBS -e err.log
6 #PBS -l ncpus=48
7 #PBS -l nodes=1:ppn=48
8 #PBS -l mem=24
9 #PBS -d /ptmp/foerster/First_Box/different_ICs/FiBo_res-128_turb-30_v2
10
11 ( mpiexec -np 48 ./P-Gadget3/P-Gadget3 paramfile_fibo 0 > out.log ) >&
     err.log
```

A.3. Automated binning of numerous snapshots

This custom bash-script was written to automatize the time-consuming binning of large numbers of snapshots into grid files. The process can take up to several minutes per file for higher resolutions.

`script_Sph2Grid.sh`

```
1 #!/bin/bash
```

```
2
3 FIRST=0  ##Number of the first snapshot file to be converted to grid
4 LAST=200  ##Number of the last snapshot file to be converted to grid
5
6 echo " "
7 echo "o
    ==========================================================================o
    "
8 echo "| Script for binning of great numbers of snapshots via Sph2Grid
      |"
9 echo "| Change the desired range in this scriptfile itself, if
    necessary |"
10 echo "o
    ==========================================================================o
    "
11 echo " "
12 echo "   Processing snapshots $(printf "%03d" $FIRST) to $(printf "%03d
    " $LAST)"
13 echo "   (this may take a while, approx 1 min per snapshot)"
14
15 for (( c=$FIRST; c<=$LAST; c++ ))
16 do
17   echo "-------------------------------------"
18   echo " "
19   echo "Performing task $(($c - $FIRST + 1))/$(($LAST - $FIRST + 1))"
20   snr=$(printf "%03d" $c)
21   echo "Sph2Grid: snap_$snr --> grid_$snr"
22   sed -i "s/snap_[0-9]*/snap_${snr}/g" ./sph2grid.par
23   sed -i "s/grid_[0-9]*/grid_${snr}/g" ./sph2grid.par
24
25   ##Run Sph2Grid (and wait for it <-- by default)
26   ./Sph2Grid sph2grid.par
27 done
28
29 echo "-------------------------------------"
30 echo " "
31 echo "  Done!"
32 echo " "
```

A.4. Plotting the power spectrum from the grid files

The IDL procedure `powerspectrum2.pro` has been used to extract the already calculated velocity power spectrum $P(k)$ from the data binned by Sph2Grid, and to plot it. It is based on the procedure `powerspectrum.pro` by Julius Donnert and has been modified extensively, for example to remove faulty data or to comfortably allow different types of output. All the power spectrum plots original to this thesis have been created with `powerspectrum2.pro`.

`powerspectrum2.pro`

```
; plot velocity powerspectrum on a grid made by Sph2Grid
; and compare to IDL FFT. Note that differences at small k are due
; to the different data layout. IDL includes the redundant Hermitian
; part of the FFT data, which gives differences in the binning.


pro powerspectrum_serial, fname=fname, fend=fend
  ;produces a series of plots and saves them as jpg, ready to be
    converted to gif later

    if not (keyword_set(fname) && keyword_set(fend)) then $
       begin
       print, 'Did not enter any of both necessary keywords:'
       print, '"fname=" (name of first file) and "fend=" (name of last
    file)'
       print, 'Aborting! Please restart properly.'
       STOP
       endif

    if not keyword_set(fname) then $
       begin
       print, 'Did not enter keyword "fname=" (name of first file)'
       print, 'Aborting! Please restart properly.'
       STOP
       endif

    if not keyword_set(fend) then $
       begin
       print, 'Did not enter keyword "fend=" (name of last file)'
       print, 'Aborting! Please restart properly.'
       STOP
       endif

    SET_PLOT, 'x'
    setcolors

      ;begin loop
    oname='really_new_plot_powerspectrum_000'
    fnumber=long(strcompress(strmid(fname,2,/reverse_offset),/
    remove_all))   ;fnumber=filenumber in long integer
    fnumber_str=strcompress(string(fnumber),/remove_all)
    strput, oname, fnumber_str, (strlen(oname)-strlen(fnumber_str))

    REPEAT BEGIN

        print, 'starting powerspectrum_subroutine with ' + fname
        powerspectrum_subroutine, fname=fname
        save_screen, oname

           ;preparing next round
        fname_old=fname
        fnumber=long(strcompress(strmid(fname,2,/reverse_offset),/
    remove_all))
```

```
50          fnumber=fnumber+1
51          fnumber_str=strcompress(string(fnumber),/remove_all)
52          strput, fname, fnumber_str, (strlen(fname_old)-strlen(
        fnumber_str))
53          strput, oname, fnumber_str, (strlen(oname)-strlen(fnumber_str))
54
55             ;exit clause
56          if (STRCMP(fname_old,fend, /fold_case) EQ 1) then $
57             fname=!NULL
58
59      ENDREP UNTIL not keyword_set(fname)
60
61      print, 'Done!'
62
63  end
64
65
66  pro powerspectrum_multi, fname=fname, f2name=f2name, f3name=f3name,
        twofiles=twofiles, threefiles=threefiles, block=block, ncols=ncols,
         nrows=nrows, nodetails=nodetails
67
68          if not keyword_set(block) then $
69                  block = 'VEL'
70
71      if not keyword_set(ncols) then $
72                  ncols = 1
73
74      if not keyword_set(nrows) then $
75                  nrows = 1
76
77         ;initialize multi-plot
78      !P.MULTI=[0,ncols,nrows,0,0]
79      SET_PLOT,'ps'
80      DEVICE,/ENCAPSULATED
81      DEVICE,FILE='plot_powerspectrum_multi.eps',xsize=10,ysize=22
82  ;     DEVICE,FILE='plot_powerspectrum_multi.eps',xsize=11,ysize=16.5
83  ;     DEVICE,FILE='plot_powerspectrum_multi.eps',xsize=15,ysize=22
84      DEVICE,COLOR=1
85      DEVICE,XOFFSET=1
86      DEVICE,YOFFSET=1
87     ;!X.OMARGIN = [8,1.5]
88     ;!Y.OMARGIN = [4,1]
89     ;!x.margin = [0,0]
90     ;!y.margin = [0,0]
91
92         ;read 1st filename of fname
93      filename=''
94      if not keyword_set(fname) then $
95                 begin
96                 read, filename, prompt='Enter Filename: '
97                 filename=strtrim(filename, 2)
98                 fname = filename
99                 endif
```

```
100
101       ;check for "threefiles" keyword
102     if keyword_set(threefiles) then $
103           begin
104             ;activate "twofiles", so entering "threefiles" suffices
105           twofiles=1
106           end
107
108       ;check for "twofiles" keyword
109     if keyword_set(twofiles) then $
110           begin
111             ;read 1st filename of f2name
112           filename=''
113           if not keyword_set(f2name) then $
114                 begin
115                 read, filename, prompt='Enter second Filename: '
116                 filename=strtrim(filename, 2)
117                 f2name = filename
118                 endif
119            endif
120
121       ;check for "threefiles" keyword
122     if keyword_set(threefiles) then $
123           begin
124             ;read 1st filename of f32name
125           filename=''
126           if not keyword_set(f3name) then $
127                 begin
128                 read, filename, prompt='Enter third Filename: '
129                 filename=strtrim(filename, 2)
130                 f3name = filename
131                 endif
132            endif
133
134
135       ;start counting
136     ccols=0
137     crows=0
138
139     REPEAT BEGIN
140           powerspectrum_subroutine, fname=fname, f2name=f2name, f3name=
    f3name, nodetails=nodetails
141
142             ;preparing next round
143           fname_old=fname
144           read, fname, prompt='Enter Filename: '
145
146           if (STRCMP(fname,'end',3, /fold_case) EQ 1) then $
147                 fname=!NULL
148
149           if (fname EQ fname_old) then $
150                 fname=!NULL
151
```

```
152             ;increase the count, check for end of plot by number
153         ccols+=ccols
154         if ccols ge ncols then $
155              begin
156              ccols = 0
157              crows+=crows
158              endif
159         if crows ge nrows then $
160              fname=!NULL
161
162             ;next round for second file
163         if keyword_set(twofiles) then $
164              begin
165              read, f2name, prompt='Enter second Filename: '
166              endif
167
168             ;next round for third file
169         if keyword_set(twofiles) then $
170              begin
171              read, f3name, prompt='Enter third Filename: '
172              endif
173     ENDREP UNTIL not keyword_set(fname)
174
175     DEVICE,/CLOSE
176 end
177
178
179 pro powerspectrum_subroutine, fname=fname, f2name=f2name, f3name=f3name
    , nodetails=nodetails, block=block
180
181     if not keyword_set(fname) then $
182                 fname = './grid_000'
183
184         if not keyword_set(block) then $
185                 block = 'VEL'
186
187     velgrid = readgrid(fname, block, head=head)
188
189     gridsize = head.gridsize
190     ngrid = head.ngrid
191     cellsize = gridsize / ngrid
192
193     kmin = 2*!pi/(gridsize)       ; box mode          ;original code,
    but
194     kmax = !pi*ngrid/gridsize     ; Nyquist mode      ;maybe not what I
    need?
195
196       ;read file before plotting
197     Pk = readgrid(fname, "VEL", /pk)
198     print, Pk                     ;for NAN control reasons
199     bin_pos = readgrid(fname, "KPK")
200
201       ;calculate time and make string (Pascal Foerster)
```

```
202     time=head.time/10.59d     ;divided by sound crossing time
203     time=ROUND(time*1000d)/1000d     ;round it off
204     time_str=strcompress(string(time),/remove_all)
205     time_str=strmid(time_str,0,5)
206 
207     ; plot
208     setcolors
209     fname_short=STRMID(fname, 2, /REVERSE_OFFSET)
210     plot, dist(2), /nodata, /xlog, /ylog, xrange=[(0.9*kmin),(1.2*kmax)
    ] $
211         ,xtitle='k = 2!4p!X/L [kpc!U-1!N]' , ytitle='P(k) [arb. units]'
    $
212         , xstyle=1,ystyle=1 $      ;  | only for individual plots ---->  ,
     yrange=minmax(Pk[where(Pk ne 0)]) $
213 ;CHANGE NAME OF THE RUN HERE! (P.Foerster)
214         , title= 'snap: '+fname_short+'     time: '+time_str+' x t!ISC!N'
    $ ;added by P.Foerster
215         , charsize=1.0 $      ;added by Pascal Foerster
216         , yrange=[1.0d-10,10]         ;added by Pascal Foerster, make them
     comparable (for just P(k))
217 
218       ; remove NANs through interpolation and plot corrected "Pk_good"
219     Pk_good = Pk
220     where_good = WHERE(FINITE(Pk_good,/NAN))
221     Pk_good[where_good] = (Pk[where_good - 1] + Pk[where_good + 1])/2.0
    d
222     Pk_good[SIZE(Pk_good, /DIMENSIONS)-1]=Pk[SIZE(Pk_good, /DIMENSIONS)
    -1]  ;correct last entry against possible "uplift"
223     oplot, bin_pos, Pk_good,col=mycolor(33),psym=-6,symsize=0.1      ;
    plotting NaN-free P(k) data
224 
225       ;ONLY IF there is a 2nd file:
226     if keyword_set(f2name) then $
227         begin
228           ;read file 2 before plotting
229         P2k = readgrid(f2name, "VEL", /pk)
230         print, P2k
231         bin2_pos = readgrid(f2name, "KPK")
232 
233           ; same corretions of P2k as for Pk (cleaning via
    interpolation)
234         P2k_good = P2k
235         where2_good = WHERE(FINITE(P2k_good,/NAN))
236         P2k_good[where2_good] = (P2k[where2_good - 1] + P2k[
    where2_good + 1])/2.0d
237         P2k_good[SIZE(P2k_good, /DIMENSIONS)-1]=P2k[SIZE(P2k_good, /
    DIMENSIONS)-1]   ;correct last entry against possible "uplift"
238         oplot, bin2_pos, P2k_good,col=mycolor(57),psym=-6,symsize=0.1
       ;plotting NaN-free P(k) data
239         endif
240 
241       ;ONLY IF there is a 3rd (!!) file:
242     if keyword_set(f3name) then $
```

```
243          begin
244 ;           ;read file 3 before plotting
245          P3k = readgrid(f3name, "VEL", /pk)
246          print, P3k
247          bin3_pos = readgrid(f3name, "KPK")
248 
249            ; same corretions of P3k as for Pk and P2k (cleaning via
    interpolation)
250          P3k_good = P3k
251          where3_good = WHERE(FINITE(P3k_good,/NAN))
252          P3k_good[where3_good] = (P3k[where3_good - 1] + P3k[
    where3_good + 1])/2.0d
253          P3k_good[SIZE(P3k_good, /DIMENSIONS)-1]=P3k[SIZE(P3k_good, /
    DIMENSIONS)-1]    ;correct last entry against possible "uplift"
254          oplot, bin3_pos, P3k_good,col=mycolor(48),psym=-6,symsize=0.1
       ;plotting NaN-free P(k) data
255          endif
256 
257       ;plot Kolmogorov_3D spectrum (-11/3)
258       oplot, bin_pos, bin_pos^(-11./3.)/2.7e9, col=mycolor(20),linestyle
    =3     ;added color and dotted lines
259 
260       ; plot average smoothing length
261       sname=fname
262       STRPUT, sname, 'snap', STRLEN(fname)-8
263       readnew, sname, hsml, 'HSML'
264       av_hsml = MEAN(hsml)
265       k_av_hsml=2.0*!pi/av_hsml
266       k_av_hsml_array=make_array(ngrid, VALUE=k_av_hsml, /DOUBLE)
267       vertical_line_array=dindgen(ngrid) * (1.0d2/(ngrid)) + 1d-20
268       oplot, k_av_hsml_array, vertical_line_array, linestyle=2, col=
    mycolor(22)
269       if not keyword_set(nodetails) then $
270          begin
271          xyouts, (0.98*k_av_hsml), 0.7d-9, 'k!ISML!N', alignment=1,
    charsize=0.9, color=mycolor(33)
272          endif
273 
274       ;ONLY IF there is a 2nd file: (will not done be done for 3rd -->
    Arepo)
275       if keyword_set(f2name) then $
276          begin
277          s2name=f2name
278          STRPUT, s2name, 'snap', STRLEN(f2name)-8
279          readnew, s2name, h2sml, 'HSML'
280          av_h2sml = MEAN(h2sml)
281          k_av_h2sml=2.0*!pi/av_h2sml
282          k_av_h2sml_array=make_array(ngrid, VALUE=k_av_h2sml, /DOUBLE)
283          vertical_line_array=dindgen(ngrid) * (1.0d2/(ngrid)) + 1d-20
284          oplot, k_av_h2sml_array, vertical_line_array, linestyle=2, col
    =mycolor(22)
285          if not keyword_set(nodetails) then $
286             begin
```

```
287             xyouts, (0.98*k_av_h2sml), 0.7d-9, 'k!ISML!N', alignment=1,
    charsize=0.9, color=mycolor(57)
288             endif
289          endif
290
291       ; plot Nyquist mode and box mode
292    k_Nyquist=!pi/cellsize
293    k_Nyquist_array=make_array(ngrid, VALUE=k_Nyquist, /DOUBLE)
294    k_box=2*!pi/gridsize
295    k_box_array=make_array(ngrid, VALUE=k_box, /DOUBLE)
296    vertical_line_array=dindgen(ngrid) * (1.0d2/(ngrid)) + 1d-20
297    oplot, k_Nyquist_array, vertical_line_array, linestyle=5, col=
    mycolor(22)
298    if not keyword_set(nodetails) then $
299       begin
300       xyouts, (0.98*k_Nyquist), 0.7d-9, 'k!INyquist!N', alignment=1,
    charsize=0.9
301
302       endif
303    oplot, k_box_array, vertical_line_array, linestyle=5, col=mycolor
    (22)
304    if not keyword_set(nodetails) then $
305       begin
306       xyouts, (1.02*k_box), 0.7d-9, 'k!Ibox!N', alignment=0, charsize
    =0.9
307       endif
308
309       ; plot k_seed_min and k_seed_max (see function make_vel @
    make_data.pro for more information)
310    k_seed_min=(6.25/4)*2*!pi/gridsize
311    k_seed_min_array=make_array(ngrid, VALUE=k_seed_min, /DOUBLE)
312    k_seed_max=(12.57/4)*2*!pi/gridsize
313    k_seed_max_array=make_array(ngrid, VALUE=k_seed_max, /DOUBLE)
314    vertical_line_array=dindgen(ngrid) * (1.0d2/(ngrid)) + 1d-20
315    oplot, k_seed_min_array, vertical_line_array, linestyle=4, col=
    mycolor(22)
316    if not keyword_set(nodetails) then $
317       begin
318       xyouts, (1.02*k_seed_min), 0.7d-9, 'k!ISEED,min!N', alignment=0,
    charsize=0.9
319       endif
320    oplot, k_seed_max_array, vertical_line_array, linestyle=4, col=
    mycolor(22)
321    if not keyword_set(nodetails) then $
322       begin
323       xyouts, (1.02*k_seed_max), 0.7d-9, 'k!ISEED,max!N', alignment=0,
    charsize=0.9
324       endif
325 end
```

B. Appendix: Configuration files

B.1. The GADGET compilation settings

The following configuration file named `Config.sh` determines the specific features of GADGET-3 available for later use when running the simulation on the compiled, executable version of it. Since there are numerous extensions and often more than a few methods to handle the same situation in GADGET, it is necessary to choose the compilation settings wisely. Also, only settings that were included at compile time can be activated later, off course.

`Config.sh`

```
# General code settings
PEANOHILBERT
WALLCLOCK
MYSORT
PERMUTATAION_OPTIMIZATION
MOREPARAMS
ALLOWEXTRAPARAMS
PERIODIC

AUTO_SWAP_ENDIAN_READIC  # Enables auto ENDIAN swapping for reading ICs
READ_HSML                # reads hsml from IC file

MULTIPLEDOMAINS=8

WENDLAND_C6_KERNEL
WC6_BIAS_CORRECTION
JD_VTURB

OPENMP=2
KD_ACTIVE_PARTICLE_LIST_FOR_OPENMP
SORT_FROM_L3
OMP_MYSORT
OMP_SORT

KD_BUFFER_MANAGEMENT=0.3
KD_HMAX_ESTIMATE
KD_RESTRICT_NEIGHBOURS

# Simulation Setup
DOUBLEPRECISION
NOGRAVITY

```

```
33 WAKEUP=3.0
34
35 ARTIFICIAL_CONDUCTIVITY     # enables Price-Monaghan art conductivity
36 TIME_DEP_ART_COND=1.0
37 AB_COND_GRAVITY=3.0
38 TIME_DEP_ART_VISC=1.0
39 AB_ART_VISC=1.0
40
41 AB_SHEAR
```

B.2. The Sph2Grid compilation settings

The compilation settings listed in the `Config` file below determine which physical quantities will be processed by Sph2Grid and how this processing will be done. Only data processed and included in the grid files will be available for further use, e.g. to be plotted.

`Config`

```
1 ## Compile Time Options  ##
2 MASS
3 RHO
4 VEL
5 #MOMENTUM
6 #INTENERGY
7 #BFLD
8 #SCALAR_BFLD          // magnetic field strength
9 #NPART
10 SCALARVEL             /* vel^2 */
11 #DENSVEL              /* Density weighted velocity */
12 #VELDISPERSION
13 #VELFILTERED          /* vel filtered on 5 cell cant be used with
       VELDISPERSION */
14
15 ## general options ##
16 PERIODIC              /* Periodic Boundary Conditions */
17
18 #VISIT                /* Visit output */
19
20 #TREE                 /* Build particle tree (not parallelised, buggy) */
21 #HSMLFIND             /* Find Hsml */
22
23 OVERSAMPLING          /* sample SPH kernel 27 times */
24
25 OPENMP                /* Use OpenMp threads on machines with shared mem
       */
26 #OMP_FFTW3            /* Use OpenMp in FFT calculations */
27
28 ## FFT options ##
29 FOURIERTRANSFORM      /* Master Switch FFT */
```

```
30 #HACKFFTW3            /* Hacks fftw_mpi_local_size_many for some cpu/
       ngrid combis */
31
32 KGRID                 /* store k values of the grid */
33
34 FFT_IDL_NORM          /* sets IDL FFT norm 1/(nx*ny*nz) */
35 #FFT_SYM_NORM         /* sets symmetric norm 1/sqrt(nx*ny*nz) */
36
37 #FFT_BIG_LAYOUT       /* store redundant data layout (=IDL) */
38
39 POWERSPECTRUM         /* compute P(k) on the fly */
40 #NO_FFT_OUTPUT        /* do not write FFT and kGrid data, only Pk */
41
42 #FFT_NO_ZEROPADDING   /* no zero padding, also when PERIODIC not set */
```

C. Appendix: Parameter files

C.1. The GADGET parameters

The following parameter file named `paramfile_fibo` contains all relevant intructions on how the GADGET executable is supposed to run the simulation. These settings are crucial, since they determine everything from computional parameters (e.g. time steps between calculations) to physical properties (e.g. viscosity settings) of the simulation. By changing any of those parameters, the outcome of the simulation may strongly change, and not necessarily in a physically sensible way.

`paramfile_fibo`

```
s%  Relevant files

InitCondFile       ICs/fibo_initial.ic
OutputDir               .

SnapshotFileBase        snap

EnergyFile         energy.txt
InfoFile           info.txt
TimingsFile        timings.txt
CpuFile            cpu.txt
TimebinFile        timebin.txt

RestartFile        restart

% CPU-time limit %%%%%%%%%%%%%% TimeLimitCPU*0.6=BetRest

TimeLimitCPU       2160000
CpuTimeBetRestartFile   90000
ResubmitOn         0
ResubmitCommand    xxx

% Code options

ICFormat                 2
SnapFormat               2
ComovingIntegrationOn 0

NumFilesPerSnapshot         1
NumFilesWrittenInParallel 1

CoolingOn 0
```

```
33 StarformationOn 0
34
35 %  Caracteristics of run
36
37 MaxMemSize             1500
38 TimeBegin              0.0
39 TimeMax                16.0
40
41 Omega0                  0
42 OmegaLambda            0
43
44 OmegaBaryon            0
45 HubbleParam            0.7    ; only needed for cooling
46
47 BoxSize                 3000.0
48 PeriodicBoundariesOn    1
49
50 % Output frequency
51
52 OutputListFilename   outputs_selection.txt
53 OutputListOn         0
54
55 TimeBetSnapshot         0.08
56 TimeOfFirstSnapshot     0
57
58 TimeBetStatistics       0.05
59
60 % Accuracy of time integration
61
62 TypeOfTimestepCriterion 0
63 ErrTolIntAccuracy        0.1
64
65 MaxSizeTimestep         0.1
66 MinSizeTimestep         1.0e-8
67
68 % Tree algorithm and force accuracy
69
70 ErrTolTheta             0.6 %%
71
72 TypeOfOpeningCriterion 1
73 ErrTolForceAcc          0.05 %%
74 MaxRMSDisplacementFac  0.25
75
76 %TreeUpdateFrequency     0.1
77
78 %  Parameters of SPH
79
80 %DesNumNgb             64
81 DesNumNgb              295                % if using quintic kernel
82 MaxNumNgbDeviation     0.01
83
84 ArtBulkViscConst      3.0
85 InitGasTemp           0
```

```
MinGasTemp              0.01
CourantFac              0.15

% Further code parameters

PartAllocFactor        3.0
BufferSize               20
TreeDomainUpdateFrequency    0.025

% System of units

UnitLength_in_cm          3.085678e21        ;  1.0 kpc /h
UnitMass_in_g             1.989e43           ;  solar masses
UnitVelocity_in_cm_per_s  1e5                ;  1 km/sec
GravityConstantInternal   0

% Softening lengths

MinGasHsmlFractional      0.0  % minimum gas smoothing in terms of the
    gravitational softening length

SofteningGas         1.0  % 0.4
SofteningHalo        1.0
SofteningDisk        1.0
SofteningBulge       1.0
SofteningStars       1.0
SofteningBndry       1.0

SofteningGasMaxPhys         1.0  %0.4
SofteningHaloMaxPhys        1.0
SofteningDiskMaxPhys        1.0
SofteningBulgeMaxPhys       1.0
SofteningStarsMaxPhys       1.0
SofteningBndryMaxPhys       1.0

%Uncommment for Magnetic dissipation
ArtificialMagneticDissipationConstant     0.25
ArtificialMagneticDissipationMin          0.001

ArtificialMagneticDissipationSource       15.0
ArtificialMagneticDissipationDecaytime    0.3

ArtCondConstant 1.0
ArtCondMin 0.0

ViscositySourceScaling 0.0
ViscosityDecayLength    4.0
ViscosityAlphaMin       0.025

ShockfinderNoiselevel 0.05
```

C.2. The Sph2Grid parameters

The parameter file named `sph2grid.par` informs the SPH2GRID executable on the properties of the simulation and instructs it on how it is supposed to bin the simulation data contained in the snapshot files to the grid. The following edition of the file is devised for one of our runs with a resolution of $N_{\text{part}} = 128^3$. Therefore a grid with $N_{\text{grid}} = 128$ points per dimensioin is used, and the number of final bins is 128.

`sph2grid.par`

```
% Sph2Grid Parameter File

Cosmology 0

% Image Properties in kpc comoving

Center_X 1500
Center_Y 1500
Center_Z 1500

Use_Barycenter 0

GridSize 3000
GridPoints 128

Input_File /ptmp/foerster/First_Box/different_ICs/FiBo_res-128_turb-30/
    snap_000
N_IOTasks 1

NoClobber 0
Output_File /ptmp/foerster/First_Box/different_ICs/FiBo_res-128_turb
    -30/grid_000

UnitLength_in_cm          3.085678e21        %  1.0 kpc
UnitMass_in_g             1.989e43           %  1.0e10 solar masses
UnitVelocity_in_cm_per_s      1e5                %  1 km/sec

Nbins 128                 % POWERSPECTRUM
```

Bibliography

Balsara, D. S. (1995). von Neumann stability analysis of smooth particle hydrodynamics – suggestions for optimal algorithms. *Journal of Computational Physics*, 121:357–372.

Bauer, A. and Springel, V. (2012). Subsonic turbulence in smoothed particle hydrodynamics and moving-mesh simulations. *Monthly Notices of the Royal Astronomical Society*, 423:2558–2578.

Box, G. E. P. and Muller, M. E. (1958). A Note on the Generation of Random Normal Deviates. *The Annals of Mathematical Statistics*, 29(2):610–611.

Cui, W., Liu, L., Yang, X., Wang, Y., Feng, L., and Springel, V. (2008). An Ideal Mass Assignment Scheme for Measuring the Power Spectrum with Fast Fourier Transforms. *The Astrophysical Journal*, 687:738–744.

Cullen, L. and Dehnen, W. (2010). Inviscid smoothed particle hydrodynamics. *Monthly Notices of the Royal Astronomical Society*, 408:669–683.

Daubechies, I., editor (1992). *Ten lectures on wavelets*.

Dolag, K., Vazza, F., Brunetti, G., and Tormen, G. (2005). Turbulent gas motions in galaxy cluster simulations: the role of smoothed particle hydrodynamics viscosity. *Monthly Notices of the Royal Astronomical Society*, 364:753–772.

Gingold, R. A. and Monaghan, J. J. (1977). Smoothed particle hydrodynamics – Theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389.

Hernquist, L. and Katz, N. (1989). TREESPH - A unification of SPH with the hierarchical tree method. *Astrophysical Journal Supplement Series*, 70:419–446.

Jasche, J., Kitaura, F. S., and Ensslin, T. A. (2009). Digital Signal Processing in Cosmology. *ArXiv e-prints*.

Jing, Y. P. (2005). Correcting for the Alias Effect When Measuring the Power Spectrum Using a Fast Fourier Transform. *The Astrophysical Journal*, 620:559–563.

Landau, L. D. and Lifshitz, E. M. (1959). *Fluid mechanics*, volume VI of *Course of theoretical physics*. Pergamon Press, 1st edition.

Landau, L. D. and Lifshitz, E. M. (1991). *Hydrodynamik*, volume VI of *Lehrbuch der theoretischen Physik in 10 Bänden. In dt. Sprache hrsg. von Paul Ziesche.* 5th edition.

Lucy, L. B. (1977). A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82:1013–1024.

Monaghan, J. J. (1992). Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, 30:543–574.

Morris, J. P. and Monaghan, J. J. (1997). A Switch to Reduce SPH Viscosity. *Journal of Computational Physics*, 136:41–50.

Price, D. J. (2008). Modelling discontinuities and Kelvin Helmholtz instabilities in SPH. *Journal of Computational Physics*, 227:10040–10057.

Price, D. J. (2012a). Resolving high Reynolds numbers in smoothed particle hydrodynamics simulations of subsonic turbulence. *Monthly Notices of the Royal Astronomical Society*, 420:L33–L37.

Price, D. J. (2012b). Smoothed particle hydrodynamics and magnetohydrodynamics. *Journal of Computational Physics*, 231:759–794.

Price, D. J. (2012c). Smoothed Particle Hydrodynamics: Things I Wish My Mother Taught Me. In Capuzzo-Dolcetta, R., Limongi, M., and Tornambè, A., editors, *Advances in Computational Astrophysics: Methods, Tools, and Outcome*, volume 453 of *Astronomical Society of the Pacific Conference Series*, page 249.

Springel, V. (2005). The cosmological simulation code GADGET-2. *Monthly Notices of the Royal Astronomical Society*, 364:1105–1134.

Springel, V. (2010a). E pur si muove: Galilean-invariant cosmological hydrodynamical simulations on a moving mesh. *Monthly Notices of the Royal Astronomical Society*, 401:791–851.

Springel, V. (2010b). Smoothed Particle Hydrodynamics in Astrophysics. *Annual Review of Astronomy and Astrophysics*, 48:391–430.

Springel, V. and Hernquist, L. (2002). Cosmological smoothed particle hydrodynamics simulations: the entropy equation. *Monthly Notices of the Royal Astronomical Society*, 333:649–664.

Springel, V., Yoshida, N., and White, S. D. M. (2001). GADGET: a code for collisionless and gasdynamical cosmological simulations. *New Astronomy*, 6:79–117.

Tasker, E. J., Brunino, R., Mitchell, N. L., Michielsen, D., Hopton, S., Pearce, F. R., Bryan, G. L., and Theuns, T. (2008). A test suite for quantitative comparison of hydrodynamic codes in astrophysics. *Monthly Notices of the Royal Astronomical Society*, 390:1267–1281.

Wadsley, J. W., Veeravalli, G., and Couchman, H. M. P. (2008). On the treatment of entropy mixing in numerical cosmology. *Monthly Notices of the Royal Astronomical Society*, 387:427–438.

List of Figures

Selbstständigkeitserklärung

Hiermit versichere ich,

dass ich diese Bachelorarbeit zum Thema “Turbulence in SPH” (deutsch: “Turbulenz in SPH”) selbstständig verfasst habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht.
Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann.

München, den 30. September 2014	
Ort, Datum	Unterschrift