Thesis for the Degree of Master of Science in Engineering Physics

Clustering and collisions in stochastic turbulence

Andreas Skyman

CHALMERS

Applied Physics
Chalmers University of Technology
January 2008

Clustering and collisions in stochastic turbulence

Andreas Skyman

Master of Science Thesis
Department of Applied Physics
Chalmers University of Technology
SE-412 96 Göteborg, Sweden

Abstract

This thesis deals with the behaviour of particles in a stochastic model of turbulence. Special attention is given to clustering and collisions of particles.

A stochastic model of turbulence is implemented and used to perform simulations of particle dynamics. Both advected particles, and particles with inertia are considered, as well as both incompressible and partly compressible flow.

The behaviour of the dynamics is quantified by studying the Lyapunov exponents, and their relation to clustering is discussed. Simulations of the advective dynamics are used to estimate the exponents for different degrees of compressibility. The simulations are shown to be in agreement with theory, and the theory's regime of validity is discussed in brief.

Collision rates are computed for particles advected in incompressible flow. The results agree with previous numerical studies, as well as with theoretical estimates for the initial transient and the long time steady state.

It is demonstrated that clustering of particles can occur both as result of compressiibility in the field and of particle inertia.

Acknowledgments

First and foremost, I would like to thank to my supervisor Bernhard Mehlig for inspiration, enthusiasm and patience.

I would also like to thank Kristian Gustavsson for taking the time to answer my questions, and to discuss and clarify muddy concepts and ideas.

Johanna Nordmark has helped me profusely with proof reading, and I humbly prostrate myself before her in unabridged gratitude. Without her unyielding fervour, this thesis would be severely lacking in readability and coherency both.

A special mention goes to the 7th floor Coffee Crew, for all our pleasant discussions of life the universe and everything.

Finally, I wish to extend my gratitude to gentlemen T. Sunhede and B. Dylan, who have unknowingly aided me throughout the work on this thesis. They have jump-started the creative process on many occasions, when the spark had gone for an early lunch. Many thanks!

Contents

List of Figures

List of Tables

1

Introduction

The topic of this thesis is the movement of particles in a randomly stirred fluid. A simplified model emulating near turbulence is implemented and studied numerically.

Two model properties associated with clustering are chiefly considered. The first is compressibility of the fluid. In a compressible fluid, particles will follow the pressure gradient and escape from regions of higher pressure, thus gathering in low pressure areas. The other, perhaps less intuitive effect, is that of the particles' inertia. The relation of particle inertia to clustering is discussed below. A third phenomenon leading to clustering is the formation of so called *caustics*. This occurs when particles with a small separation in position space have a large separation in momentum space, so that faster particles catch up with and pass slower [1, 2]. This phenomenon is not studied in this thesis.

One suggested explanation for the effect of inertia, is that areas of high vorticity tend do expunge particles trough centrifugal forces [3]. For this mechanism to be significant it is required that the particles are sufficiently massive compared to the damping action of the fluid medium. If this holds, the result is a tendency for higher particle density where the flow is more laminar. It has been suggested that the centrifuge mechanism cannot fully account for the clustering however, as it it cannot explain clustering in the limit of small damping [4]. When damping is small the inertia of the particles plays a considerable role. Thus particles moving through the fluid will experience much too rapid fluctuations in their local velocity field to be caught in the eddies.

In [4], the proposed mechanism is instead the clustering of particles onto fractal sets, as a result of random stirring. The dimensions of the fractal sets are determined by the so called Lyapunov exponents.

The two effects, to which attention is given here, are of notably different natures. The first is due to a property of the fluid, whereas the second stems from the interaction of fluid- and particle-properties, as quantified by the damping. They can be expected to factor out and are hence studied separately.

Also of interest in this thesis is how the rate at which particles collide varies over time. In this context particles of negligible inertia in an incompressible fluid are studied.

2

Background

2.1 Fluid dynamics

In the study of fluid dynamics one leg inevitably stands on the Navier-Stokes equation (2.1) [5].

$$\rho\partial_t \boldsymbol{u} + \rho\boldsymbol{u}\cdot\nabla\boldsymbol{u} = -\nabla p + \mu\nabla^2\boldsymbol{u} + \boldsymbol{F} \tag{2.1}$$

Here, and throughout this thesis, ∂_i is taken to mean differentiation with respect to the variable i. Analogously ∂_{ij} gives the derivative with respect to i and$j$, and ∂_{i^2} means twice differentiating with respect to i.

The Navier-Stokes equation governs the flow of a fluid built of infinitesimal fluid elements. Hence the terms in equation (2.1) have the dimension $\dim N/\dim m^3$ and not $\dim N$. Each term has a physical interpretation. The left hand side of equation (2.1) describes how the flow's velocity changes with time, as well as in space. The terms on right hand side describe the forces driving this change.

The term $\rho\partial_t\boldsymbol{u}$ on the left hand side of equation (2.1)is simply the acceleration of the fluid element, while the second left hand term $\rho\boldsymbol{u}\cdot\nabla\boldsymbol{u}$ is know as the inertia force. External forces, such as gravity, are described by the term $\boldsymbol{F}$, and $-\nabla p$ describes the force from the pressure gradient. More elusive is the second term of the right hand side, $\mu\nabla^2\boldsymbol{u}$. This is descriptive of the *viscous* forces in the flow. These are forces opposing the deformation of fluid elements, brought about by the shearing motions of vortices and other fluctuations. The coefficient μ is known as the fluid's *viscosity* [5]. It is a constant for the so called *Newtonian fluids* with which this thesis is concerned.

Much of a fluid's behaviour is tied to a dimensionless number know as the *Reynolds number*. It is denoted Re, and is defined

$$\mathrm{Re} \equiv \frac{\rho u_0 R}{\mu} = \frac{u_0 R}{\nu}, \tag{2.2}$$

where u_0 is a typical velocity and R a typical length for the flow. The quantity $\nu \equiv \mu/\rho$ is known as *kinematic viscosity*. This is used when it is convenient to combine the viscosity and density of the fluid into one parameter.

The Reynolds number can be given a physical interpretation through

$$\mathrm{Re} \sim \frac{inertial\ forces}{viscous\ forces}. \tag{2.3}$$

This can be seen by reviewing the terms of the Navier-Stokes equation described above. Given typical the values u_0 and R, the inertial and the viscous term should comply to $|\rho \boldsymbol{u} \cdot \nabla \boldsymbol{u}| \sim u_0^2/R$ and $|\mu \nabla^2 \boldsymbol{u}| \sim \mu u_0/R^2$ respectively. Dividing the former similarity by the latter, the Reynolds number is recovered on the right hand side, as seen in equation (2.4) [5].

$$\frac{|\rho \boldsymbol{u} \cdot \nabla \boldsymbol{u}|}{|\mu \nabla^2 \boldsymbol{u}|} \sim \frac{\rho u_0 R}{\mu} = \mathrm{Re}. \tag{2.4}$$

As equation (2.1) is highly nonlinear, its application more often than not requires some simplifications. A first step in this thesis is assuming that the external forces $\boldsymbol{F}$ are negligible. It is, however, often very costly to model the simulated fluid on even a simplification of the Navier-Stokes equations. Moreover, in this thesis the fluid is but the canvas on which the real objects of interest, the particles, move about. Considering this, it is justified turning to more drastic measures than dropping a few terms. The route taken here is shifting the focus from the dynamics themselves, trying instead to replicate their emergent statistical properties using a vastly simpler model. The fluid is modelled as a random but spatially and temporally correlated velocity field, with a stochastic time evolution. The spatial correlation is Gaussian and the temporal correlation is exponentially decaying.

There is only one time- and one length-scale characterising the modelled field. The latter fact points to one of the more obvious deviations from an exact model of turbulence, as one feature of turbulence is a multitude of associated characteristic distances, and the interaction between length scales [5]. Despite being a simplification in the extreme, it has been demonstrated in [4] that this method captures most of the features essential to the topics of interest in this thesis.

The equation of motion for the particles is derived taking less liberty with the foundations. Assuming that the flow is not turbulent at length scales comparable to the size of the particles, a known analytical result can be applied directly for the force exerted by the fluid on the particles.

2.2 Advection

The act of advecting something is the act of bringing a quantity of something somewhere[1]. *Advection* refers to transport by the macroscopic motions of a fluid, as opposed to *diffusion* which is due to microscopic fluctuations. These two are the principal components constituting *convection*.

What is advected by the flow depends on the context. Often the commodity is heat[2], or some other continuously distributed quantity. When effects of inertia can be neglected, in what is called the *advective limit* [7], the currents can instead carry small particles. These could for example be drifting plankton [5], dust particles in stellar accretion discs [8] or miniscule water droplets in clouds [9]. The various particles can be modelled as continua by focusing on particle density, but in this thesis they are regarded as discrete, localised spheres with small but finite radius. This approach can

[1] "Latin *advection, advectio* act of bringing, from *advehere* to carry to, from *ad-* + *vehere* to carry" [6]

[2] In the context of heat, the word *conduction* is often used instead of diffusion. Both terms refer to the same process.

easily be augmented to account for deviations from the advected limit, and is much more convenient when studying collisions.

2.3 Collisions

Collisions in advection or near-advection is a widely studied topic. It is important since collision rates can limit the speed at which other processes can take place. Examples of areas where detailed knowledge of the underlying processes is of importance range in size from the very small to the astronomical. At the lower end of the spectrum are chemical reactions and at an intermediate stage the formation of rain drops from smaller droplets in clouds. On a far greater length scale, and at a far greater distance from the everyday workings of man, is the formation of planets from the dust in the accretion discs around young stars.

One of the earliest accounts of such collisions was presented in [9] in 1956. The authors considered water droplets of equal size in an initially uniform distribution. These were suspended in a turbulent airflow, where the smallest eddies were much larger than the individual drops. An expression for the collision frequency was derived, and the change from an equilibrium distribution due to collisions was studied numerically. Inserting physically plausible parameters, the model was applied to study the effect of turbulence on the initiation of rain showers for different cloud types.

The validity regime of the result in [9] for the collision frequency, is studied in [7]. The conclusion is that the change from a uniform distribution will change the rate at which collisions occur, which was anticipated also in [9]. A theoretical estimate of the long time collision frequency is presented, and the transition between the two regimes is investigated through numerical simulations for a variety of velocity fields. When particles do not cluster, the result from [9] is shown to be an upper bound on the collision frequency. Compressibility in the background field, on the other hand, may lead to a larger rate of collisions as time progresses, and the geometry of particle set changes.

The importance of turbulence for the onset of rain is revisited in [10]. The focus there is the importance of caustics in increasing the collisions frequency and consequently the rate at which larger drops are formed.

3

Model

3.1 Description and parameters

3.1.1 Parameters

Table 3.1: Essential parameters

symbol:	dimension:	comment:
a	m	particle radius
m	kg	particle mass
η	m	correlation length of velocity field
τ	s	correlation time of velocity field
u_0	m/s	mean speed of velocity field
β	$-$	relative strength of compressible component of velocity field (see section 3.4.2)
L	m	size of system simulated
γ	s^{-1}	Stokes' damping constant
d	$-$	number of spatial dimensions
n_0	m^{-d}	number density of particles

The model for the flow and the particles is described by the parameters presented in table 3.1. The system modelled is characterised by the side length L and the number density of particles n_0. Periodic boundary conditions are assumed.

Table 3.2: Dimensionless combinations of parameters

symbol:	definition:	comment:
Ku	$u_0\tau/\eta$	Kubo number
St	$(\gamma\tau)^{-1}$	Stokes number
$-$	$n_0 a^d$	packing fraction
$-$	a/η	
$-$	η/L	

In table 3.2 the most important dimensionless combinations of parameters are presented. Of special interest in this thesis are the so called Kubo and Stokes numbers. The Kubo number is a dimensionless velocity, while the Stokes number is related to damping in such a way that a small St corresponds to strong damping [2]. When $\mathrm{St} \to 0$ the advective dynamics come into play (see section 3.3).

The packing fraction and the ratios a/η and η/L are all taken to be small. If $n_0 a^d$ is not small enough, particles will cover a large proportion of the available area, and there will be little space to move. There will also be a high probability of the particles overlapping after initialisation, which should be avoided for technical reasons (see section 6). Requiring $a/\eta \ll 1$ is necessary for the equation of motion to be a fair approximation of reality, as outlined in section 3.3. Finally, due to the finite size of the simulated system, too large a value of η/L might induce artefacts.

3.2 The particles

A particle is characterised by its radius a, mass m, position $\boldsymbol{r}$ and momentum $\boldsymbol{p} = \dot{\boldsymbol{r}}/m$. In the advective limit the particles act as 'passive tracers', meaning that they follow the flow along the streamlines exactly without neither affecting it nor each other. Their trajectories provide a map of the flow's time evolution. In this limit the momentum and mass of the particles are of no consequence.

Effects of inertia come into play when γ is finite. The particles then remain passive in the sense that the flow is still just a backdrop, untouched by the particles themselves, but because the momentum of the particles is no-longer negligible they will deviate from the tracer trajectories.

For simplicity, all particles are considered to be identical spheres, with the same mass and radius. The implementation does, however, allow for the possibility of using heterogeneous values for both of those parameters (see section 4).

3.3 Equations of motion

Stokes' law for the drag force on a sphere moving in a fluid reads

$$\boldsymbol{D} = 6\pi\mu a(\boldsymbol{u} - \dot{\boldsymbol{r}}) = m\gamma(\boldsymbol{u} - \dot{\boldsymbol{r}}), \tag{3.1}$$

where μ is the fluid's viscosity, $\dot{\boldsymbol{r}}$ is the velocity of the sphere and $\boldsymbol{u}$ the velocity of the fluid near the sphere [5]. The law is derived as a solution to the approximation of the Navier-Stokes equation (2.1) known as the *equation of creeping motion*, which reads [5]

$$-\nabla p = \mu\nabla^2\boldsymbol{u}. \tag{3.2}$$

In the dimensionless formulation of the Navier-Stokes equation, the neglected term is of order 1, while the remaining terms are of order Re^{-1}. Therefore the approximation is valid only for Reynolds numbers strictly smaller than 1[1] [5]. For the flow around a particle with radius a the Reynolds number of equation (2.2) becomes

$$\mathrm{Re}_a = \frac{\rho u_0 2a}{\mu} = \frac{12\pi\rho u_0 a^2}{m\gamma}, \tag{3.3}$$

[1] Experimental data compiled in [5] suggests that Stokes' law is applicable for $\mathrm{Re} < 0.5$.

where ρ is the density of the fluid. In the second step of equation (3.3), the second identity in equation (3.1) has been used. The density is not a parameter specified in the model, but it can be assumed to be much smaller than the mass density of the particles. A small ρ means that viscous forces dominate the inertial forces, thus guaranteeing a small Re [5]. This may seem to be in conflict with the expressed purpose of studying turbulence, since turbulent flows are characterised by large Re. For the fluid system as a whole, however, the relevant length scale is not the particle diameter a, but the length associated with the stirring of the fluid. This length is at least as large as the Kolmogorov length η, though generally it is much greater[2]. Since η is much larger than a, there is no conflict in requiring laminar flow around the particles and near turbulence on system scale.

The derivation of Stokes' law further requires the fluid to have uniform ambient velocity in the sphere's immediate surroundings. This translates to the condition $a \ll \eta$, which has to be fulfilled if equation (3.1) is to be used as an approximation of the forces acting on the particles. The equation of motion obtained is [1]

$$\ddot{\boldsymbol{r}} = \gamma(\boldsymbol{u} - \dot{\boldsymbol{r}}). \tag{3.4}$$

In this thesis, the concern is for the most part with the case of strong damping. Letting $\gamma \to \infty$, corresponding to $\mathrm{St} \to 0$, particle inertia will have a negligible effect and it is easily verified that in this limit the equation of motion (3.4) simplifies to

$$\dot{\boldsymbol{r}} = \boldsymbol{u}. \tag{3.5}$$

This implies that the particles follow the flow of the fluid's currents perfectly; this is the *advective limit*.

3.4 Modelling the flow

The velocity field $\boldsymbol{u}$ is generated as the combination of a compressible and an incompressible part. Both components are generated in essentially the same way through random Fourier sums. The resulting velocity field has a spatial correlation length η. It changes smoothly over time, with correlation time τ.

The flow has only one defining length scale, the correlation length η. This is interpreted as the Kolmogorov length [2], which is the characteristic size of the smallest vortices in the turbulent flow. In real turbulence there is a spectrum of relevant length scales associated with the flow [5].

In [2] it is argued that for fully developed turbulence $\mathrm{Ku} \sim 1$. Further it is argued that the model for the flow, described in section 3.4.1 below, can only be valid in the regime where the Kubo number conforms to $\mathrm{Ku} \leq 1$. This implies that the model is not applicable to fully developed turbulence.

3.4.1 Incompressible component

The incompressible, or *solenoidal*, part of the velocity field is created from a 'stream function' $\psi(\boldsymbol{r})$, so named because it is constant along streamlines [5]. Dealing here exclusively with two dimensional flows, this part of the velocity field is found as the

[2] In clouds, for example, the air is set in motion by convection currents, not uncommonly many kilometers in diameter.

curl of ψ multiplied with a vector orthonormal to the plane in which fluid is modelled[3]. This yields

$$\boldsymbol{u}_{sol} = \nabla \times \psi(\boldsymbol{r})\hat{\boldsymbol{n}}_z = \begin{bmatrix} \partial_y \psi \\ -\partial_x \psi \end{bmatrix}. \tag{3.6}$$

The stream function is constructed as a normalised Fourier sum,

$$\psi(\boldsymbol{r}) = u_0 \sqrt{\pi} \frac{\eta^2}{L} \sum_{\boldsymbol{k}} a_{\boldsymbol{k}} \exp\left(\mathrm{i}\boldsymbol{k} \cdot \boldsymbol{r} - \frac{\eta^2 k^2}{4} \right), \tag{3.7}$$

where the circular wave number $\boldsymbol{k} = \frac{2\pi}{L} \begin{bmatrix} n_x \\ n_y \end{bmatrix}$. The coefficients $a_{\boldsymbol{k}}$ are complex valued random numbers from a Gaussian distribution with the properties (3.8).

$$\begin{aligned} \langle a_{\boldsymbol{k}} \rangle &= 0 \\ a^*_{\boldsymbol{k}} &= a_{-\boldsymbol{k}} \\ \langle a_{\boldsymbol{k}}\, a_{-\boldsymbol{k}'} \rangle &= \delta_{\boldsymbol{k},\boldsymbol{k}'} \end{aligned} \tag{3.8}$$

3.4.2 Compressible component

To accommodate for compressibility, a function φ is constructed from the same recipe as ψ (equation (3.7)). This function is given its own set of Fourier coefficients, uncorrelated to those of ψ. The irrotational part of the velocity field is then defined as the gradient of φ, so that

$$\boldsymbol{u}_{pot} = \nabla \varphi(\boldsymbol{r}) = \begin{bmatrix} \partial_x \varphi \\ \partial_y \varphi \end{bmatrix}. \tag{3.9}$$

The function φ is referred to as the 'velocity potential'. It should be noted for clarity that it is not a physical potential in the same sense as the gravitational or electrostatic potentials, φ not sharing their close relation to energy [5].

The two components are combined giving a total velocity field

$$\boldsymbol{u} = \frac{\boldsymbol{u}_{sol} + \beta \boldsymbol{u}_{pot}}{\sqrt{1+\beta^2}}, \tag{3.10}$$

where the parameter β is used to tune the relative strengths of the two components. Letting $\beta = 0$ thus recovers the solenoidal velocity field of equation (3.6), whereas letting $\beta \to \infty$ creates a pure potential velocity field, as described in equation (3.9). Following [4], a parameter Γ is introduced such that

$$\Gamma = \frac{\beta^2 + 3}{3\beta^2 + 1}. \tag{3.11}$$

This parameter is bounded and takes values between $\frac{1}{3}$ (potential flow) and 3 (solenoidal flow). It characterises the degree of compressibility more conveniently than β.

[3] The same method can be employed in three dimensions as well, if $\psi(\boldsymbol{r})\hat{\boldsymbol{n}}_z$ is complemented with components in the two other Cartesian directions. The three components need not be correlated, depending on the physical situation being modelled.

3.4.3 Time dependence

The smooth time evolution of the field is approximated by subjecting the Fourier coefficients $a_{\boldsymbol{k}}$ to a discretised Ornstein-Uhlenbeck process, as described in [11] and [12]. The time evolution is written

$$a_{\boldsymbol{k}}(t+dt) = a_{\boldsymbol{k}}(t)\left(1-\frac{dt}{\tau}\right) + \sqrt{2\frac{dt}{\tau}}dw_{\boldsymbol{k}}, \tag{3.12}$$

where $dw_{\boldsymbol{k}}$ are complex Gaussian numbers, sharing the properties of $a_{\boldsymbol{k}}$ established in equations (3.8). It is easily verified that this process preserves these properties for $a_{\boldsymbol{k}}$, and results in an exponentially decaying correlation over time with half-life τ^{-1}.

4

Simulation

For more on the implementation, see appendix A.

4.1 Design overview

The simulations were implemented in C. Effort was put into making the program easy to use, as well as computationally efficient. At the heart of the design are the two structs[1] `particle` and `simulation`, which between the two of them keep track of all relevant parameters and variables for the particles. For collision detection they were joined by a struct called `box`.

Most operations to be performed on the particles have been implemented in such a way that they only have to be called with a pointer to the relevant `simulation` as an argument.

The particles are represented as elements in an array of `particle` structs. Each element thus keeps track of its own position and momentum, as well as radius and mass. In this thesis the latter two are kept constant and homogeneous for the entire ensemble, but in practice they could be changed dynamically, for example as particles collide.

For added flexibility, routines for reading parameter values from standardised settings-files were implemented.

4.2 Initialisation

The particles were assigned random initial positions uniformly in an box with side length L centered on the origin. The particle momenta were initialised to zero.

To avoid an extreme number of collisions the very first time step, the particle ensemble was checked for overlaps before the dynamics were switched on. Just as when checking for collisions (see section 4.4 below), one member of an overlapping pair was picked at random, but assigned a new position instead of removed from the simulation. The process was repeated until no particles overlapped. For consistency, overlaps were removed with the same relentless scrutiny for pairwise and test particle collisions both.

[1] See e.g. [13] for an introduction to C programming terms.

4.3 Movement

The particles move about in a box with side length L. Periodic boundary conditions are enforced, in order to conserve the number density of particles n_0. To avoid artefacts stemming from the finite size of the system, it is important to choose $\eta \ll L$.

The equations of motion are integrated in the simplest conceivable way using Euler's forward method, which reads [14]

$$\boldsymbol{r}(t + dt) = \boldsymbol{r}(t) + dt \cdot \dot{\boldsymbol{r}}(t). \tag{4.1}$$

In the advective limit (see section 3.3) $\dot{\boldsymbol{r}} = \boldsymbol{u}(\boldsymbol{r}(t))$. When inertial dynamics of equation (3.4) apply $\dot{\boldsymbol{r}} = \boldsymbol{p}(t)/m$. In this case, the momentum is also updated using Euler integration, so that

$$\boldsymbol{p}(t + dt) = \boldsymbol{p}(t) + dt \cdot \dot{\boldsymbol{p}}(t) = \boldsymbol{p}(t)(1 - \gamma dt) + m\gamma \boldsymbol{u}(\boldsymbol{r}(\boldsymbol{t}))dt. \tag{4.2}$$

The forward Euler method was chosen, despite it's notorious numerical instability [14], for the ease of implementation as well as for its transparency. Suspicions of numerical artefacts are easily verified or cast aside by decreasing the time step.

Care was taken to compute quantities in common for all particles, especially time independent factors, only when necessary. As the velocity of a particle depends only locally on the velocity field $\boldsymbol{u}$, this was only computed at the location of each particle. The Fourier sum was cut off at a maximal wave number $\pm k_{max} = \pm(2\pi/L)\lceil 2L/\eta \rceil$, corresponding to a weight $\leq \mathrm{e}^{-4\pi^2}$ for the least significant Fourier component. The maximal wave number is inversely proportional to the size smallest features regarded in the simulation.

Heeding a resource saving piece of advice from [12], the complex exponential in equation (3.7) was rewritten as trigonometric functions. Euler's formulae and trigonometrical identities [15] were applied in order to obtain an expression where the spatial dimensions where separated. This allowed for creating a table of evaluated trigonometric terms depending on only one spatial variable. Thus, when evaluating the sum for each particle, instead of evaluating the exponential explicitly for every single $a_{\boldsymbol{k}}$, the table can be consulted, and the appropriate terms combined.

4.4 Collisions

If two particles, both with their respective collision flag $\xi = 1$, come within a distance less than the sum of their radii, a collision is noted.

In principle the particles can have any radius, but for this simulation all particles have the same radius a. It is important to choose parameters so that $a \ll u_0 dt$, otherwise particles may jump through each other, when they should really have collided.

At a collision event, one of the involved particles is always removed from the simulation. In practice removing a particle means setting the internal flag $\xi = 0$ in the corresponding `particle` struct, telling the functions that it is to be skipped henceforth. The test particle is always left untouched, but when checking pairwise collisions the particle to remain is chosen by the toss of a fair coin. No new particles are added to compensate this loss, as this could not be done in compliance with the emergent statistics of the ensemble. It was found however, that for the small fraction of removed particles (about 4% for test particle and 17% for pairwise collisons[2] at $t = 100000\,\tau$)

[2] 17% might seem very much, but bearing in mind that the crucial quantity is the available number of pairs this is in no way alarming.

the resulting decrease in collision frequency R was negligible. Decreasing the number of particles also decreases the computational intensity as a welcome synergetic effect.

The removal of one particle from the system upon collision can be thought of as the two drops merging into one particle of comparable size as in a completely inelastic collision. As noted in both [9] and [12], the removal of particles changes the spatial particle distribution and affects the collision rate directly as time progresses. When allowing particles to collide with each other many times, for example by resetting ξ when the particles have steered clear of each other, the transient result from [9] never ceases to be accurate [12].

The actual implementation of collisions is severely unphysical away from the advective limit, since it conserves neither mass nor momentum. It would be a simple matter letting the particle which remains after the collision have the combined mass and inertia of the pair. Further, the particle's position could be shifted to the centre of mass coordinates of the two, and perhaps its radius increased. Whether such corrections would significantly influence the collision rates is, however, beyond the scope of this thesis.

In order to save resources when checking for collisions, the system containing the particles is divided into a lattice of N^d d-dimensional boxes. In each time step the particles are put in the appropriate box. When using a test particle, the box in which this resides and the surrounding boxes (eight in total in two dimensions) are checked for particles. In the case of pairwise collisions the boxes are traversed in order and collisions are checked for each resident particle with particles in the same box and in surrounding boxes. Checking all eight neighbouring boxes would be redundant, only three need be checked to cover all combinations.

The idea behind this boxing is that, instead of having to check collisions with every particle, it is only necessary to calculate the distance to the few particles residing in the neighbouring boxes. If the number of boxes is chosen so that there is approximately one box for every particle, then only an average of 8 checks are needed for collisions with a test particle in two dimensions. As a comparison, checking collisions between the test particle and every other particle would require $n_0 - 1$ checks. For pairwise collisions, only $4n_0$ checks are needed on average when boxing, in contrast to $\frac{n_0(n_0-1)}{2}$ checks when checking between every pair. This is of course based on the assumption that the particles are uniformly distributed, which they in general are not. Therefore this gives an upper bound for the number of operations which can be saved by this method. It is notable though that the number of operations needed with this method is one order of n_0 lower than that of the naïve implementation. The boxing algorithm itself is of order n_0.

5

Lyapunov exponents

5.1 Definition and significance

The Lyapunov exponents are quantities describing how quickly trajectories (of e.g. particles), initially infinitesimally close in phase space, will separate or coalesce under the dynamics governing their time evolution. The exponents have dimension s^{-1} and are termed rate constants [2]. The separation of two trajectories roughly obeys equation (5.1) [16].

$$|d\boldsymbol{r}(t)| \approx |d\boldsymbol{r}(0)|\mathrm{e}^{\lambda_1 t} \tag{5.1}$$

For two dimensions, two Lyapunov exponents $\lambda_1 > \lambda_2$ are defined. The first exponent is related to the time evolution of the short distance dr, which is the length of the vector pointing between two trajectories near to each other. Two such separation vectors span a parallelogram with the small area dA. The second exponent is related to how this area changes with time[1] [1]. The relations are

$$\lambda_1 = \lim_{t\to\infty} \frac{1}{t} \ln \left| \frac{dr(t)}{dr(0)} \right|, \tag{5.2}$$

$$\lambda_1 + \lambda_2 = \lim_{t\to\infty} \frac{1}{t} \ln \left| \frac{dA(t)}{dA(0)} \right|. \tag{5.3}$$

A trajectory's sensitivity to perturbations in the starting conditions, for which chaotic systems are notorious, is captured by the Lyapunov exponents. Thus they can have utility as a measure of how chaotic a given system is. A positive Lyapunov exponent λ_1 is characteristic of a chaotic system, whereas a negative value means that nearby trajectories will converge [16]. In the context of this thesis, it means that particles will tend to cluster.

The sum in equation (5.3) is positive iff $\lambda_1 > 0$, but it can be negative regardless of the sign of λ_1. If this quantity is negative, it means that the area dA is stretched

[1] For three dimensions, a third exponent, $\lambda_3 < \lambda_2$, is defined. In analogy with the lower dimensional cases, λ_3 is related to dV, the small volume of a parallelepiped defined by three separation vectors.

and compressed to a line, onto which the trajectories converge [1]. This is also a cause of particle clustering.

A theoretical estimate of the first Lyapunov exponent, valid for small Ku, is presented in [4]. The theoretical result is for the equation of motion (3.4) in a partly compressible field. By letting $\gamma \to \infty$ the advective case is recovered. In this limit the Lyapunov exponent is given by

$$\lambda_1 \tau = \mathrm{Ku}^2, \tag{5.4}$$

or, allowing for compressibility, by

$$\lambda_1 \tau = -2\frac{1+3\beta^2}{1+\beta^2}(1-\Gamma)\,\mathrm{Ku}^2 = 2\frac{\Gamma-1}{\Gamma+1}\mathrm{Ku}^2. \tag{5.5}$$

For the derivation of these formulae, see appendix B.1.

In the incompressible case the dynamics must be area preserving. From the discussion above, this translates to $\lambda_1 + \lambda_2 = 0 \Leftrightarrow \lambda_2 = -\lambda_1$. The second Lyapunov exponent for compressible fields is not studied in this thesis.

For a thorough review of the Lyapunov exponents and other statistical properties of turbulent fluids see [17].

5.2 Method

The Lyapunov exponents are sought as *dynamical averages* over long time. The idea behind this is that any typical trajectory will, given time, explore the entire phase space [16]. Following equation (5.1) it is tempting to numerically estimate the first Lyapunov exponent by

$$\lambda_1 = \lim_{n\to\infty} \frac{1}{n\,dt} \ln \frac{|d\boldsymbol{r}_n|}{|d\boldsymbol{r}_0|}, \tag{5.6}$$

where $d\boldsymbol{r}_n \equiv d\boldsymbol{r}(n\,dt)$ and dr_0 is ideally infinitesimal. As is hinted by the d in the $d\boldsymbol{r}$ of equation (5.1), however, the Lyapunov exponent is descriptive of the local behaviour of trajectories. Estimating it by observing a single trajectory's separation from it's initial coordinates as $t \to \infty$ is therefore not an appropriate method, though for particularly well behaved systems it is plausible. More reliable and computationally efficient methods, capturing more explicitly the local properties of the Lyapunov exponents, can be devised by noting that

$$\ln \frac{|d\boldsymbol{r}_n|}{|d\boldsymbol{r}_0|} = \ln \prod_{l=0}^{n-1} \frac{|d\boldsymbol{r}_{l+1}|}{|d\boldsymbol{r}_l|} = \sum_{l=0}^{n-1} \ln \frac{|d\boldsymbol{r}_{l+1}|}{|d\boldsymbol{r}_l|} = \sum_{l=0}^{n-1} \ln |\mathbf{J}_l|. \tag{5.7}$$

Here $|\mathbf{J}_l| \equiv |\mathbf{J}(d\boldsymbol{r}_l)|$ is the determinant of the Jacobi matrix $\mathbf{J}_l$, evaluated at $d\boldsymbol{r}_l$. This describes the mapping between consecutive points in phase space by the discretised dynamics [16]. For the advective dynamics of equation (3.5) this is

$$\mathbf{J}(\boldsymbol{r}) = \begin{bmatrix} \partial_{x_n} x_{n+1} & \partial_{y_n} x_{n+1} \\ \partial_{x_n} y_{n+1} & \partial_{y_n} y_{n+1} \end{bmatrix} = \underset{2\times 2}{\mathbf{1}} + dt \left(\begin{bmatrix} \partial_{xy}\psi & \partial_{yy}\psi \\ \partial_{xx}\psi & -\partial_{xy}\psi \end{bmatrix} + \beta \begin{bmatrix} \partial_{xx}\varphi & \partial_{xy}\varphi \\ \partial_{xy}\varphi & \partial_{yy}\varphi \end{bmatrix} \right) / \sqrt{1+\beta^2}, \tag{5.8}$$

and for the dynamics with inertia, described by equation (3.4), it is

$$\tilde{\mathbf{J}}(\boldsymbol{r},\boldsymbol{p}) \;=\; \begin{bmatrix} \partial_{\boldsymbol{r}_n}\boldsymbol{r}_{n+1} & \partial_{\boldsymbol{p}_n}\boldsymbol{r}_{n+1} \\ \partial_{\boldsymbol{r}_n}\boldsymbol{p}_{n+1} & \partial_{\boldsymbol{p}_n}\boldsymbol{p}_{n+1} \end{bmatrix} \;=\; \begin{bmatrix} \underset{2\times 2}{\mathbf{1}} & m^{-1}\underset{2\times 2}{\mathbf{1}} \\ m\gamma\left(\mathbf{J}(\boldsymbol{r}) - \underset{2\times 2}{\mathbf{1}}\right) & -\gamma dt\,\underset{2\times 2}{\mathbf{1}} \end{bmatrix}. \quad (5.9)$$

Here $\underset{m\times m}{\mathbf{1}}$ denotes the identity matrix of dimension n.

The next step is to compute the QR-factorisation of the matrix product of all n $\mathbf{J}_i$. Letting this product be denoted by $\mathbf{J}^{(n)}$, this means splitting $\mathbf{J}^{(n)}$ into a matrix $\mathbf{Q}_n$ and an upper-triangular matrix $\mathbf{R}$, such that $\mathbf{Q}_n\mathbf{R} = \mathbf{J}^{(n)}$. Following [18], let $\mathbf{Q}_0 = \underset{m\times m}{\mathbf{1}}$. The factorisation can then be performed in steps, so that

$$\mathbf{J}^{(n)} = \mathbf{J}_{n-1}\mathbf{J}_{n-2}\ldots\mathbf{J}_1\mathbf{J}_0 = \prod_{l=0}^{n-1}\mathbf{J}_l = \left(\prod_{l=1}^{n-1}\mathbf{J}_l\right)\mathbf{J}_0\mathbf{Q}_0 = \left(\prod_{l=2}^{n-1}\mathbf{J}_l\right)(\mathbf{J}_1\mathbf{Q}_1)\mathbf{R}_1 = \left(\prod_{l=3}^{n-1}\mathbf{J}_l\right)(\mathbf{J}_2\mathbf{Q}_2)(\mathbf{R}_2\mathbf{R}_1) = \ldots = \mathbf{Q}_n\left(\prod_{l=1}^{n}\mathbf{R}_l\right) = \mathbf{Q}_n\mathbf{R}. \quad (5.10)$$

The QR-factorisation $\mathbf{J}_l\mathbf{Q}_l = \mathbf{Q}_{l+1}\mathbf{R}_{l+1}$ is computed on the fly in each time step. The diagonal elements of $\mathbf{R}$ are the products of the corresponding diagonal elements of the $\mathbf{R}_i$ matrices, owing to their upper-triangular form. Now, assuming that the initial separation $dr_0 \to 0$, the approximate Lyapunov exponents are extracted from $\mathbf{R}$ in the manner of equation (5.11) [18].

$$\lambda_i \approx \frac{1}{n\,dt}\ln|\mathbf{R}[i,i]| = \frac{1}{n\,dt}\sum_{l=1}^{n}\ln|\mathbf{R}_l[i,i]| \quad (5.11)$$

The diagonal elements of $\mathbf{R}$ can grow very large. Therefore, in order to avoid rounding errors [14], computing the sum in equation (5.11) is numerically sounder than estimating the exponents directly from $\mathbf{R}$.

The method employed for the QR factorisation was Gram-Schmidt orthogonalisation. It was chosen mainly because it is simple to implement. As discussed in [18], this may not be the best choice of method. This is because, despite being up to par with other methods in computational efficiency, it can exhibit numerical instability. Since the results (see section 5.3) agree very well with theoretical predictions, however, this instability cannot have had a significant impact on the simulations. Nonetheless, this is a factor worth keeping in mind if designing a Lyapunov exponent calculator, since such numerical errors are notoriously hard to debug.

For details on the Gram-Schmidt recipe for QR-factorisation, and for examples of other methods, see e.g. [14].

5.3 Results

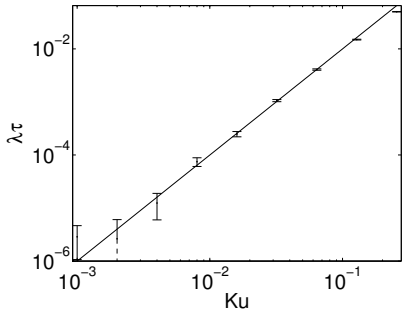
The Lyapunov exponents λ_1 and λ_2 were calculated using the program. The results, showing the dependence of λ_1 on Ku, are presented in figures 5.1–5.2 for various degrees of compressibility.

Even though the first Lyapunov exponent $\lambda_1 > 1$ for flows dominated by the solenoidal component, as seen in figures 5.1(a)–5.1(c), clustering is still observed when

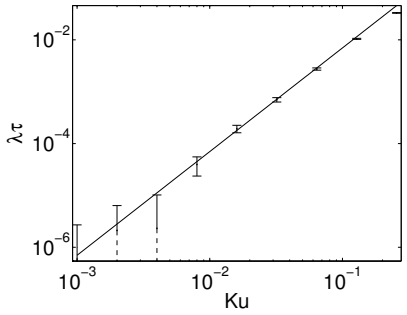
a compressible component is present. This is evident in figure 7.1(a), and implies that $\lambda_1 + \lambda_2 < 0$ due to shearing forces in the fluid. No such clustering is seen in flows lacking a compressible component.

When $\Gamma < 1$ the compressible component of the velocity field dominates. Just as expected, $\Gamma = 1$ is the limiting case where λ_1 changes sign, as can be observed in figures 5.1(d)–5.2(c).

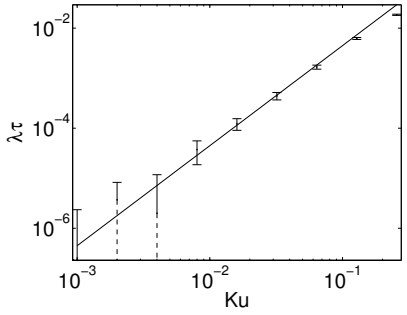
The area preservation property of the incompressible field lends itself as a tool for assessing, qualitatively, the numerical stability of the program. In figure 5.3(a) the computed value of $\lambda_1 + \lambda_2$ is shown as a function of Ku for an incompressible velocity field. As can be seen , the deviation from theory is small compared to the values of λ_i. At worst, the deviation and the exponents are still separated by two orders of magnitude. This indicates an acceptable stability, even away from the limit of small Ku, which is of primary interest. Since Euler's method is only first-order accurate, the global error is expected to be proportional to dt [14]. By fitting a polynomial to the data, however, the deviation from theory is observed to have a definite quadratic dependence on the time step dt, which suggests a better stability than the expected. This behaviour can be seen in in figure 5.3(b). Fore more on the accuracy and stability of Euler's method, see [14].



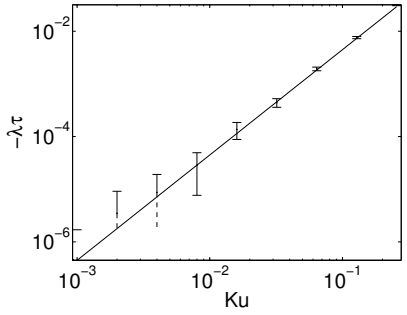
(a) $\Gamma = 3$, purely solenoidal field



(b) $\Gamma \approx 2.08$

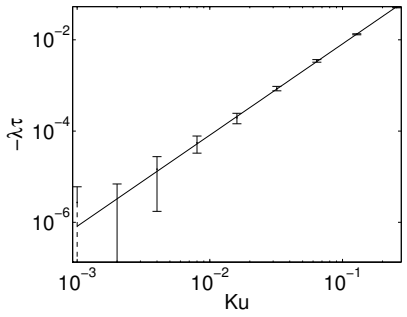


(c) $\Gamma \approx 1.58$

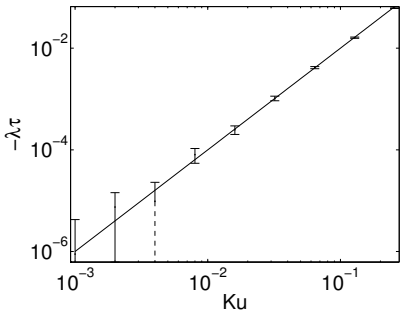


(d) $\Gamma \approx 0.63$

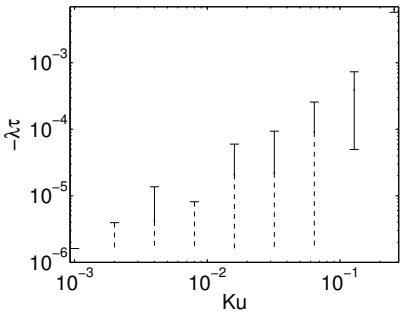
Figure 5.1: Logarithmic plots of the Lyapunov exponent λ_1 showing the deviation from theory (—) as Ku increases, with $\eta = 0.1$ and $\tau = 4 \cdot 10^{-3}$ fixed. Each data point (•) is an average of 100 runs up to $t = 125000\,\tau$, with $dt = 5 \cdot 10^{-5}$. Error bars show a confidence interval of two standard deviations. A dashed line indicates that the error bar extends below the x-axis, and therefore cannot be shown in a logarithmic plot.



(a) $\Gamma \approx 0.42$

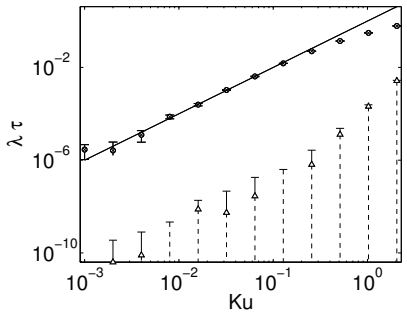


(b) $\Gamma = \frac{1}{3}$, pure potential field

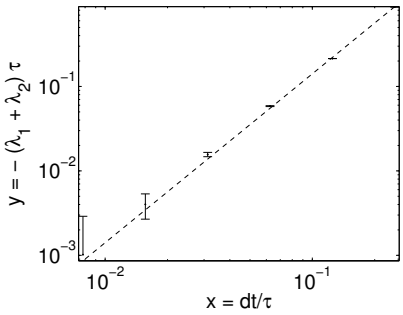


(c) $\Gamma = 1.0$, theory predicts $\lambda_1 = 0$

Figure 5.2: Logarithmic plots of the Lyapunov exponent λ_1 showing the deviation from theory (—) as Ku increases, with $\eta = 0.1$ and $\tau = 4 \cdot 10^{-3}$ fixed. Each data point (•) is an average of 100 runs up to $t = 125000\,\tau$, with $dt = 5 \cdot 10^{-5}$. Error bars show a confidence interval of two standard deviations. A dashed line indicates that the error bar extends below the x-axis, and therefore cannot be shown in a logarithmic plot.



(a) Comparison of $-(\lambda_1 + \lambda_2)\tau$ (△) and the respective values of $\lambda_1\tau$ (◯) and $-\lambda_2\tau$ (×), as functions of Ku with $\eta = 0.1$ and $\tau = 4 \cdot 10^{-3}$ fixed. Theoretical estimate $|\lambda_i|\tau = \mathrm{Ku}^2$ (—) included for comparison. Each data point is an average of 100 runs up to $t = 125000\,\tau$ with $dt = 5 \cdot 10^{-5}$.



(b) The sum $\lambda_1 + \lambda_2$ in the limit of large Ku ($u_0 = 51.2$, $\eta = 0.1$, $\tau = 4 \cdot 10^{-3} \rightarrow \mathrm{Ku} = 2.048$) for different values of dt. Data (•) is compared to a curve fit $y = Ax^2$ (- - -). Each data point is an average of 200 runs up to $t = 125000\,\tau$.

Figure 5.3: Logarithmic plots of the sum of the Lyapunov exponents for $\Gamma = 3$, a purely solenoidal field. Theory predicts that the sum $\lambda_1 + \lambda_2 = 0$ in the solenoidal limit. Error bars show a confidence interval of two standard deviations. A dashed line indicates that the error bar extends below the x-axis, and therefore cannot be shown in a logarithmic plot.

6

Collisions

6.1 Preliminaries

Simulations were performed in order to study the time dependence of the collision frequency $R(t)$. This collision rate was defined as the accumulated number of collisions at time t divided by t. Due to time constraints, the simulations were restricted to advection in incompressible flows. Both pairwise collisions and collisions with a test particle have been studied and compared to theory, as well as to a previous numerical study in [7].

6.2 Predictions

Collision frequency for the short time transient, which was first studied by Saffman and Turner [9], as well as for the long time steady state, were presented in [7]. They are presented in equations (6.1) and (6.2) respectively. The expressions are valid for collisions with a test particle. For pairwise collisions, the density of pairs is the relevant quantity, not the number density. The total number of possible pairs that can be formed from n_0 particles is $\frac{n_0(n_0-1)}{2}$. Consequently, the equations should be multiplied with $(n_0-1)/2$ when applied to pairwise collisions.

$$R_0 = \frac{8\sqrt{\pi}}{\sqrt{\Gamma+1}}\frac{n_0 a^2 u_0}{\eta} = \{\Gamma = 3\} = 4\sqrt{\pi}\frac{n_0 a^2 u_0}{\eta} \tag{6.1}$$

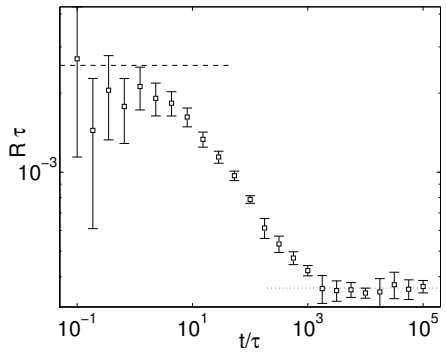
$$R_\infty = 4\pi\frac{\Gamma-1}{\Gamma+1}\mathrm{Ku}\, n_0 u_0 \left(\frac{2a}{\eta}\right)^{D_2} = \left\{ \begin{array}{l} D_2 = \Gamma - 1, \\ \Gamma = 3 \end{array} \right\} = 8\pi\frac{\mathrm{Ku}\, n_0 u_0 a^2}{\eta} \tag{6.2}$$

6.3 Practice

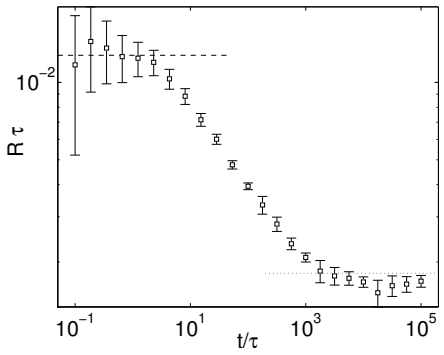
The results of the simulations comply with both theoretical predictions and the results presented in [7].

Acquiring reliable data for short times required considerably more runs than for longer times. It was also necessary to use a shorter time step for the initial transient, in order to avoid numerical artefacts. Therefore dt was divided by ten for $t < \tau$.

When checking pairwise collision frequencies, the radius proved another obstacle. It was found that using too large a radius gave extremely unreliable short time statistics. This was solved by using a radius one tenth of that specified for collisions with a test particle when checking pairwise collisions.



(a) Collision rate over time for collisions with a test particle, with $a = 0.003$. Error bars show a confidence interval of one standard deviation.



(b) Collision rate over time for pairwise collisions, with $a = 0.0003$. Error bars show a confidence interval of two standard deviations.

Figure 6.1: Logarithmic plots of collision frequencies for particles moving in an incompressible flow ($\Gamma = 3$), under advective dynamics ($\mathrm{St} = 0$). For both simulations $u_0 = 1.0$, $\eta = 0.1$ and $\tau = 4 \cdot 10^{-3} \Rightarrow \mathrm{Ku} = 0.04$. Short time theory (- - -) from equation (6.1) and long time theory ($\cdots$) from equation (6.2). For 6.1(b), both equations have been multiplied by $(n_0 - 1)/2$ (see section 6.2).

7

Discussion

7.1 Lyapunov exponents

As seen in figures 5.1–5.2, the data produced by the program shows a good correspondence with the theory derived in [4] for the advective case, and presented here as equations (5.4)–(5.5). The deviation from theory is more appreciable for velocity fields with $\Gamma > 1$, meaning that the solenoidal component dominates. The deviation becomes significant for $\mathrm{Ku} \gtrsim 0.1$. A deviation is expected as Kubo increases, since the theoretical prediction is derived for small Ku. From figure 5.3, along with the discussion in sections 5.3 and 7.4, it can be surmised that the deviation observed is the true deviation, and not a numerical artefact.

7.2 Collisions

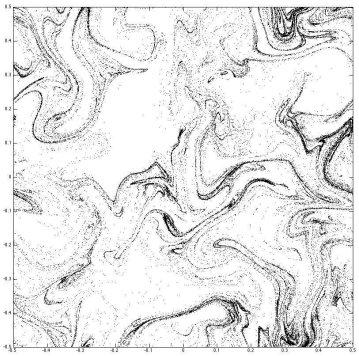
The collision frequencies measured by the program are consistent with those of previous studies, and the results are in accord with theoretical estimates for both the short and the long time limit [7], as can be seen in figure 6.1. For the parameter values used, the short time theory, equation (6.1), is valid for $t \lesssim \tau$. The transition to a steady state, in agreement with the long time theory of equation (6.2), is completed for $t \gtrsim 1000\,\tau$.

7.3 Clustering

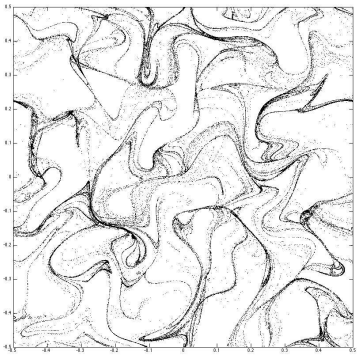
Two instances of clustering are seen in figure 7.1. Figure 7.1(a) shows clustering in the advective limit, due to a compressible component in the background field. Figure 7.1(b), on the other hand, shows clustering away from the advective limit. Here the flow is purely solenoidal, and the clustering is instead due to finite damping.

The Stokes number for the simulation portrayed in figure 7.1(b) is as high as $\mathrm{St} = 100$, meaning that the damping is slight. This agrees with the position held in [4], that the centrifuge mechanism of [3] cannot fully account the clustering, as the regime where that explanation is valid is supposedly $\mathrm{St} \sim 1$.

Though the snapshots in figure 7.1 are similar, they have quantitative differences. One of the more apparent dissimilarities is that, in the advective limit (figure 7.1(a)),



(a) Clustering of advected particles due to the compressibility of the flow.
$\Gamma = 2.08$, $\gamma = \infty \Rightarrow \mathrm{St} = 0$



(b) Clustering of particles in an incompressible flow, due to particle inertia.
$\Gamma = 3$, $\gamma = 2.5 \Rightarrow \mathrm{St} = 100.0$

Figure 7.1: Particles suspended in different flows. Similar clustering of particles occurs in all cases, despite very different contexts. Parameter values in common for the two plots are $\tau = 0.04$, $\eta = 0.1$ and $u_0 = 1.0 \Rightarrow \mathrm{Ku} = 0.04$. Snapshots are from simulation time $t = 10000\tau$, with time step $dt = 4 \cdot 10^{-4}$.

the lines, onto which the particles cluster, do not cross. This is not the case when the inertial dynamics are at work. The reason behind this is that, in the advective limit, a particle's velocity is determined solely by its position in the velocity field. Therefore the future time evolution of the particle trajectories must be uniquely specified by the coordinates the particles are at. If trajectories crossed, they would have to merge, or else the velocity would be ambiguous at their point of intersection. Trajectories can, however, intersect at points where the velocity $\boldsymbol{u} = 0$ [5]. Away from the advective limit each particle also has two independent coordinates in momentum-space[1], which are relevant for determining the velocity. Figure 7.1(b) shows only a projection of the particles onto physical space, and the projected trajectories can cross without ambiguity.

7.4 The time step

As discussed in section 5, in the case of advection in an incompressible medium, there is no mechanism bringing particles of negligible inertia together. Still, when looking at the quantity $\lambda_1 + \lambda_2$, this can appear not to hold for the simulations. The sum of the two first Lyapunov exponents quantifies what becomes of the area of an small fluid element over time. As seen in figure 5.3 $\lambda_1 + \lambda_2 < 0$, implying that shearing forces in the fluid causes the area to shrink, and consequently particles to cluster. It is easily verified, by varying the time step dt, that this phenomenon is due to the imperfect Euler integration.

If deliberately choosing much too large a time step, patterns strikingly similar to those shown in figure 7.1 can be observed. Analysing how a numerical error can give results so similar to what is indeed expected and observed for other parameters, can

[1] In the advective limit, the coordinates in momentum-space are uniquely determined by the coordinates in ordinary space.

give valuable insights into how future errors can be avoided.

One explanation for these artefacts is that the integration introduces 'inertia–like' effects. When the background velocity field changes direction too quickly, the finite step size of the Euler integration causes the integrated trajectory to overshoot its target, and spiral out from its true path. Particles thus tend to diverge from areas of high vorticity quicker than they ought to. This is similar to the centrifugal mechanism for particles with inertia, suggested by Maxey in [3].

The quantity to keep in check here is the average distance a particle travels each time step, namely $u_0 dt$. Combining this with the η, the defining length scale in the system[2], gives the dimensionless ratio $\frac{u_0 dt}{\eta}$, which resembles the Kubo number (see table 3.2). If this is large, particles will tend skip across the velocity field. Like stones skipping on a water surface, they interact with the bakcground only at distinctly separated points in space. In a sense, the finer details of the field are missed, due to sampling it at too large intervals. This is reminiscent of the sampling theorem of Fourier analysis [19]. Applying this theorem to the system would imply choosing dt so that $\frac{u_0 dt}{\eta} \leq \frac{1}{2}$, but this was concluded not to guarantee the absence of numerical artefacts. A more sensible advice is quite simply $\frac{u_0 dt}{\eta} \ll 1$.

Even though the artefacts cannot be removed completely, the knowledge of their nature can be used to get a qualitative measure of the validity of the simulations. If it is observed that $|\lambda_1 + \lambda_2| \ll |\lambda_i|$ does not hold, then the time step is not sufficiently small. Such a comparison is presented in figure 5.3(a). There it can be seen that the error, even at its worst, is several orders of magnitude smaller than the Lyapunov exponents. Furthermore, as seen in figure 5.3(b), this error is quadratic in dt. This knowledge may be of use when considering which time step to choose.

The natural solution for keeping this faulty integration in check is decreasing the time step. Since this unavoidably leads to an increase in computational intensity, a more attractive idea is the use of an adaptive time step, dependent on the current local magnitude of $\boldsymbol{u}$. Since the velocity field changes randomly over time, however, having different time steps for different particles in the ensemble would be impractical, unless the time evolution was calculated in advance. Instead, not recalculating $\boldsymbol{u}$ every time step, for particles in tranquil regions of the velocity field, might be a plausible way of saving resources.

7.5 Improvements

A way of saving computations, based on the properties (3.8) of the Fourier coefficients of the field, is suggested in [12]. There the symmetry $a^*_{\boldsymbol{k}} = a_{-\boldsymbol{k}}$ is utilised in order to skip calculating the imaginary part of $a_{\boldsymbol{k}}$ entirely. This effectively cuts the number of Gaussian deviates to be drawn in half. Because the construction of such deviates is a costly procedure, this could ostensibly improve the run time a lot if implemented correctly.

An increased the computational efficiency of the integration could also be achieved by implemented some sort of adaptive time step. It is argued above that a relevant quantity to keep in check, in this context, is the ratio $\frac{u(\boldsymbol{r}(t))dt}{\eta}$, evaluated at the location of the individual particles. If this is too large, the finer structure of the flow is not captured. On the other hand, if this is small, there may be leeway for time saving schemes. In sufficiently still areas of the flow, it may not be necessary to

[2] Another possible choice of length would be the system size L, but having $u_0 dt$ in the order of L is wholly implausible.

recalculate the $\boldsymbol{u}(\boldsymbol{r})$ every time step for all particles. When designing the adaptive algorithm, the data indicating how the error depends on the time step, presented in figure 5.3(b), may be of use. Other principles for choosing the time step are discussed in [14].

In order to increase the numerical stability of longer simulations, it may be worthwhile implementing a more stable integration algorithm than Euler's forward method, as well as a more physically plausible collision routine. For more on the stability and accuracy of Euler's method, and of other numerical methods, see e.g. [14].

When calculating Lyapunov exponents, a method other than Gram-Schmidt orthogonalisation should ideally be used, since that is not numerically stable [18]. It need, however, only be modified slightly to achieve a boost in numerical stability, as described in [14], wherein there are also several examples of how to implement other algorithms for finding the QR-factorisation.

8

Conclusions

A stochastic model of turbulence was implemented and used to perform simulations of particle dynamics. Both advected particles, and particles with inertia were considered, as well as both incompressible and partly compressible flow.

In order to asses the behaviour of the dynamics quantitatively, the Lyapunov exponents were studied. These were estimated from the simulated advective dynamics for different degrees of compressibility. The results were shown to be in agreement with theory valid for the limit of small Ku. The deviation from theory was observed to be more pronounced for fields where the solenoidal component dominates the flow, but was found to be slight for $\mathrm{Ku} \lesssim 1$ regardless of compressibility.

Collision rates for particles advected in incompressible flow were computed. The results agree very well with previous numerical studies, as well as with theoretical estimates for the initial transient and the long time steady state. The transient theory was found to be valid for times $t \lesssim \tau$. Steady state was reached at about $t \approx 1000\tau$.

Clustering of particles was demonstrated to occur as a result of compressibility in the field, as well as of particle inertia. The two phenomena were studied separately. Inertia-induced clustering was observed in the regime of small damping, which strengthens the position that the centrifugal mechanism presented in [3] cannot fully explain observations. A comparison of the contribution from this mechanism and from the fractal clustering presented in [4] remains to be done.

A

Program documentation

For the programs used in this thesis three function libraries have been implemented. Below is a documentation of the data types and functions contained in each of them, as well as a brief example of how they can be implemented. For more on the implementation see section 4.

A.1 The Simulation library

The Simulation library is a library containing functions for simulating particles moving in stochastic, time correlated flows, as discussed in this thesis.

A.1.1 Data types

`struct particle;`

The contents of `struct particle` are presented in table A.1. For technical reasons the value of `DIM` is by default defined to 3 in `simulation.h` and need not be changed.

Table A.1: Contents of `struct particle`

name:	type:	description:
`pos`	`double[]`	array of size `DIM` for storing particle position
`momentum`	`double[]`	array of size `DIM` for storing particle momentum
`u`	`double[]`	array of size `DIM` for storing the velocity field at the particle's position
`radius`	`double`	particle radius
`mass`	`double`	particle mass
`chi`	`int`	collision parameter, set to 0 after collision
`Box`	`box *`	pointer to the `box` keeping the particle when checking collisions (see section A.2)
`next`	`particle *`	pointer to next `particle` in the same box, `NULL` for last particle in list (see section A.2)

```
struct simulation;
```

The contents of `struct simulation` are presented in table A.2.

other types

Aside from the more complicated structures, the library also defines the enumerated type `boolean` with values `false` (= 0) and `true` (= 1) and the structure `complex` which contains two doubles `Re` and `Im`.

A.1.2 Functions

Below are descriptions of the functions available in the Simulation library. They are presented with their function prototype and brief note on their use. The functions use pointers to objects of types `particle` and `simulation`. Hence, a reference in the text to e.g. a "variable `beta` in `S`" should be understood as meaning "variable `beta` in `simulation` pointed to by `S`".

```
void init_sim(int n0, int k_max, simulation *S);
```

Sets parameters `n0` and `k_max` in `S` and then allocates space for `ak`, `ak_phi` and `PsiC` according to values of `k_max`, `beta` and `dim`. If $0.0 < \beta < 10000.0$ Fourier coefficients `ak` and `ak_phi` are both initialised. For $\beta = 0.0$ memory is prepared only for `ak`, and for $\beta \geq 10000.0$ only for `ak_phi`. All three arrays are of size $(2k_{max} + 1)^{\texttt{dim}}$.

```
void reset_particle(particle *P, simulation S);
```

Radius of `P` is set and then a position is drawn randomly from a uniform distribution in a box with width and dimension specified by `S`. Initial momentum is set to 0.0 and collision parameter χ is set to 1. Pointers `Box` and `next` (see section A.2) are both initialised to `NULL`.

```
void reset_particles(simulation *S);
```

Calls `reset_particle()` for each of the n_0 particles in `S`.

```
particle *init_particles(simulation *S);
```

Reads parameters from `S` and then allocates memory for `n0` objects of type `particle`. Calls `reset_particles()` to set initial values or all particles. Function returns pointer to the initialised array.

```
void free_sim(simulation *S);
```

Frees space occupied by `simulation` pointed to by `S`.

```
void free_particle(particle *P);
```

Frees space occupied by `particle` pointed at by `P`.

```
void new_ak(simulation *S);
```

Fills `ak` and/or `ak_phi` (see `init_sim` above) in `S` with Gaussian random numbers with the properties presented in (3.8). This function uses the NRrand library.

Table A.2: Contents of `struct simulation`

name:	type:	description:
`L`	`double`	width of system
`radius`	`double`	radius of particles
`mass`	`double`	mass of particles
`eta`	`double`	correlation length for velocity field
`tau`	`double`	correlation time for velocity field
`u0`	`double`	mean velocity of velocity field
`dt`	`double`	time step
`t_max`	`double`	maximum simulation time
`beta`	`double`	parameter for setting compressibility (details in section 3.4.2)
`gamma`	`double`	damping constant (details in equation (3.4))
`t`	`double`	current simulation time
`A`	`double`	constant coefficient for velocity field
`Ku`	`double`	Kubo number (see table 3.2)
`St`	`double`	Stokes number (see table 3.2)
`Gamma`	`double`	measure of compressibility (see section 3.4.2)
`PsiC`	`double **`	array of size $(2k_{max}+1)^{\texttt{dim}}$ for storing k-dependent coefficients of velocity field
`ak`	`complex **`	array of size $(2k_{max}+1)^{\texttt{dim}}$ for storing Fourier coefficients for ψ
`ak_phi`	`complex **`	array of size $(2k_{max}+1)^{\texttt{dim}}$ for storing Fourier coefficients for φ
`n0`	`int`	number of particles initially
`dim`	`int`	number of spatial dimensions considered
`n`	`int`	for storing number of particles at a given time
`k_max`	`int`	When calculating Ψ and Ψ only Fourier coefficients with $\lvert k\rvert \leq (2\pi/L)k_{max}$ are included in the sum. It is recommended to choose $k_{max} \geq 2L/\eta$.
`pairwise`	`boolean`	set to `true` to check pairwise collisions
`testparticle`	`boolean`	set to `true` to check collisions with test particle at the origin
`timecorrelated`	`boolean`	toggles time correlation in the velocity field
`periodic`	`boolean`	toggles periodic boundary conditions
`intertial`	`boolean`	toggles inertial dynamics (see equation (3.4) for details)
`synthetic`	`boolean`	toggles advection in synthetic field
`P`	`particle *`	points to the `particle` array containing the simulated particles

`void update_ak(simulation *S);`

Subjects `ak` and/or `ak_phi` (see `init_sim` above) in `S` to an Ornstein-Uhlenbeck process as described in section 3.4.3, which introduces a random drift with correlation time `tau`, still keeping the properties of the distribution the same. This function uses the NRrand library.

`void make_Psiconst(simulation *S);`

Fills array `PsiC` in `S` with values calculated from settings stored in `S`. This function has to be called before simulation starts, but only this once as long as the physical parameters u_0, L or η remain unchanged.

`void get_velocity(particle *P, simulation *S);`

Calculates the velocity field `u`, described in section 3.4, at the position of `P` using specifications from `S`.

`void get_velocities(simulation *S);`

Goes through all `n0` particles in `S` and calls `get_velocity()` for each particle with $\chi = 1$.

`void move_particle(particle *P, simulation *S);`

Moves `particle P` using Euler integration, equation (4.1). Depending on the value of `inertial` in `S`, either inertial (equation (3.4)) or advective dynamics (equation (3.5)) are used.

`void move_particles(simulation *S);`

Goes through all `n0` particles in `S` and calls `move_particle()` for each particle with $\chi = 1$.

`void print_sim(FILE *target, simulation *S);`

Prints simulation parameters stored in `S` to the file specified by stream `target`. Data is stored in a format designed to be interpretable by *Matlab*.

`void args_from_file(FILE *settings, simulation *S);`

Reads simulation parameters from the file specified in stream `settings` into `simulation S`. Settings file has to follow the specifications detailed in section A.1.3.

A.1.3 Standard settings file

The format for settings files is presented below. The values are taken from the file `defaultsettings.set` which should be included with the library. When making a new settings file, care must be taken not to change the order of the arguments or add extra spaces. Settings are read starting on the first line, and consequently comments should be restricted to below the last argument. Even if comments are left out, please note the line break after the last argument.

```
L = 1.0;
radius = 0.003;
mass = 1.0;
eta = 0.1;
tau = 0.004;
u0 = 1.0;
dt = 0.0004;
t = 0.0;
t_max = 4.0;
beta = 0.0;
gamma = 1.0;
n0 = 100;
dim = 1;
pairwise = 0;
testparticle = 0;
timecorrelated = 1;
periodic = 1;
inertial = 0;
synthetic = 0;
% Simulation settings. Do not remove or change order of arguments.
% No extra spaces. Remember line break after last argument.
% Restrict comments to below last argument.
```

A.2 The MPC library

The MPC[1] library contains functions for checking collisions between particles, and for sorting the particles into boxes depending on their spatial coordinates.

A.2.1 Data types

```
struct box;
```

The contents of `struct box` are presented in table A.3.

Table A.3: Contents of `struct box`

name:	type:	description:
`np`	`int`	number of particles currently in box
`N`	`int`	`box` array is of size $\texttt{N}^{\texttt{dim}}$, should be chosen so that $\texttt{N}^{\texttt{dim}} \approx n_0$
`I, J`	`int`	coordinates of box in `box` array, $0 \leq \texttt{I}, \texttt{J} < \texttt{N}$
`residents`	`particle *`	pointer to first `particle` in a linked list of particles currently in the box, `NULL` for empty boxes

[1] Macroscopic Particle Collider

A.2.2 Functions

`box **init_boxes(int N, simulation *S);`

Allocates memory for a `box` array of size $\texttt{N}^{\texttt{dim}}$ and sets internal parameters of each element to appropriate values. Initialises `np` to 0 and `residents` to `NULL`. Returns a pointer to the initialised `box` array.

`void box_it(box *Box, particle *P);`

Inserts `P` as top element of the linked list of residing particles by letting `next` in `P` point to current top element, and then letting `residents` in `Box` point to `P`. Increments `np` in `Box`.

`void box_them(box **Boxes, simulation *S);`

Goes through all `n0` particles in `S` and calls `box_it()` for each particle with $\chi = 1$, after having calculated which `box` it should belong to based on size of array `Boxes` and parameters in `S`.

`void empty_box(box *Box);`

Empties `Box` by setting `np` $= 0$ and `residents` $=$ `NULL`.

`void empty_boxes(box **Boxes, simulation *S);`

Calls `empty_box()` for all $\texttt{N}^{\texttt{dim}}$ boxes in array `Boxes`.

`int collide_it(particle *P, box *Box, simulation *S);`

Checks collisions between `P` and the particles with $\chi = 1$ in `Box`. Unless `P` is contained in `Box` collisions are checked against all residing particles. If `P` is a resident, however, collisions are only checked with consecutive members in the linked list of residents. Returns number of collisions.

`int collide_box(box *Box, box **Boxes, simulation *S);`

Calls `collide_it()` for all particles with $\chi = 1$ in `Box`, checking collisions against members of `Box` and the neighbouring boxes in array `Boxes`. Note that only three out of eight (one quadrant) neighbouring boxes are checked. This is to avoid duplicate checks when traversing the `Boxes` array in `collide_them()`. Returns number of collisions.

`int collide_testparticle(box **Boxes, simulation *S);`

Checks boxes surrounding a test particle (index 0 in particle array) for particles colliding with it. Returns number of collisions.

`int collide_them(box **Boxes, simulation *S);`

Depending on the settings specified in `S` this functions checks all boxes in array `Boxes` for collisions and/or checks collisions with a test particle by calling `collide_box()` and `collide_testparticle()` respectively. Returns number of collisions.

```
int remove_overlaps(box **Boxes, simulation *S);
```

Used before starting a new simulation to make sure that no particles overlap in the initial time step. In particle pairs overlapping one particle is assigned a new random position and the check is restarted. Returns number of restarts.

A.3 The NRrand library

The NRand library contains functions for producing various types of random numbers. The methods used are taken from *Numerical Recipes* [20].

A.3.1 Functions

Except for `next_Grand()` all functions in this library are from [20] and were implemented originally by Björn Andersson in [12].

```
float nextFloat(void);
```

Returns a floating point number in the range [0, 1[.

```
int nextInt(int upper_limit);
```

Returns an integer number in the range [0, `upper_limit`[.

```
float nextGauss(void);
```

Returns a Gaussian deviate created using the Box-Muller method. The resulting number has mean $\mu = 0$ and standard deviation $\sigma = 1$.

```
double next_Grand(double mu, double sigma);
```

Gets Gaussian deviate from `nextGauss` and transforms it to give it mean $\mu =$ `mu` and standard deviation $\sigma =$ `sigma`. The resulting transformed deviate is returned.

```
long rand_seed(long seed);
```

Seeds the random number generators above with `rand_seed`. An appropriate seed is `time(NULL)`.

A.4 Example of implementation

The program below initialises a `simulation` and and array of `particle`s based on parameters from a settings-file called `defaultsettings`. The actual simulation consists of moving the particles about in the velocity field, updating it each time step. The program should be quite self explanatory. As in the program below, the first time that functions from the NRrand library are used will probably be when calling `init_particles()`. Please remember seeding the random number generator before this.

```
#include <stdio.h>
#include <time.h>
#include "NRrand.h"
```

```
#include "simulation.h"
#include "mpc.h"

int main(void)
{
    // It is more convenient to work with pointers
    // to both _simulation_s and _particle_s:
    simulation Sim, *S = &Sim;
    particle *P;
    box **Boxes;
    char filename[] = "defaultsettings.set";
    FILE *file;
    int collisions = 0;

    // Getting and setting settings:
    if(!(file = fopen(filename, "rb"))) {
        fprintf(stderr, "Could not read from file %s!\n", filename);
        exit(-1);
    };
    args_from_file(file, S);
    fclose(file);
    S->k_max = (int)ceil(2.0*S->L/S->eta);

    // Initialising simulation:
    rand_seed(time(NULL));
    init_sim(S->n0, S->k_max, S);
    P = init_particles(P, S);
    make_Psiconst(S);
    Boxes = init_boxes((int)ceil(sqrt(S->n0)), S);

    // Start simulation:
    new_ak(S);
    for(S->t = 0.0; S->t <= S->t_max; S->t += S->dt) {
        get_velocities(S);
        move_particles(S);
        box_them(Boxes, S);
        collisions = collide_them(Boxes, S);
        empty_boxes(Boxes, S);
        update_ak(S);
    };

    printf("Collisions: %d", collisions);
    return 0;
}
```

B

Mathematical details

B.1 Lyapunov exponents

In [4] a theoretical estimate of the Lyapunov exponent, valid in the limit of small Ku, is presented

$$\lambda_1 = -\gamma(1-\Gamma)\varepsilon^2. \tag{B.1}$$

Here $\varepsilon^2 = D_1/(m^2\gamma^3)$. The diffusion constant D_1 is defined by

$$D_i = \frac{1}{2}\int_{-\infty}^{\infty} dt \langle F_{i1}(t)\, F_{i1}(0)\rangle, \tag{B.2}$$

with $F_{ij}(t) = \frac{\partial f_i}{\partial r_j}$ and $\boldsymbol{f} = \gamma m \boldsymbol{u}$. The results presented in equation (5.4) and the more general (5.5), where a potential component of relative strength β has been included in the velocity field (section 3.4.2), are derived below.

As a first step, the force is inserted into the expression (B.2) for D_1. The time dependent part is separated and evaluated, yielding

$$D_1 = \frac{1}{2}\int_{-\infty}^{\infty} dt \langle F_{11}(t)\, F_{11}(0)\rangle = \frac{m^2\gamma^2}{2}\int_{-\infty}^{\infty} dt \langle \partial_x u_x(t)\, \partial_x u_x(0)\rangle = \\ \frac{m^2\gamma^2}{2}\langle \partial_x u_x(0)\, \partial_x u_x(0)\rangle \int_{-\infty}^{\infty} dt\, e^{-|t|/\tau} = m^2\gamma^2\tau\langle \partial_x u_x(0)\, \partial_x u_x(0)\rangle \tag{B.3}$$

When proceeding to calculate the correlation in B.3 above, no assumptions will be made regarding where $\boldsymbol{u}$ is evaluated, except that it is at two points $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ such that $\boldsymbol{r} = \boldsymbol{r}_1 - \boldsymbol{r}_2$ is well defined. When evaluating D_1, $\boldsymbol{r} = 0$ is implied.

Inserting the definitions of the velocity field from section 3.4, the bracketed part of (B.3) becomes

$$\langle \partial_x u_x(0)\, \partial_x u_x(0)\rangle = \frac{1}{1+\beta^2}\langle (\partial_{xy}\psi(\boldsymbol{r}_1)+\beta\partial_{xx}\varphi(\boldsymbol{r}_1))(\partial_{xy}\psi(\boldsymbol{r}_2)+\beta\partial_{xx}\varphi(\boldsymbol{r}_2))\rangle = \frac{1}{1+\beta^2}\left(\langle \partial_{xy}\psi(\boldsymbol{r}_1)\,\partial_{xy}\psi(\boldsymbol{r}_2)\rangle + \beta^2\langle \partial_{xx}\varphi(\boldsymbol{r}_1)\,\partial_{xx}\varphi(\boldsymbol{r}_2)\rangle\right) \quad \text{(B.4)}$$

In the last step the fact that ψ and φ are uncorrelated is used to kill off the mixed terms. The terms can now be evaluated separately. Inserting the definition of ψ, and the properties of the Fourier coefficients from section 3.4 yields

$$\begin{aligned}\langle \partial_{xy}\psi(\boldsymbol{r}_1)\,\partial_{xy}\psi(\boldsymbol{r}_2)\rangle &= \frac{u_0^2\eta^4\pi}{L^2}\sum_{\boldsymbol{k}} k_x^2k_y^2\exp\left(\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{r}-\frac{\eta^2k^2}{2}\right)\approx \\ &\frac{u_0^2\eta^4\pi}{L^2}\left(\frac{L}{2\pi}\right)^2\int_{\mathbb{R}^2}\mathrm{d}^2\boldsymbol{k}\,k_x^2k_y^2\exp\left(\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{r}-\frac{\eta^2k^2}{2}\right)= \\ &\frac{u_0^2\eta^4}{4\pi}\int_{-\infty}^{\infty}\mathrm{d}n_x\frac{2\pi}{L}\left(\frac{2\pi n_x}{L}\right)^2\exp\left(\frac{2\pi}{L}\mathrm{i}n_x x-\left(\frac{2\pi}{L}\right)^2\frac{\eta^2n_x^2}{2}\right)\cdot \\ &\int_{-\infty}^{\infty}\mathrm{d}n_y\frac{2\pi}{L}\left(\frac{2\pi n_y}{L}\right)^2\exp\left(\frac{2\pi}{L}\mathrm{i}n_y y-\left(\frac{2\pi}{L}\right)^2\frac{\eta^2n_y^2}{2}\right), \quad \text{(B.5)}\end{aligned}$$

where in replacing the sum with with an integral, multiplication with the density of states in k-space is needed, hence the factor $\left(\frac{L}{2\pi}\right)^2$.

The expression in (B.5) can be evaluated using a standard trick from mathematical physics. The first step is solving the integral

$$\begin{aligned}I_0(x) &= \int_{-\infty}^{\infty}\mathrm{d}n\frac{2\pi}{L}\exp\left(\frac{2\pi}{L}\mathrm{i}nx-\left(\frac{2\pi}{L}\right)^2\frac{\eta^2n^2}{2}\right)= \\ &\frac{2\pi}{L}\int_{-\infty}^{\infty}\mathrm{d}n\exp\left(\frac{4\pi^2\eta^2}{2L^2}\left(2\frac{L}{2\pi\eta^2}\mathrm{i}nx-n^2\right)\right)= \\ &\frac{2\pi}{L}\int_{-\infty}^{\infty}\mathrm{d}n\exp\left(-\frac{4\pi^2\eta^2}{2L^2}\left(n^2-2\frac{L\mathrm{i}x}{2\pi\eta^2}+\left(\frac{L\mathrm{i}x}{2\pi\eta^2}\right)^2\right)+\frac{4\pi^2\eta^2}{2L^2}\left(\frac{L\mathrm{i}x}{2\pi\eta^2}\right)^2\right)= \\ &\frac{2\pi}{L}\exp\left(\frac{4\pi^2\eta^2}{2L^2}\left(\frac{L\mathrm{i}x}{2\pi\eta^2}\right)^2\right)\int_{-\infty}^{\infty}\mathrm{d}n\exp\left(-\frac{4\pi^2\eta^2}{2L^2}\left(n-\frac{L\mathrm{i}x}{2\pi\eta^2}\right)^2\right)= \\ &\frac{2\pi}{L}\exp\left(-\frac{x^2}{2\eta^2}\right)\int_{-\infty}^{\infty}\mathrm{d}n\exp\left(-\frac{4\pi^2\eta^2n^2}{2L^2}\right)= \\ &\frac{2\pi}{L}\exp\left(-\frac{x^2}{2\eta^2}\right)\sqrt{2\pi}\frac{L}{2\pi\eta}=\frac{\sqrt{2\pi}}{\eta}\exp\left(-\frac{x^2}{2\eta^2}\right). \quad \text{(B.6)}\end{aligned}$$

The integral (B.5) can now be written in terms of derivatives of I_0, obtaining

$$\langle \partial_{xy}\psi(\boldsymbol{r}_1)\,\partial_{xy}\psi(\boldsymbol{r}_2)\rangle = \frac{u_0^2\eta^4}{4\pi}\left(-\partial_{xx}I_0(x)\right)\left(-\partial_{yy}I_0(y)\right) = \left\{-\partial_{xx}I_0(x) = \frac{\sqrt{2\pi}}{\eta^3}\left(1-\frac{x^2}{\eta^2}\right)\exp\left(-\frac{x^2}{2\eta^2}\right)\right\} = \frac{u_0^2}{2\eta^2}\left(1-\frac{x^2}{\eta^2}\right)\left(1-\frac{y^2}{\eta^2}\right)\exp\left(-\frac{r^2}{2\eta^2}\right). \quad \text{(B.7)}$$

In the same way the second component of (B.4) can be calculated. This yields

$$\langle \partial_{xx}\varphi(\boldsymbol{r}_1)\,\partial_{xx}\varphi(\boldsymbol{r}_2)\rangle = \frac{u_0^2\eta^4}{4\pi}\left(\partial_{x^4}I_0(x)\right)I_0(y) = \left\{\partial_{x^4}I_0(x) = \frac{\sqrt{2\pi}}{\eta^3}\left(\frac{3}{\eta^2}-\frac{6x^2}{\eta^4}-\frac{x^4}{\eta^6}\right)\exp\left(-\frac{x^2}{2\eta^2}\right)\right\} = \frac{u_0^2}{2\eta^2}\left(\frac{3}{\eta^2}-\frac{6x^2}{\eta^4}-\frac{x^4}{\eta^6}\right)\exp\left(-\frac{r^2}{2\eta^2}\right). \quad \text{(B.8)}$$

Using the results above, and letting $r = 0$, the equation (B.3) becomes

$$D_1 = \frac{m^2\gamma^2\tau}{1+\beta^2}\left(\langle \partial_{xy}\psi(\boldsymbol{r}_1)\,\partial_{xy}\psi(\boldsymbol{r}_2)\rangle + \beta^2\langle \partial_{xx}\varphi(\boldsymbol{r}_1)\,\partial_{xx}\varphi(\boldsymbol{r}_2)\rangle\right) = m^2\gamma^2\tau\frac{u_0^2}{2\eta^2}\frac{1+3\beta^2}{1+\beta^2} = \frac{m^2\gamma^2}{2\tau}\frac{u_0^2\tau^2}{\eta^2}\frac{1+3\beta^2}{1+\beta^2} = \frac{m^2\gamma^2}{2\tau}\mathrm{Ku}^2\frac{1+3\beta^2}{1+\beta^2} = \left\{\Gamma = \frac{\beta^2+3}{3\beta^2+1} \Leftrightarrow \beta^2 = \frac{3-\Gamma}{3\Gamma-1}\right\} = \frac{2}{\tau}\frac{m^2\gamma^2\mathrm{Ku}^2}{\Gamma+1} \quad \text{(B.9)}$$

Finally, inserting D_1 in (B.1) produces the theoretical expression for the Lyapunov exponent

$$\lambda_1 = -\gamma(1-\Gamma)\varepsilon^2 = (\Gamma-1)\frac{D_1}{m^2\gamma^2} = \frac{2}{\tau}\frac{\Gamma-1}{\Gamma+1}\mathrm{Ku}^2. \quad \text{(B.10)}$$

B.2 Answer

As a side result, it has been confirmed that the answer [21] is indeed

$$42. \tag{B.11}$$

Bibliography

[1] M. Wilkinson and B. Mehlig, *Caustics in turbulent aerosols*, Europhysics Letters **71**, 186–192 (July, 2005).

[2] K. Duncan, B. Mehlig, S. Östlund and M. Wilkinson, *Clustering by mixing flows*, Physical Review Letters **95**, 240602 (December, 2005).

[3] M. R. Maxey, *The gravitational settling of aerosol particles in homogeneous turbulence and random flow fields*, Journal of Fluid Mechanics **174**, 441–465 (January, 1987).

[4] M. Wilkinson, B. Mehlig, S. Östlund and K. P. Duncan, *Unmixing in random flows*, Physics of Fluids **19**, 113303 (November, 2007).

[5] D. J. Tritton, *Physical Fluid Dynamics*. Oxford University Press, Oxford, UK, 2nd ed., 1988.

[6] F. Mish, ed., *Merriam-Webster's Collegiate Dictionary*. Merriam-Webster, Springfield, USA, 11th ed., 2003. `Merriam-Webster.com`.

[7] B. Andersson, K. Gustavsson, B. Mehlig and M. Wilkinson, *Advective collisions*, Europhysics Letters **80**, 69001 (December, 2007).

[8] M. Wilkinson, B. Mehlig and V. Uski, *Stokes trapping and planet formation*, `arXiv:0706.3536v2`. Revised version.

[9] P. G. Saffman and J. S. Turner, *On the collision of drops in turbulent clouds*, Journal of Fluid Mechanics **1**, 16–30 (May, 1956).

[10] M. Wilkinson, B. Mehlig and V. Bezuglyy, *Caustic activation of rain showers*, Physical Review Letters **97**, 048501 (July, 2006).

[11] H. Sigurgeirsson and A. M. Stuart, *A model for preferential concentration*, Physics of Fluids **14**, 4352–4361 (November, 2002).

[12] B. Andersson, *Collision rate of particles advected in random flows*, Master's thesis, Chalmers University of Technology, Gothenburg, Sweden, March, 2007.

[13] A. Kelley and I. Pohl, *A Book on C*. Addison-Wesley, Boston, USA, 4th ed., 1998. Programming in C.

[14] M. T. Heath, *Scientific Computing*. McGraw-Hill, 2nd ed., 2002. International edition.

[15] L. Råde and B. Westergren, *Mathematics Handbook*. Studentlitteratur, Lund, Sweden, 4th ed., 1998. "Beta".

[16] P. Cvitanović, R. Artuso, R. Mainieri, G. Tanner and G. Vattay, *Chaos: Classical and Quantum*. Niels Bohr Institute, Copenhagen, Denmark, 2005. `ChaosBook.org`.

[17] G. Falkovich, K. Gawędzki and M. Vergassola, *Particles and fields in fluid turbulence*, Reviews of Modern Physics **73**, 913–975 (November, 2001).

[18] F. E. Udwadia and H. F. von Bremen, *On the efficient computation of Lyapunov exponents*, in *Control of Oscillations and Chaos*, vol. 3, pp. 484–487. St. Petersburg, Russia, August, 1997. 1st international conference.

[19] G. B. Folland, *Fourier analysis and its applications*. Brooks/Cole, Pacific Grove, USA, 1st ed., 1992.

[20] W. H. Press, S. A. Teulosky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 2nd ed., 1992. `NRBook.com`.

[21] D. Adams, *The Hitchhiker's Guide to the Galaxy*. Pan Books, London, UK, 1st ed., 1979.