Lecture given at the International Summer School Modern Computational Science
(August 16-28, 2009, Oldenburg, Germany)

A short ad hoc introduction to spectral methods for parabolic PDE and the Navier-Stokes equationsr

Hannes Uecker
Faculty of Mathematics and Science
Carl von Ossietzky Universität Oldenburg
D-26111 Oldenburg
Germany

Abstract. We provide and explain some simple self-contained `matlab` (`octave`) implementations of Fourier spectral solvers for nonlinear parabolic PDE and for the 2D Navier-Stokes equations in a periodic box, including a discussion of anti-aliasing and power spectra. We illustrate the solvers with various examples including some (weakly) turbulent flows.

Contents

1 Introduction

Ordinary differential equations (ODE) can rarely be solved analytically, and this holds even more for partial differential equations (PDE). Therefore, numerical approximations of solutions (or, somewhat inaccurately, numerical solutions) play a big role in applied science. In this lecture we first review very briefly basic facts about ODEs and their numerical solution in Sec. 2. In Sec. 3 we first consider one of the simplest PDE, namely the linear diffusion equation in one and two space dimensions, and introduce spectral methods for its numerical solution. These are then generalized to a number of semilinear parabolic equations, e.g., the Allen-Cahn equation, the Burgers equation, and the KS equation, where parabolic means that the linear part is smoothing, and then semilinear means that the nonlinearity contains less derivatives than the linear part. This then forms a starting point for the numerical solution of the Navier-Stokes equations in Sec. 4, which are the ultimate focus of this lecture.

Although there is a vast literature about the numerical solution of ODE and PDE, for instance [CHQZ88, HNW93, HW96, DB02], and a huge choice of highly developed software tools, here we only treat very basic methods and for didactic reasons set up some software by hand. For this we use `matlab/octave`. In particular the former comes with a number of highly developed ODE solvers, and some PDE packages, which however we do not use. [1]

After touching very briefly on finite difference method we only consider simple spectral methods, and for the Navier-Stokes equations focus on 2D problems, 2π-periodic in both directions. Moreover, we use the simplest possible linearly implicit nonlinearly explicit first order time-stepping, without stepsize control. The short `matlab/octave` scripts[2] can be downloaded from

`www.staff.uni-oldenburg.de/hannes.uecker/soft.html`

as well as – in due time – updates of this document and extensions of the scripts, for instance to 3D Navier-Stokes equations, which are straightforward but lengthy to explain. The programs are intended for easy understanding, modification and interactive exploration of flows. The latter is possible for Reynolds numbers up to a few hundreds on a laptop computer, and in Sec. 4.4 we give a small selection of examples, including some first steps towards turbulence.

We assume that the reader has a basic familiarity with `matlab`, or otherwise looks at some tutorial. In fact, our usage of `matlab` is very basic and in particular we do not aim at optimization (by, e.g., heavy vectorization) of code but rather at simplicity and easy readability. We do not assume that the reader has prior knowledge about PDE, although this is certainly helpful. We keep Moreover, we keep mathematics to a minimum and focus on the practical algorithmic side, and thus do not discuss, e.g., function spaces or existence of solutions to any of the PDEs considered.

[1]`octave` is a free software alternative to `matlab`, see `www.gnu.org/software/octave/` In this text we use `matlab` synonymous for both. The scripts have been tested on `matlab` 7.3.0 and `octave` 3.0. If available, `matlab` is somewhat more convenient w.r.t. plotting and appears to be somewhat faster.

[2]we maintain two versions due to tiny differences concerning the plotting

Finally, this text and the associated programs can be worked through from beginning to end, including a number of exercises which illustrate a number of famous nonlinear parabolic PDE, but the text is rather modular and hence there are other options. For instance, for readers with a basic knowledge of PDE who are only interested in spectral methods for the Navier-Stokes it should be sufficient to only look at Examples 3.5 and 3.6 and then directly go to Sec. 4.

2 Ordinary differential equations

An ordinary differential equation is an equation of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = f(u(t), t) \tag{1}$$

for an unknown function $u \in C^1(I, \mathbb{R}^d)$, where $I \subset \mathbb{R}$ is an interval, $f : \mathbb{R}^d \times I \to \mathbb{R}^d$ is called the vector field, and $C^1(I, \mathbb{R}^d)$ denotes the space of continuously differentiable functions from the interval I to $\mathbb{R}^d$. The vector field f is called autonomous if it does not depend explicitly on the time t, i.e., if $f = f(u(t))$. An initial value problem consists of (1) together with initial conditions (IC) $u_0 \in \mathbb{R}^d$ at some time t_0, i.e., $u|_{t=t_0} = u_0$. Instead of $\frac{\mathrm{d}}{\mathrm{d}t}u$ we sometimes write u'.

It is well-known that for f continuous in time and locally Lipschitz continuous in u we have the local existence and uniqueness of solutions, i.e., given u_0, t_0 there exists a $\delta = \delta(\|u_0\|_{\mathbb{R}^d}, t_0) > 0$ such that (1) has a unique solution $u \in C^1((t_0 - \delta, t_0 + \delta), \mathbb{R}^d)$ with $u(t_0) = u_0$. A function $f : \mathbb{R}^d \to \mathbb{R}^d$ is called locally Lipschitz continuous if for all C_1 there is a C_2 such that

$$\max\{\|u\|_{\mathbb{R}^d}, \|v\|_{\mathbb{R}^d}\} \le C_1 \quad \Rightarrow \quad \|f(u) - f(v)\|_{\mathbb{R}^d} \le C_2 \|u - v\|_{\mathbb{R}^d}.$$

If the local solution $u : (t_0 - \delta, t_0 + \delta) \to \mathbb{R}^d$ stays bounded, then it can be continued in time, and then often iteratively to all $t \in \mathbb{R}$ or at least all $t \ge t_0$, which is then called a global solution. However, the following two examples should remind the reader of basic problems with uniqueness and global existence.

Example 2.1 a) Solutions to ordinary examples can explode in finite time. The scalar equation $u' = 1 + u^2$ with initial condition $u(0) = 0$ has the unique solution $u(t) = \tan(t)$ which becomes unbounded for $t = \pi/2$.

b) The scalar equation $u' = \sqrt{|u|}$ with initial condition $u(0) = 0$ has infinitely many solutions. Two examples are $u(t) = 0$ and $u(t) = t^2/4$. ⌋

There are a number of special cases where solutions of (1) can be found explicitly. In other cases we are only interested in specific questions such as the long time behaviour of solutions: do solutions exist for all (positive) times? Do they converge to a stationary solution, or to a periodic solution? These questions are the starting point of so called dynamical system theory, and often (1) has enough structure to answer many of these questions in a qualitative sense.

However, to extract quantitative information one usually has to resort to numerical simulations. The basic idea is to choose some small time step h and replace the differential quotient $\frac{\mathrm{d}}{\mathrm{d}t}u(t)$ in (1) by some difference quotient involving $u(t), u(t+h)$ and possibly u at a number of additional discrete points. The easiest method is the *Euler method*, given by $\frac{\mathrm{d}}{\mathrm{d}t}u(t) \approx \frac{1}{h}(u(t+h)-u(t))$, hence, in case of an autonomous equation,

$$u(t+h) = u(t) + hf(u(t)). \tag{2}$$

Setting $u_m = u(t_m)$ with $t_m = mh$, this is often also written as $u_{m+1} = u_m + hf(u_m)$ and called *explicit* Euler method, since the solution at $t+h$ immediately follows from $u(t)$. In contrast, the so called *implicit* Euler method

$$\frac{1}{h}(u(t+h)-u(t)) = f(u(t+h)), \text{ i.e. } u_{m+1} = u_m + hf(u_{m+1}), \tag{3}$$

requires at each time step the solution of a (in general nonlinear) d-dimensional system for u_{m+1}. We will learn more about the significant differences between explicit and implicit methods in Sec. 3. As already said, there are various refinements of (2), for instance so called Runge-Kutta methods, Adams-Bashforth methods, and so on, and a well-developed theory concerning the pros and cons of the different methods as well as the convergence of the generated approximations to the (unique) solution of (1). For this we refer to the literature, e.g., [HNW93, HW96, DB02], see also the Lecture by *Burkard Kleihaus* at this Summer School. Here we only illustrate with one small example that different integration methods may perform differently, in particular over long time scales. This also gives the opportunity to introduce our first `matlab` script.

Example 2.2 Consider $u' = u$, $u(0) = u_0$. The solution is of course $u(t) = \mathrm{e}^t u_0$, but here we are interested in the comparison of the explicit (2) and the implicit (3) Euler methods. The explicit method becomes

$$u_{m+1} = u_m + hf(u_m) = (1+h)u_m,$$

and here we can also immediately solve the implicit method $u_{m+1} = u_m + hf(u_{m+1})$ for

$$u_{m+1} = \frac{1}{1-h}u_m.$$

This is very untypical and essentially only works for linear ODE (which generally do not require numerics anyway). Thus, it is only intended for illustration here. The two methods are implemented in the script `euler.m`,

```
% euler.m, solving u'=u, explicit vs implicit Euler
n=50; h=0.1; u=zeros(n,1);v=zeros(n,1);t=0:h:(n-1)*h;u(1)=1;v(1)=1;
for k=1:n-1; %  integration loop
    u(k+1)=u(k)+h*u(k); v(k+1)=v(k)/(1-h);
end
plot(t,exp(t),t,u,'-ko',t,v,'-k*');
```

This gives the plot in Fig.1. We see that both methods approximate the true solution (middle line) well for short times. Over larger times, the explicit/implicit method (circles/stars) underestimates/overestimates $u(t)$. This can be explained by the Taylor expansion of $\exp(h) = 1 + h + \frac{1}{2}h^2 + \frac{1}{6}h^3 + \ldots$. In each step we have

$$u(t+h) = \mathrm{e}^h u(t) = (1 + h + \frac{1}{2}h^2 + \ldots)u(t) \begin{cases} > (1+h)u(t) =: u_{\mathrm{ex}}(t+h) \\ < (1 + h + h^2 + h^3 + \ldots)u(t) =: u_{\mathrm{im}}(t+h) \end{cases}$$

where $(1 + h + h^2 + h^3 + \ldots)$ is the Taylor expansion of $1/(1-h)$. This also shows, that the *local* approximation error of both methods is of order h^2, but again we refer to the literature for details on this. For the sake of completeness we remark that `f=@(t,x) x; [t,x] = ode45(f,[0 5],1); clf;plot(t,exp(t),t,x,'-ks');` invokes one of the `matlab` built in ODE solvers (and emulated in `ode45.m` for `octave`) which produces the very accurate solution on the right of Fig. 1. But, as already said, for reasons of exposition here we shall only use self written code. ⌋

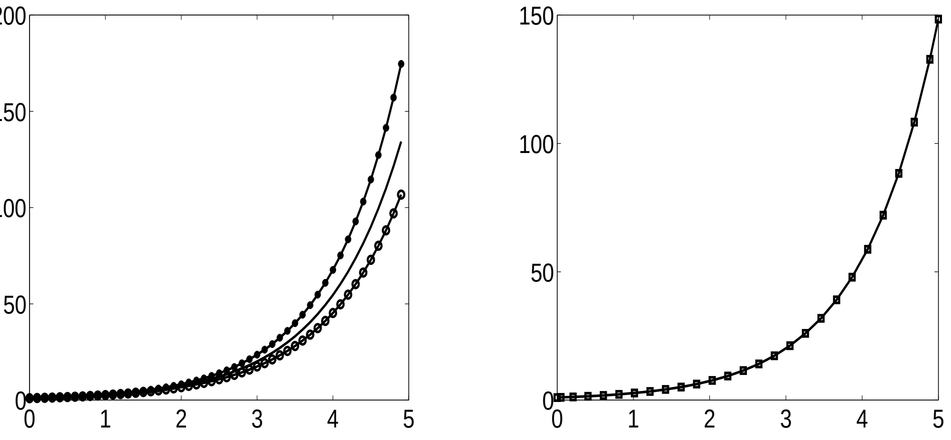

Figure 1: Left: `matlab` output from running `euler.m`; right: output of `ode45`.

3 Diffusion as a model problem

A very basic PDE is the linear heat equation

$$\partial_t u(x,t) = \Delta u(x,t), \quad x \in \Omega \subset \mathbb{R}^d,\ t \geq 0,\ \Delta = (\partial_{x_1}^2 + \partial_{x_2}^2 + \ldots + \partial_{x_d}^2), \tag{4}$$

also called linear diffusion equation. Here x denotes position, t time, and $u(x,t) \in \mathbb{R}$ can describe any quantity that may diffuse, for instance some ink in water, but specifically we may think of the temperature at some point x and time t. First we focus on the 1D case and w.l.o.g. (without loss of generality) take $\Omega = (0, 2\pi)$. Thus we consider

$$\partial_t u(x,t) = \partial_x^2 u(x,t). \tag{5}$$

The equation has to be supplemented by an

$$\text{initial condition } u(x,0) = u_0(x), \tag{6}$$

and boundary conditions which prescribe u (or, e.g., $\partial_x u$) at the boundary of the spatial domain and at all times $t > 0$. Physically reasonable boundary conditions are for instance Dirichlet boundary conditions $u(0,t) = a(t)$ and $u(2\pi, t) = b(t)$ with given functions a and b (prescribed temperatures at the boundary) or Neumann boundary conditions $\partial_x u(0,t) = c(t)$ and $\partial_x u(2\pi, t) = d(t)$ (prescribed heat fluxes at the boundary, for instance, insulation for $c = d \equiv 0$). However, for simplicity here we choose so called

$$\text{periodic boundary conditions, i.e. } u(0,t) = u(2\pi, t), \tag{7}$$

and similarly for x-derivatives of u, if defined. Although one could think of the heat distribution in a metal wire in ring shape, or, for instance, of diffusion of something (e.g., advantageous genes in population dynamics) around the shoreline of an island with no way across, these periodic boundary conditions often have little physical significance. However, mathematically they are very convenient for the solution of (5) by Fourier series.

3.1 First solutions

To get a feeling for the solution of (5), and also an idea of the numerical solution, we first consider some special solutions. For this we make an ansatz called separation of variables and write $u(x,t) = v(x)w(t)$. Plugging into (5) we obtain

$$w'(t)v(x) = v''(x)w(t) \quad \Leftrightarrow \quad \frac{w'(t)}{w(t)} = \frac{v''(x)}{v(x)},$$

where we assumed $w(t) \neq 0$ and $v(x) \neq 0$. Since the left hand side depends only on t but the right hand side only on x we obtain that both must be constant. It turns out that this constant must be non-positive, say equal to $-k^2 < 0$, where the square is for notational convenience. Thus

$$v''(x) = -k^2 v(x) \Rightarrow v(x) = c_1 \cos(kx) + c_2 \sin(kx)$$

with arbitrary $c_{1,2} \in \mathbb{R}$, and

$$w'(t) = -k^2 w(t) \Rightarrow w(t) = \mathrm{e}^{-k^2 t} w(0).$$

Moreover, to fulfill the boundary conditions $w(2\pi, t) = w(0, t)$ we need $k \in \mathbb{N}$. In summary, all so called Fourier modes

$$u(x,t) = \mathrm{e}^{-k^2 t}(c_1 \cos(kx) + c_2 \sin(kx)) \tag{8}$$

are solutions of (5), and since (5) is linear, any linear combination of these modes is again a solution. Notationally and computationally it is much more convenient to use complex notation and hence consider solutions of (5) in the form

$$u(x,t) = \sum_{k\in\mathbb{Z}} c_k \mathrm{e}^{-tk^2} \mathrm{e}^{\mathrm{i}kx}, \tag{9}$$

where, since u is a real-valued function, we must have $c_{-k} = \overline{c_k}$. Below we discuss which initial conditions can be represented by such linear combinations of $\mathrm{e}^{\mathrm{i}kx}$, but for now we note that, for $|k| \geq 1$, the Fourier modes decay exponentially in time. The decay is faster for higher $|k|$, i.e., faster oscillations (in x) decay more quickly, and therefore (5) is smoothing in time, which is what we expect diffusion to do. See Fig.2 for an illustration.

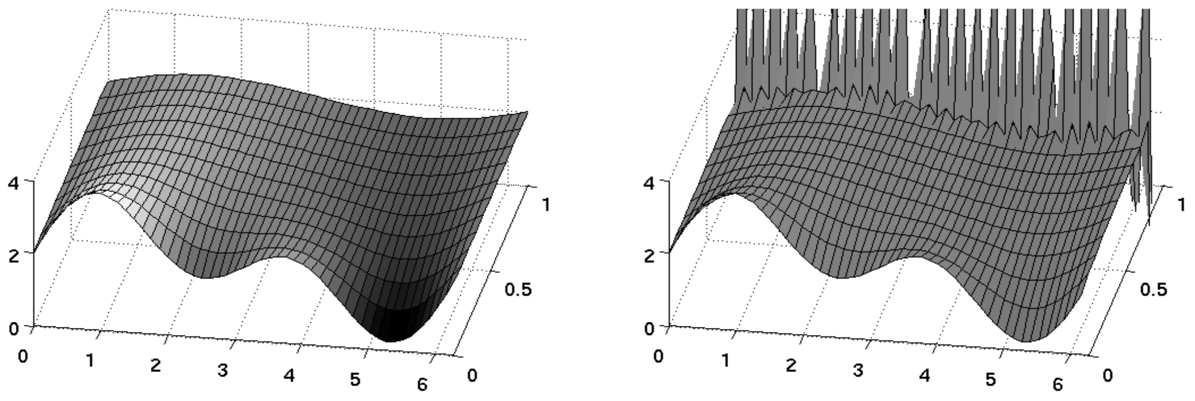

Figure 2: Left: the exact solution $u(x,t) = 2 + \mathrm{e}^{-t}\sin(x) + \mathrm{e}^{-4t}\sin(2x)$ of (5). Right: failure of numerical integration for the initial condition $u_0(x) = 2 + \sin(x) + \sin(2x)$ using an explicit scheme, see Example 3.3.

3.2 Numerics: basic idea, and a numerical failure

To approximate solutions of (5) we write $u^m(\cdot)$ for a desired approximation at a time $t_m = mh$. Thus, *the superscript m denotes the time-slice and not a power.* The basic idea clearly is to obtain u^{m+1} from u^m. A straightforward generalization of the (explicit) Euler method from Example 2.2 to (5) may run into serious trouble. We explain the problem by first choosing a so called finite difference discretization of the spatial derivatives.

Example 3.1 For $u \in C^3(0, 2\pi)$ we know that

$$\partial_x^2 u(x) \approx \frac{1}{\delta_x^2}\big[u(x+\delta_x) - 2u(x) + u(x-\delta_x)\big]$$

gives an approximation of order $\mathcal{O}(\delta_x^3)$ as $\delta_x \to 0$. Thus

$$\frac{1}{h}\big[u(x, t+h) - u(x,t)\big] \approx \frac{1}{\delta_x^2}\big[u(x+\delta_x) - 2u(x) + u(x-\delta_x)\big],$$

and given $u^m = (u^m(x_j))_{j=1,\dots,n}$ we may attempt to approximate u^{m+1} from u^m via

$$u^{m+1}(x_j) = u^m(x_j) + \frac{h}{\delta_x^2}\Big[u^m(x_{j+1}) - 2u^m(x_j) + u^m(x_{j-1})\Big], \tag{10}$$

where due to the periodic boundary conditions we put $x_{n+1} = x_1$, $x_0 = x_n$. To see what may go wrong with this scheme we consider an example, taken from [Str92, §8.2], where it is analyzed in some detail. Let $h = \delta_x^2$, for which (10) becomes

$$u^{m+1}(x_j) = u^m(x_{j+1}) - u^m(x_j) + u^m(x_{j-1}), \tag{11}$$

and for u^0 consider a function which is 1 at some x_j and 0 else, i.e.,

$$u^0 = (0, 0, \dots, 0, 1, 0, \dots, 0).$$

Using (11) to advance u^0 we obtain

$$\begin{array}{lccccccccc}
m=0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
m=1 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 \\
m=2 & 0 & 0 & 1 & -2 & 3 & -2 & 1 & 0 & 0 \\
m=3 & 0 & 1 & -3 & 6 & -7 & 6 & -3 & 1 & 0 \\
\vdots & & & & & & & & &
\end{array} \tag{12}$$

which is completely wrong (in the true solution the peak in the middle decays and widens). The oscillations in (12) are related to a phenomenon called instability (of the numerical method). It can be shown that solutions produced via (10) approximate solutions of (5) as $\delta_x \to 0$ provided that the CFL (Courant-Friedrichs-Lewy) condition $h/\delta_x^2 < 1/2$ holds. In practice, this often introduces a too severe restriction on the time step h. There are (well known) ways to modify (10) to relax this condition, for instance by treating diffusion implicitly, i.e.,

$$u^{m+1}(x_j) = u^m(x_j) + \frac{h}{\delta_x^2}\big[u^{m+1}(x_{j+1}) - 2u^{m+1}(x_j) + u^{m+1}(x_{j-1})\big]. \tag{13}$$

Then, however, in each time step we have to solve the (tridiagonal) linear system

$$\left[I - \frac{h}{\delta_x^2} A\right] \begin{pmatrix} u^{m+1}(x_1) \\ u^{m+1}(x_2) \\ \vdots \\ u^{m+1}(x_n) \end{pmatrix} = \begin{pmatrix} u^m(x_1) \\ u^m(x_2) \\ \vdots \\ u^m(x_n) \end{pmatrix}, \tag{14}$$

where, for periodic boundary conditions,

$$A = \begin{pmatrix} -2 & 1 & 0 & \cdots & 0 & 1 \\ 1 & -2 & 1 & 0 & \cdots & \\ 0 & \ddots & \ddots & \ddots & 0 & \cdots \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ 1 & 0 & \cdots & 0 & 1 & -2 \end{pmatrix}.$$

Schemes of type (14) are typically unconditionally stable, i.e., there is no restriction on the quotient h/δ_x^2. Moreover, there are efficient numerical methods for systems of type (14) and variants of it, see for instance Crank-Nicholson in the literature. Here, however, we shall consider implicit methods based on the Fourier modes (9). ⌋

3.3 Spectral methods

Methods which use a representation of u by orthogonal eigenfunctions of some linear (partial differential) operator A, and which thus are related to the spectrum of A, are called "spectral methods". The eigenfunctions of the operator $A = \Delta$ over the domain $\Omega = (0, 2\pi)^d$ with periodic boundary conditions are the Fourier modes $\mathrm{e}^{\mathrm{i}k\cdot x}$, $k \in \mathbb{Z}^d$, since

$$\Delta \mathrm{e}^{\mathrm{i}k\cdot x} = -|k|^2 \mathrm{e}^{\mathrm{i}k\cdot x}.$$

The representation

$$u(x) := \mathcal{F}^{-1}(\hat{u}_k)(x) := \sum_{k\in\mathbb{Z}^d} \hat{u}_k e^{\mathrm{i}k\cdot x}, \tag{15}$$

of a function u by these modes is called the Fourier series of u, and the Fourier coefficients are given by

$$\hat{u}_k := (\mathcal{F}u)_k := \frac{1}{(2\pi)^d} \int_{(0,2\pi)^d} u(x) \mathrm{e}^{-\mathrm{i}k\cdot x} \,\mathrm{d}x. \tag{16}$$

The question what functions u can be represented as (15) and in what sense is discussed in Appendix A. Here we note that if (15) holds, i.e., $u(x,t) = \sum_{k\in\mathbb{Z}^d} \hat{u}_k(t)\mathrm{e}^{\mathrm{i}k\cdot x}$, then

$$\partial_t u(x,t) = \sum_{k\in\mathbb{Z}^d} \partial_t \hat{u}_k(t)\mathrm{e}^{\mathrm{i}k\cdot x} = \Delta u(x,t) = \sum_{k\in\mathbb{Z}^d} -|k|^2\hat{u}_k(t)\mathrm{e}^{\mathrm{i}k\cdot x}, \tag{17}$$

and the linear PDE (5) becomes the infinite system of uncoupled linear ODEs

$$\partial_t \hat{u}_k = -|k|^2\hat{u}_k \quad \text{with solution} \quad \hat{u}_k(t) = \mathrm{e}^{-t|k|^2}\hat{u}_k(0), \quad k \in \mathbb{Z}^d. \tag{18}$$

Thus, for (5) we now have the following solution steps:

1) decompose (or analyze) $u_0(x) = \sum_{k\in\mathbb{Z}^d} \hat{u}_k(0)\mathrm{e}^{\mathrm{i}k\cdot x}$;
2) solve $\partial_t \hat{u}_k = -|k|^2\hat{u}_k$, i.e. $\hat{u}_k(t) = (\hat{S}(t)\hat{u}(0))_k = \mathrm{e}^{-t|k|^2}\hat{u}_k(0)$;
3) synthesize $u(x,t) = \sum_{k\in\mathbb{Z}^d} \hat{u}_k(t)\mathrm{e}^{\mathrm{i}k\cdot x}$.

If we denote the solution operator of (5) by $S(t)$, i.e., $u(x,t) = (S(t)u_0)(x)$, then we have $S(t) = \mathcal{F}^{-1}\hat{S}(t)\mathcal{F}$, i.e., the following diagram commutes.

$$\begin{tikzcd} u(\cdot,0) \arrow[r, "S(t)"] \arrow[d, "\mathcal{F}"'] & u(\cdot,t) \\ \hat{u}_\cdot(0) \arrow[r, "\hat{S}(t)"] & \hat{u}_\cdot(t) \arrow[u, "\mathcal{F}^{-1}"] \end{tikzcd}$$

In other words, $\mathcal{F}$ is a change of coordinates which diagonalizes ∂_x and hence also ∂_x^2 and $S(t)$. Again we remark that (18) shows that all Fourier modes except for $k=0$ decay exponentially in time. As a consequence, the solution $u(\cdot,t)$ becomes C^∞ (even analytic) for $t>0$.

To set up a numerical solver, we first remark that for reasons explained below we do not want to use the explicit solution $\hat{u}_k(t) = \mathrm{e}^{-t|k|^2}\hat{u}_k(0)$ in (18), but only the structure $\partial_t\hat{u}_k = -|k|^2\hat{u}_k$. The point is that now it is very simple to set up an implicit scheme: choose an implicity parameter $\eta \in [0,1]$ and a time step h, set $t_m = hm$ and $\hat{u}_k^m = \hat{u}_k(t_m)$ and write

$$\frac{1}{h}(\hat{u}_k^{m+1} - \hat{u}_k^m) \approx \partial_t\hat{u}_k(t_m) \approx -|k|^2\Big[(1-\eta)\hat{u}_k^m + \eta\hat{u}_k^{m+1}\Big].$$

Solving for $\hat{u}_k^{m+1}$ yields

$$\hat{u}_k^{m+1} = \mu_k\hat{u}_k^m \quad \text{with so called multipliers} \quad \mu_k = \frac{1-(1-\eta)h|k|^2}{1+h\eta|k|^2}. \tag{19}$$

The key observation is that $|\mu_k| \le 1$ for $\eta \ge 1/2$ such that the Fourier coefficients (and hence the solution $u(\cdot,t)$) can never grow in time, and moreover $0 < \mu_k \le 1$ for the fully implicit scheme with $\eta = 1$, which excludes oscillations in time of the Fourier coefficients.

Of course, to set up the numerics we have to truncate the Fourier series of u, i.e., approximate

$$u(x,t) = \sum_{|k|\leq n/2} \hat{u}_k(t)\mathrm{e}^{\mathrm{i}k\cdot x} \text{ with suitable } n,$$

and, moreover, need an efficient method to calculate (approximate) the Fourier coefficients $\hat{u}_k$ from u and vice versa. To approximate the $\hat{u}_k$ we use the so called discrete Fourier transform DFT, see Appendix A. The DFT can be evaluated by the so called Fast Fourier Transform FFT [CT65], which is one of the major computational achievements of the 1960ties, since it allows to compute the (approximate) Fourier transform of n sampling points in $\mathcal{O}(n \log n)$ time, in contrast to $\mathcal{O}(n^2)$ time using a naive approach.

3.3.1 1D FFT and 1D diffusion

In `matlab`, let $u = (u_1, \dots, u_n)$ be an n-vector, n even, to be read as a sampling of a (2π-periodic) function u on a domain of length 2π. Then the command `d=fft(u)` produces a (complex) n-vector $(d_1, \dots, d_n)$ which is related to the first n so called discrete Fourier coefficients $\tilde{u}_j$ of u via

$$\begin{array}{r} \big(d_1 \quad d_2 \quad d_3 \quad \dots \quad d_{n/2-1} \quad\quad d_{n/2} \quad\quad d_{n/2+1} \quad\quad d_{n/2+2} \quad\quad \dots \quad\quad d_n\big) \\ n\big(\tilde{u}_0 \quad \tilde{u}_1 \quad \tilde{u}_2 \quad \dots \quad \tilde{u}_{n/2-2} \quad \tilde{u}_{n/2-1} \quad \tilde{u}_{-n/2} \quad \tilde{u}_{-n/2+1} \quad \dots \quad \tilde{u}_{-1}\big) \end{array} \tag{20}$$

The $\tilde{u}_j$ approximate the first $n-1$ Fourier coefficients $\hat{u}_j$, $|j| < n/2 - 1$ of u, *if* the sampling rate of u is sufficiently high; otherwise we may have so called aliasing, see Appendix A and Example 3.5 below.

In the following we assume that n (i.e., the sampling rate) is sufficiently large and thus in (20) we identify $\hat{u}_k = \tilde{u}_k$. The main difficulty then is to always keep in mind the indexing: `d=fft(u)` starts with $\hat{u}_0$, then contains $n/2-1$ Fourier coefficients with $k > 0$, and then $n/2$ Fourier coefficients for $k < 0$. Consequently, swapping the left and right halves of `d` gives the FFT of u with zero frequency $\hat{u}_0$ at the "center" position $n/2+1$. In `matlab`, this can be achieved by `uhat=fftshift(d)`, see also Example 3.15 below. The inverse FFT is then given by $u = \mathcal{F}^{-1}$`(uhat)/n`, i.e. `u=ifft(d)`.

Remark 3.2 The Fourier coefficients $\hat{u}_k$ of a real (i.e. real-valued) function u satisfy

$$\hat{u}_{-k} = \overline{\hat{u}_k}. \tag{21}$$

In particular, one always has $\hat{u}_0 \in \mathbb{R}$. Moreover, the complex vector `d=fft(u)` contains a lot of redundant information. There are alternative real FFTs, but for simplicity we shall not use these. Finally, for n even, there is a slight imbalance in `d=fft(u)` since $\tilde{u}_{-n/2}$ appears but $\tilde{u}_{n/2}$ does not. However, `ifft` in `matlab` takes care of this. ⌋

Example 3.3 The script `h1d.m` solves (5) using FFT. Line 5 defines the space and time parameters. In lines 8 we store the multipliers μ_k in an array to later take advantage of array multiplication in `matlab` for increased speed. Note that `nv` by `fftshift` takes care of the indexing (20). Lines 9,10 prepare the data, and Lines 11-13 are the actual integration loop. However, note that the inverse FFT in line 12 is only needed for later plotting, and the solution is entirely advanced in Fourier variables.

```
1  % h1d.m solving u_t=u_xx by FFT and implicit (eta>0) time stepping
2  % Note:  if c_k is the k-th DFT-coeff then
3  %         fft(u)=(c_0 c_1...c_(n/2-1)  c_(-n/2)  c_(-n/2+1)...c_(-1))
4  % i.e.: position   1   2      n/2        n/2+1    n/2+2        n
5  n=50; dx=2*pi/n; x=0:dx:2*pi-dx; eta=0.5; h=0.1;
6  psteps=11; tmax=h*(psteps-1); % psteps=# time-slices to plot
7  nv=fftshift(-n/2:1:n/2-1); % wave numbers
8  mu=(1-h*(1-eta)*nv.^2)./(1+eta*nv.^2*h); % F-multiplier vector
9  u0=2+sin(x)+sin(2*x);  u=u0; uf=fft(u); % initial condition
10 t=0:h:h*(psteps-1); ua=zeros(psteps,n); ua(1,:)=u0; % for plotting
11 for i=2:psteps % integration loop
12   uf=mu.*uf; u=ifft(uf);ua(i,:)=u; % back to x only for plotting
13 end
14 [X,T]=meshgrid(x,t);surf(X,T,ua); % plot
```

Executing `h1d.m` with $\eta \geq 0.5$ produces an output which is indistinguishable from the exact solution on the left of Fig.2. On the other hand, similar to the explicit Euler method for finite differences, running `h1d.m` with $\eta = 0$ yields again (wrong!) fast growing oscillations as shown on the right of Fig. 2. ⌋

3.3.2 Some nonlinear examples

In Example 3.3 the only reason for going back to x-space by inverse FFT during time stepping is for plotting. For nonlinear problems, such as the Navier-Stokes equations, it is often useful – though not strictly necessary – to go back to x space to evaluate the nonlinearity. Moreover, the nonlinearity usually cannot be handled implicitly in an easy way. Therefore a so called semi-implicit scheme is used, where the linear part is treated implicitly and the nonlinearity explicitly. In particular, in the following examples we always treat the linear part fully implicit. *Thus we always set η from Example 3.3 equal to* 1 *(and hence drop it).* We remark that our treatment of these examples is extremely short and each of these famous equations has hundreds of research papers (and some entire books) devoted to it. The first nonlinear problem is a simple so called semi-linear variant of (5), namely the Allen-Cahn equation.

Example 3.4 The Allen-Cahn equation, discussed in most modern textbooks on nonlinear PDE, describes, for instance, coarsening in non-conservative phase separation. It is given by

$$\partial_t u = \nu \partial_x^2 u + u - u^3$$

where $u = u(x,t) \in \mathbb{R}$ is a so called order parameter field which describes the (two) phases of some medium: $u(x,t) \approx 1$ means that at position x and time t the medium is in phase 1, while $u(x,t) \approx -1$ corresponds to phase 2. Phase separation and coarsening means that starting from initial conditions u close to 0 (mixed phases) after a short time the solution develops a structure consisting of bulk regions of one of the two phases. The diffusion term $\nu\partial_x^2 u$ with small $\nu{>}0$ models some diffuse interfaces of width $\sqrt{\nu}$ between the regions.

We approximate

$$\frac{1}{h}(\hat{u}_k^{m+1} - \hat{u}_k^m) = -\nu k^2 \hat{u}_k^{m+1} + \hat{f}^m \quad \Leftrightarrow \quad \hat{u}_k^{m+1} = \mu_k(\hat{u}_k^m + h\hat{f}_k^m), \tag{22}$$

where $\mu_k = (1+\nu h k^2)^{-1}$ and $\hat{f}^m = \mathcal{F}(f(u^m))$ is the Fourier transform of the "nonlinearity" $f(u^m) = u^m - (u^m)^3$. Thus, $\hat{f}$ is evaluated in x-space and then transformed by FFT, and it appears with index m as it is treated explicitly. Alternatively to (22) one could use a split step scheme which uses the (available) exact solution

$$\hat{u}_k^{m+1} = \mathrm{e}^{-\nu|k|^2 h}\hat{u}_k^m \tag{23}$$

of the linear part, see Remark 3.9. However evolving the nonlinear part by Euler stepping introduces an $\mathcal{O}(h^2)$ error anyway, and we think that split-step schemes are conceptually slightly more involved than (22) and require a longer discussion.

Here is a `matlab` script which implements the scheme (22), and which we reproduce completely since it serves as a template for later scripts, for which we then only give the key changes.

```
 1 % ac.m solving u_t=u_xx+u-u^3 by semi-implicit time stepping via FFT
 2 n=32; dx=2*pi/n; x=0:dx:2*pi-dx; nu=0.05; h=0.1;
 3 ssteps=5; % ssteps=small steps, internal integration loop
 4 psteps=20;% psteps=plotting steps=number of time slices to plot
 5 tmax=h*(psteps-1)*ssteps;t=0;
 6 nv = fftshift(-n/2:1:n/2-1);mu=1./(1+nv.^2*h*nu); % mu holds F-multipliers
 7 % generate IC 'blockwise random'
 8 nb=4; u0=[]; for k=1:n/nb; u0=[u0 0.5*(rand(1,1)-0.5)*ones(1,nb)]; end
 9 tv=0:h*ssteps:h*(psteps-1)*ssteps;    % for plotting
10 [X,T]=meshgrid(x,tv); [N,T]=meshgrid(-n/2:n/2-1,tv);
11 ua=zeros(psteps,n); ua(1,:)=u0; u=u0; uh=fft(u0);% ua holds u   for plotting
12 va=zeros(psteps,n); va(1,:)=abs(fftshift(uh))/n; % va holds uhat for plotting
13 for i=2:psteps       % outer integration loop, with plotting
14     for j=1:ssteps   % inner integration loop, no plotting
15     fh=fft(u-u.*u.*u); uh=mu.*(uh+h*fh);u=ifft(uh); % integration step
16     end
17     ua(i,:)=real(u);va(i,:)=abs(fftshift(uh))/n;t=t+ssteps*h;
18 end
19 figure(1);surf(X,T,ua);figure(2);surf(N,T,va);  % plotting
```

We only remark that in the integration loop we use an inner loop (lines 14-16) since we are not interested in plotting the solution every step. This is controlled by the parameters `ssteps` and `psteps` in lines 3,4, and which will again be used in most further examples. Figure 3 shows an example output with small n. If, however one uses smaller ν and larger n then it can be seen that the dynamics of the equation take place on two different time scales. Initially, the nearly uniform solution develops on an $\mathcal{O}(1)$ time scale a fine structure with sharp gradients between the two phases. Then, on an $\mathcal{O}(e^{1/\nu})$ time scale the solution coarsens, see also Example 3.16. ⌋

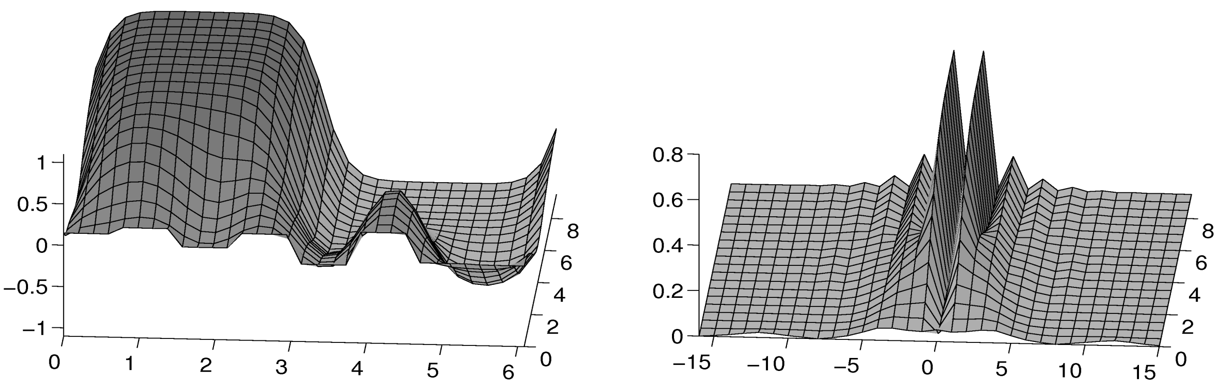

Figure 3: Example output of `ac.m`, u_0 is blockwise random, $\nu = 0.01$, $n = 32$, left $u(x,t)$, right $\hat{u}_k(t)$.

The next scalar nonlinear example is the Burgers equation, sometimes called a 1D model of the Navier-Stokes equations. However, before this we need to discuss aliasing by the nonlinearity.

Example 3.5 Band limits, aliasing and anti-aliasing. Given some function $f : [0, 2\pi] \to \mathbb{R}$, its sampling $f_j = f(x_j)$, $j = 1, \dots, n$ in general contains less information than f and thus we expect that any Fourier coefficients obtained from the sampling can only be approximations. Interestingly, f *can* be reconstructed *exactly* from $(f_j)_{j=1,\dots,n}$ if f is *band-limited*, i.e., if $\hat{f}_k = 0$ for $|k| > N$, and if the sampling frequency is sufficiently high. Otherwise we have so called aliasing, explained now by means of an example, see Appendix A for mathematical background.

We consider the simple example in Fig. 4, where again for reasons of exposition we use very small n. If we sample $\cos(3x)$ with $n = 8$ points then we get a somewhat faithful representation (dotted line). Moreover, the FFT $\tilde{u}$ of u_{dis} yields `0.0000 0.0000 0.0000 4.0000 -0.0000 4.0000 0.0000 0.0000` which is $n = 8$ times the *exact* $\hat{u}_k$ since u is band limited. However, if we choose $n = 4$ we rather believe the original function to be $\cos(x)$, and indeed FFT then gives `-0.0000 2.0000 0.0000 2.0000`. Thus, the original function $\cos(3x)$ appears under the alias $0.5\cos(x)$ if “undersampled”. The problem is: not only is the small scale structure ($\cos(3x)$) undetected (unresolved), but it falsely appears as a large scale $\cos(x)$.

Finally, due to $e^{ikx}e^{imx} = e^{i(k+m)x}$ nonlinearity is a problem for aliasing: if k is the highest wave-number in u, then u^2 contains modes at $2k$. Therefore, a solution to a nonlinear t-dependent PDE is never band limited, even if the initial condition is.

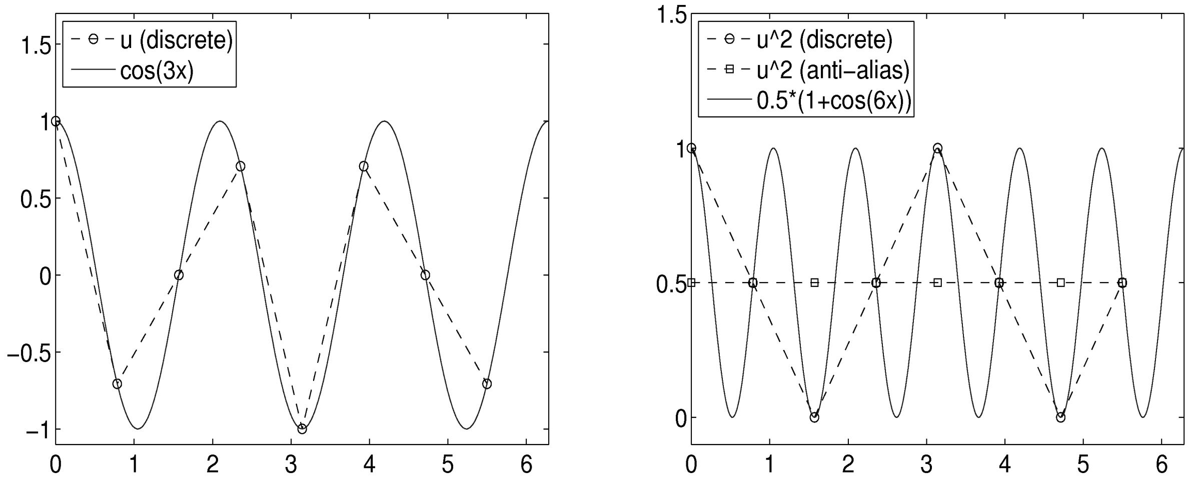

Figure 4: Nonlinearity aliasing and anti-aliasing as discussed in Example 3.5.

In Fig. 4 we have $u^2(x) = \frac{1}{2}(1 + \cos(6x))$ but $\cos(6x)$ cannot be sampled by 8 points and instead appears as $\cos(2x)$.

In the context of numerics for parabolic PDE a way to counter aliasing (and of course to improve the accuracy of approximation in general) is to make sure that high wavenumber Fourier coefficients, i.e., $|k|$ close to $n/2$ should be "well decayed". In practice, this means that n should be chosen sufficiently large, which obviously produces costs. However this aliasing of the nonlinearity can be avoided at little extra costs using the so called 3/2 rule. The idea is to pad $\hat{u}$, $\hat{v}$ by $m - n$ zeros with $m = 3n/2$ before transforming back to x-space, i.e., set

$$\hat{u}_p = (0, \dots, 0, \hat{u}_{-n/2}, \hat{u}_{-n/2+1}, \dots, \hat{u}_{n/2-1}, 0, \dots, 0)$$

and similar for $\hat{v}_p$. Then u_p =ifft($\hat{u}_p$) and v_p =ifft($\hat{v}_p$) will consist of m sampling points, and hence higher frequencies can be resolved in $u_p v_p$. It then remains to extract the pertinent n Fourier coefficients from fft($u_p v_p$). See [CHQZ88, §3.2] for the mathematics why this works and how the factor 3/2 arises, and for more sophisticated anti-aliasing methods. A `matlab` implementation of this anti-aliased product is given in `aap.m`, i.e.

```
function ph=aap(uh,vh) %anti-aliased product,
% in: uh,vh from fft with n samples, out: product in F-space
n=length(uh);m=n*3/2;
uhp=[uh(1:n/2) zeros(1,(m-n)) uh(n/2+1:n)]; % pad uhat with zeros
vhp=[vh(1:n/2) zeros(1,(m-n)) vh(n/2+1:n)]; % pad vhat with zeros
up=ifft(uhp); vp=ifft(vhp); w=up.*vp; wh=fft(w);
ph=1.5*[wh(1:n/2) wh(m-n/2+1:m)]; % extract F-coefficients
end
```

For the example in Fig.4 anti-aliasing yields u_p^2=0.5. This might not look like a good approximation of $\frac{1}{2}(1+\cos(6x))$, but at least the Fourier modes for $-4 \le k \le 3$ are correct, i.e., u_p^2 no longer contains the completely wrong contribution $0.5\cos(2x)$. ⌋

In the Allen-Cahn equation in Example 3.4 we did not use the anti-alias-product `aap` since we want to introduce things step by step, and because in the Allen-Cahn equation the nonlinearity does not contain derivatives and, therefore, if $\hat{u}_k$ is small for $|k|$ large, $\hat{f}_k$ is also small for k large, and nonlinearity-aliasing does not play a crucial role. However, if the nonlinearity contains derivatives, then high wave numbers get multiplied by factors involving k with large k, and therefore $\hat{f}_k$ may be large even if $\hat{u}_k$ decays quickly. As an example consider $f(u) = \partial_x(u^2)$ such that $\hat{f}_k = \mathrm{i}k\mathcal{F}(u^2)$ which appears in the Navier-Stokes equations in a similar way. We now illustrate this with the Burgers equation.

Example 3.6 The (viscous) Burgers equation

$$\partial_t u = \nu\partial_x^2 u - \frac{1}{2}\partial_x(u^2), \quad u = u(x,t) \in \mathbb{R},$$

where $\nu > 0$ is called viscosity, arises as a model in various settings, for instance traffic flow, see [BK00, Chapters 7 and 10] for a very readable account. The Burgers equation has remarkable structural properties, but here we exclusively focus on a numerical simulation. We approximate

$$\frac{1}{h}(\hat{u}_k^{n+1} - \hat{u}_k^n) = -\nu k^2 \hat{u}_k^{n+1} + \hat{f}_k^n \quad \Leftrightarrow \quad \hat{u}_k^{n+1} = \mu_k(\hat{u}_k^n + h\hat{f}_k^n), \tag{24}$$

where $\mu_k = (1+\nu h k^2)^{-1}$ and $\hat{f}_k = -\frac{\mathrm{i}}{2}k\mathcal{F}(u^2)_k$. The integration step from `burgers.m` thus reads `fh=-0.5*aap(uh,uh); uh=mu.*(uh+h*kv.*fh);` here `kv.*fh` implements the multiplication by $\mathrm{i}k$, also called spectral differentiation.

Of particular interest is the case of small ν, similar to large Reynolds number in the Navier-Stokes equations, see Sec. 4.4. To illustrate the significance of anti-aliasing in the Burgers equation, in Fig. 5 we use $\nu = 10^{-2}$ and $n = 32$ which can be shown to be a minimal necessary resolution. ⌋

Remark 3.7 Spectral methods which do not use anti-aliasing, or, more generally, which do not attempt to project all terms onto the (finite set of) basis functions, are often called pseudo-spectral methods. ⌋

Example 3.8 The Kuramoto-Sivashinsky equation

$$\partial_\tau v = -\partial_\xi^2 v - \partial_\xi^4 v - \frac{1}{2}\partial_\xi(v^2), \qquad \xi \in I \subset \mathbb{R}, \quad \tau \geq 0, \quad v(\xi,\tau) \in \mathbb{R} \tag{25}$$

can be considered as a variant of the Burgers equation with a long wave instability. It arises for instance in the description of instabilities of flame fronts in pipes, and in various other contexts with a long wave instability. The term $-\partial_\xi^2 v$ models anti-diffusion, which is arrested by the fourth order dissipative term $-\partial_\xi^4 v$. In passing we note that solving (25) with an explicit method as in Example 3.1 would require the extremely restrictive stability condition $h \sim \delta_x^4$ due to the term $-\partial_\xi^4 v$.

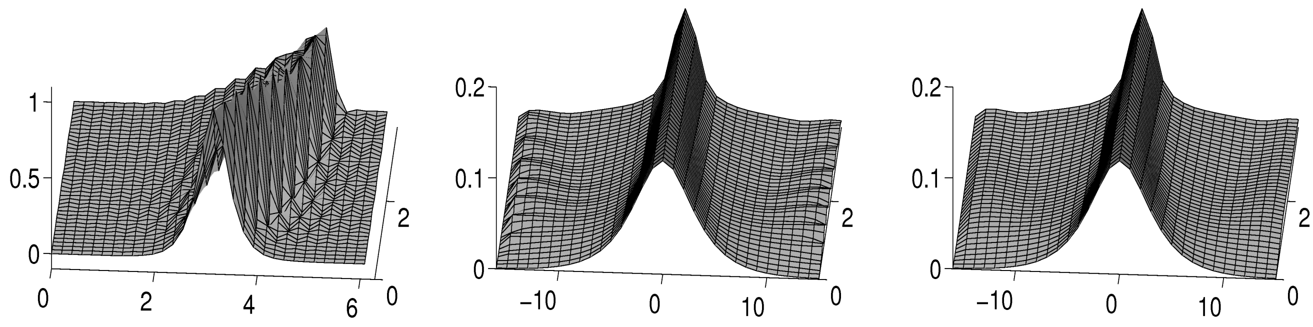

Figure 5: Example output of `burgers.m`, $u_0 = \mathrm{sech}(4x)$, $\nu = 0.01$, n=32. Left and middle: $u(x,t)$ and $|\hat{u}_k(t)|$ without anti-aliasing. Right: $|\hat{u}_k(t)|$ with anti-aliasing. With anti-aliasing, also the solution u in x space shows slightly less oscillations, which are unphysical, i.e. only due to the poor numerical resolution.

The Kuramoto-Sivashinsky equation is usually considered for $I = \mathbb{R}$ or I some *large* domain $[0, L]$ with periodic boundary conditions. Additionally to (25) being a famous and very interesting equation, here we consider (25) for two reasons: a) to explain how to rescale an equation to $x \in [0, 2\pi]$ such that FFT can directly be applied; b) to explain how the spectral properties are useful to get a first analytical understanding.

To rescale (25) we let $u(x,t) = v(Lx/(2\pi), Lt/(2\pi))$ to obtain

$$\partial_t u = -s\partial_x^2 u - s^3\partial_x^4 u - \frac{1}{2}\partial_x(u^2), \quad x \in [0, 2\pi], \tag{26}$$

where $s = 2\pi/L$ is a scale factor[3]. To illustrate the significance of s we consider the linearization $\partial_t u = -s\partial_x^2 u - s^3\partial_x^4 u$ of (25) around $u \equiv 0$. The solutions are $\mathrm{e}^{\mathrm{i}kx+\lambda_k t}$, $k \in \mathbb{Z}$ and $\lambda(k) = sk^2 - s^3k^4$, and in Fig. 6 we show the growth rates λ_k for $s = 1$ and $s = 1/20$, respectively. We see that for $s = 1$ there are three neutral modes $k = 0$ and $k = \pm 1$, and all other modes again decay exponentially since $\lambda_k < 0$ for $|k| \geq 2$. In contrast, for $s = 1/20$ we have *bands of unstable modes*, i.e., modes $\mathrm{e}^{\mathrm{i}kx}$ with $1 \leq |k| \leq 19$ linearly grow exponentially with maximal rate near $|k| = 10\sqrt{2} \approx 14$. These k correspond to long waves in the original formulation (25), hence the name long wave instability.

The (difficult) question then is, if and how the nonlinearity $-\frac{1}{2}\partial_x(u^2)$, for which it is unclear if it has a stabilising or destabilising influence,[4] can arrest this linear growth, see, e.g., [Tem97, Chapter III.4.1] for a detailed discussion. In any case, for $L > 2\pi$ large we may expect interesting dynamics. As usual, numerically we

[3]the rescaling of time is not strictly necessary; it is convenient in order to keep the factor $-\frac{1}{2}$ in front of the nonlinearity.

[4]In fact, it is both.

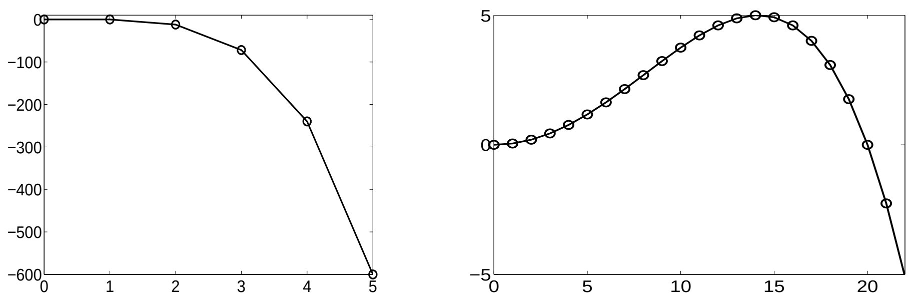

Figure 6: The spectrum of the linearization of the KS equation 26 for $s = 1$ (left) and $s = 1/20$ (right).

approximate

$$\hat{u}_k^{m+1} = \mu_k(\hat{u}_k^m + h\hat{f}_k^m), \quad \text{where now} \quad \mu_k = \frac{1}{1 + h(s^3k^4 - sk^2)}. \tag{27}$$

Except for (27) and different ways of plotting, the implementation of the initial value problem for (6) is identical to the one for the Burgers equation, see `ks.m`. For small L ($s \geq 1$), (26) is indeed rather boring (or more precisely: qualitatively similar to the Burgers equation). But Fig. 7a) shows the so called “chaos” (or “turbulence”) which is produced by (26) for $s = 1/20$, and b) illustrates this chaos in the behavior of the (low $|k|$) Fourier coefficients. Moreover, averaging $|\hat{u}_k(t)|$ over time, denoted by $\langle|\hat{u}_k|\rangle = \lim_{T\to\infty} \frac{1}{T}\int_0^T |\hat{u}_k(t)|\,\mathrm{d}t$, shows that $\langle|\hat{u}_k(t)|\rangle$ is maximal for $|k|$ close to 14 as expected from the linear instability analysis illustrated in Fig.6. Also note that $\hat{u}_0$ is exactly conserved. Finally we remark that to plot the original v as a solution of (25) we simply have to use the original time and space scales, e.g., use `waterfall(X/s,T/s,ua)` instead of `waterfall(X,T,ua)` for plotting. ⌋

Remark 3.9 An alternative to the very simple “linearly implicit nonlinearly explicit” one-step method (22) are split step methods, where one integration step (of length h) is split into two (or more) steps as follows. Assume that we want to integrate $\partial_t u = Au + f(u)$ where A is some linear (differential) operator, for instance $A = \partial_x^2$, and f some nonlinearity. Moreover, assume that we have one method M to advance the solution of $\partial_t u = Au$ by h, i.e., $u^{m+1} = Mu^m$ and a second method G to advance the solution of $\partial_t v = f(v)$ by h, i.e., $v^{m+1} = G(v^m)$. A method for $\partial_t u = Au + f(u)$ can then be set up as, for instance,

$$\tilde{u}^{m+1} = Mu^m, \quad u^{m+1} = \tilde{u}^{m+1} + G(u^m)$$

or some variant, taking e.g. $u^{m+1} = \tilde{u}^{m+1} + G(\tilde{u}^{m+1})$ or $u^{m+1} = \tilde{u}^{m+1} + \frac{1}{2}(G(u^m) + G(\tilde{u}^{m+1}))$ in the second step. Such split step methods often become necessary if the nonlinearity requires some special treatment, e.g., for so-called stiff nonlinearities, but in this lecture methods of type (22) will be sufficient, although we remark again that accuracy of solutions *can* be strongly improved if some problem adapted time stepping is used.

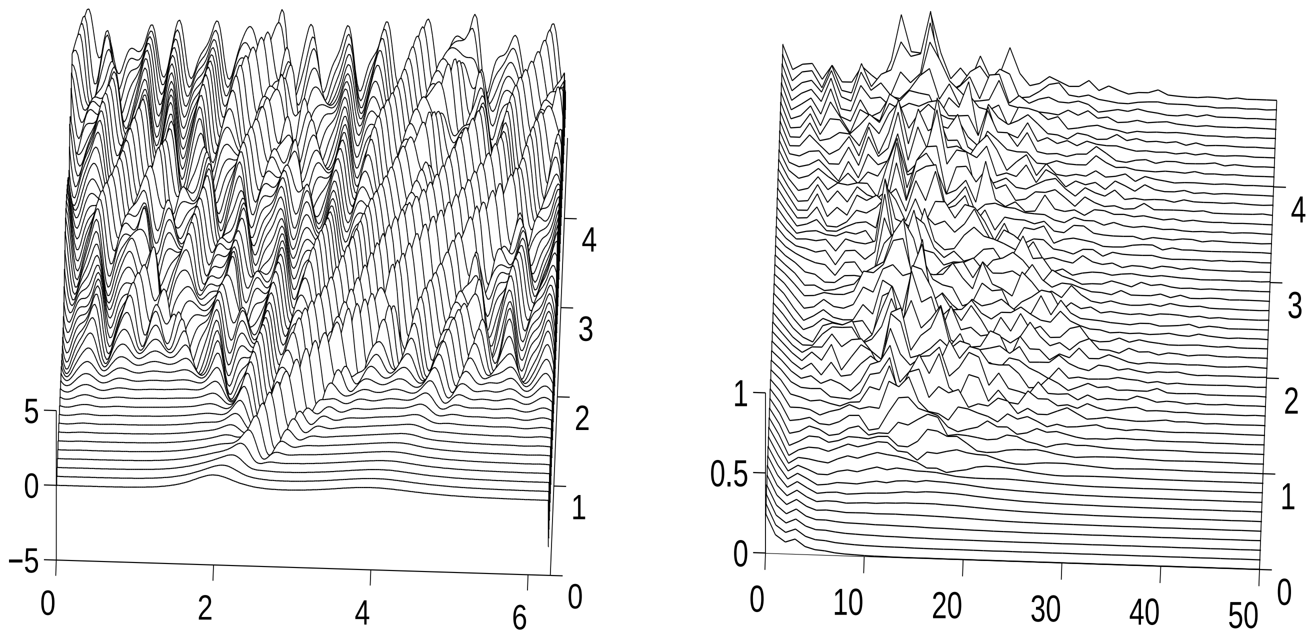

Figure 7: So called chaos in the KS equation over a large domain, $L = 40\pi$, hence $s = 1/20$ in (26); initial conditions $u_0(x) = \mathrm{sech}(4(x-2)) + \frac{1}{2}\mathrm{sech}(2(x-4))$. Left: the solution; right: the evolution of the $|\hat{u}_k|$.

In any case, comparing (22) with, e.g., the "linearly implicit nonlinearly explicit" split-step method

$$\hat{u}_k^{m+1} = \mu_k \hat{u}_k^m + h\hat{f}(u^m)_k \tag{28}$$

we note the following: for both methods the local approximation error is (formally) of order h^2, but while (22) has significant damping of high wave numbers of $\hat{f}$ due to multiplication of $h\hat{f}_k$ by μ_k, this is not the case for (28). The (over)damping of f in (22) is sometimes also called numerical viscosity. For "nice" parabolic (i.e., diffusion-like) problems this usually plays no important role since high wave numbers are strongly damped anyway, but it should be kept in mind if very accurate solutions are required, or if the underlying physics suggest that solutions do contain significant high wavenumber contributions as in turbulence simulations. $\rfloor$

The following series of exercises is intended to get some familiarity with `matlab` and the general idea (22), and at the same time to get to know some further interesting and famous equations. In particular, Exercises 3.12 to 3.14 study *systems* of (roughly) reaction diffusion equations and hence require a bit more programming effort. As a hint, you should get pictures as in Fig.8, or more fancy ones.

Exercise 3.10 In `ks.m`, implement and plot the averaging of $\hat{u}_k$. $\rfloor$

Exercise 3.11 a) Modify `ac.m` to study the Kolmogorov, Petrovsky, Piscounuv (KPP) equation $\partial_t u = \partial_x^2 u + u - u^2$, $x \in [0, L]$, which occurs as a model for various systems in nature, for instance for chemical reactions or population dynamics. Rescale to $x \in [0, 2\pi]$, take periodic boundary conditions and as intial condition a small localized perturbation of $u = 0$, and watch the evolution of fronts.
b) Modify `ks.m` (or ac.m) to study the Swift-Hohenberg equation $\partial_\tau v = -(1+\partial_\xi^2)^2 v + \alpha v - v^3$, $\xi \in [0, L]$, where $\alpha \in \mathbb{R}$. Take periodic boundary conditions, rescale to $x \in [0, 2\pi]$, take arbitrary initial conditions, and watch the difference between $\alpha < 0$ and $\alpha > 0$. ⌋

Exercise 3.12 As an example of a reaction diffusion *system* consider the Schnakenberg model [Sch79, Mur89]

$$\begin{aligned} \partial_t u &= \partial_x^2 u - u + u^2 v, \\ \partial_t v &= d\partial_x^2 v + b - u^2 v, \end{aligned} \tag{29}$$

where $u = u(x,t) \in \mathbb{R}$ and $v = v(x,t) \in \mathbb{R}$ denote the concentrations of two chemical species and $b > 0$ and the ratio $0 < d = d_{\mathcal{V}}/d_{\mathcal{U}}$ of the diffusion constants are parameters. This is a prototype for a reaction diffusion system with a Turing instability. Implement (29) for $x \in [0, L]$, with large L and periodic boundary conditions, and watch the pattern formation for, e.g., $(b, d) = (3, 60)$ and $(u_0, v_0) = (b + \text{sech}(x - L/2), 1/b + 0.01\text{sech}(x - L/2)$. ⌋

Exercise 3.13 The Gray Scott model [GS83] is a model for cubic autocatalysis of two species. After suitable nondimensionalization the system reads

$$\begin{aligned} \partial_t u &= d_1\partial_x^2 u - uv^2 + f(1-u), \\ \partial_t v &= d_2\partial_x^2 v + uv^2 - (f+k)v, \end{aligned} \tag{30}$$

with typically $0<d_2<d_1$ in applications, and where the two parameters f, k appear in the above way for modeling reasons. Implement (30) for $x \in [0, L]$, with large L, initial conditions $(u_0, v_0) = (1 - 0.5\text{sech}(0.1(x - 3L/5)), 0.3\text{sech}(0.1(x - 3L/5)))$, and periodic boundary conditions, and study fronts for (30) for $(d_1, d_2) = (1, 0.5)$, $(f, k) = (0.023, 0.05)$, and self replicating patterns for $(d_1, d_2)=(1, 0.5)$, $(f, k)=(0.038, 0.06)$. ⌋

Exercise 3.14 The Fitz-Hugh-Nagumo (FHN) system is used as a (cartoon) model for the transmission of a nerve impulses, see [BK00] for background. After suitable nondimensionalization it reads

$$\begin{aligned} \partial_t u &= \partial_x^2 u + f(u) - v, \\ \partial_t v &= d\partial_x^2 v + bu - cv, \end{aligned} \tag{31}$$

where $f(u) = u(1-u)(u-a)$ and where $a \in \mathbb{R}$, $b, c > 0$ and $d \geq 0$ are some parameters. Implement (31) for $x \in [0, L]$, with large L and periodic boundary conditions, Watch the emergence of pulses in the so called excitable regime, i.e., let $(a, b, c, d) = (1/4, 0.00075, 0.005, 0.1)$ and $(u_0, v_0)(x) = (\text{sech}(x - 3L/5), -\text{sech}(x - 3L/5)\sin(x - 3L/5)$. ⌋

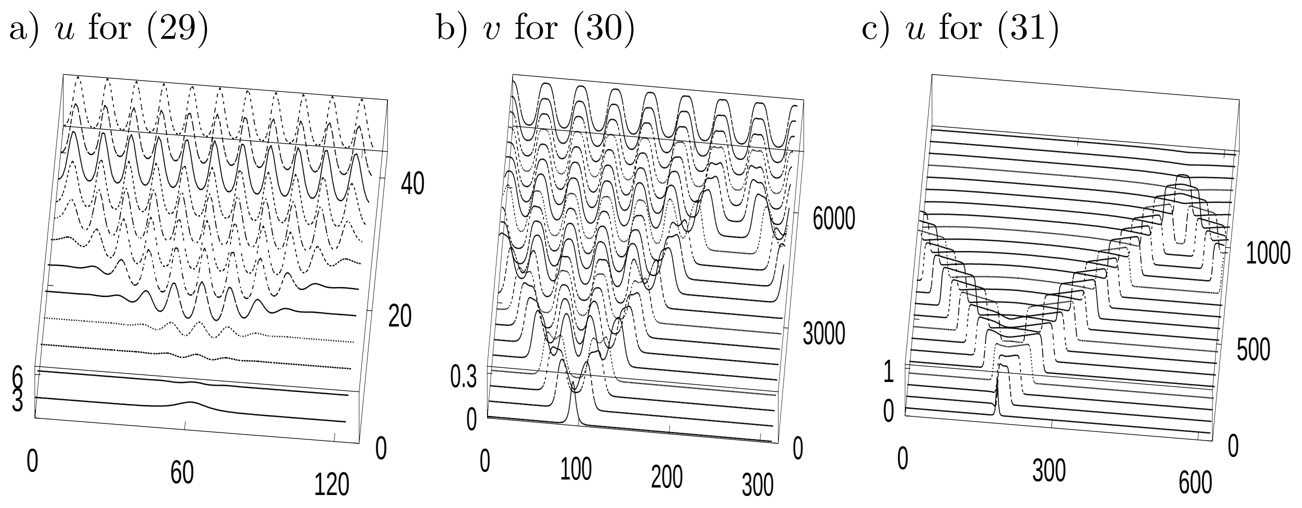

Figure 8: Example outputs for Exercises 3.12 to 3.14

3.3.3 2D FFT and 2D diffusion

All we need to extend the above method to higher space dimensions is a suitable FFT. In `matlab` this is provided by `fftn` and `ifftn`. Here we focus on $d = 2$. The 2D heat equation reads $\partial_t u = \nu \Delta u$, hence $\partial_t \hat{u}_k = -\nu(k_1^2 + k_2^2)\hat{u}_k = -|k|^2 \hat{u}_k$ gives the scheme

$$\hat{u}_k^{m+1} = \mu_k \hat{u}_k^m \quad \text{with multipliers} \quad \mu_k = \frac{1}{1 + \nu h |k|^2}. \tag{32}$$

Let u_{ij} be an $n \times n$ matrix, n even, to be considered as a sampling of a function u over $(x, y) \in [0, 2\pi]^2$, for instance obtained from the typical `matlab` commands
`n=6; dx=2*pi/n;dy=dx;x=-pi:dx:pi-dx;y=x; [X Y]=meshgrid(x,y); u=sin(X)+cos(Y);`
Thus

$$\texttt{u(i,j)} = u(x_j, y_i), \tag{33}$$

where it is important to keep in mind the "swapping of the indexing" due to `matlab`s "rows first" convention as in matrices. `fft2(u)` then is the $n \times n$ matrix (again with taking $\tilde{u}_k$ for $\hat{u}_k$)

$$\begin{pmatrix}
\hat{u}_{0,0} & \hat{u}_{0,1} & \cdots & \hat{u}_{0,n/2-1} & \hat{u}_{0,-n/2} & \cdots & \hat{u}_{0,-1} \\
\hat{u}_{1,0} & \hat{u}_{1,1} & \cdots & \hat{u}_{1,n/2-1} & \hat{u}_{1,-n/2} & \cdots & \hat{u}_{1,-1} \\
\vdots & \cdots & \ddots & & \cdots & \cdots & \\
\hat{u}_{n/2-1,0} & \hat{u}_{n/2-1,1} & \cdots & \hat{u}_{n/2-1,n/2-1} & \hat{u}_{n/2-1,-n/2} & \cdots & \hat{u}_{n/2-1,-1} \\
\hat{u}_{-n/2,0} & \hat{u}_{-n/2,1} & \cdots & \hat{u}_{-n/2,n/2-1} & \hat{u}_{-n/2,-n/2} & \cdots & \hat{u}_{-n/2,-1} \\
\vdots & \cdots & & & & \ddots & \vdots \\
\hat{u}_{-1,0} & \hat{u}_{-1,1} & \cdots & \hat{u}_{-1,n/2-1} & \hat{u}_{-1,-n/2} & \cdots & \hat{u}_{-1,-1}
\end{pmatrix}, \tag{34}$$

where again as a consequence of the `matlab` "matrix" indexing the wave numbers indices in (34) denote (k_2, k_1), k_1 corresponding to x and k_2 corresponding to y.

Example 3.15 `h2d.m` implements the scheme (32); here we only give the code snippet where the multiplier matrix `mu=`μ_k is defined, namely

```
1  %h2d.m, solving u_t=Delta u by FFT
2  n=32; dx=2*pi/n; dy=dx; x=-pi:dx:pi-dx; y=x; [X Y]=meshgrid(x,y);
3  nu=1; h=0.1; mu=zeros(n,n); % holds F-multipliers
4  for j1=1:n; % fill multiplier matrix mu as if fft(u) was centered
5      for j2=1:n;
6          k1=j2-n/2-1;k2=j1-n/2-1;mu(j1,j2)=1/(1+nu*(k1^2+k2^2)*h);
7      end
8  end
9  mu=fftshift(mu); % now adapt mu to true fft,
10 % then define IC and go to integration loop with uf=mu.*uf
```

In lines 4-8 the multipliers μ_k are again first stored corresponding to $\hat{u}$, and not corresponding to `fft2(u)`, see (34) for the different indexing. To adapt the matrix `mu` to the indexing (34) we then need to swap the upper left with the lower right quarter, and the lower left with the upper right. This is done by `fftshift` in line 9. Also note the swapping of indizes between `k1,k2` and `j1,j2` in line 6, which is due to (33). This is actually irrelevant for the problem considered here, namely *isotropic* diffusion with domain length 2π in both directions, but nevertheless should be kept in mind for generalization to anisotropic diffusion ($\partial_t u = d_1\partial_{x_1}^2 u + d_2\partial_{x_2}^2 u$ with $d_1 \neq d_2$) or boxes with aspect ratio different from 1 ($(x_1, x_2) \in [0, L_1] \times [0, L_2]$ with $L_1 \neq L_2$), which after rescaling to $[0, 2\pi]^2$ lead to anisotropic diffusion. The actual integration step is given by `uf=mu.*uf`. Call `h2d` to see the result. ⌋

Example 3.16 As a simple nonlinear generalization of Example 3.15 consider the 2D Allen-Cahn equation $\partial_t u = \nu\Delta u + u - u^3$. Using the multiplier matrix `mu` from above the integration step now becomes `uf=mu.*(uf+h*fft2(u-u.^3)); u=ifft2(uf);` see `ac2d.m`. Here we mainly comment on this to once more illustrate the interesting (transient) patterns during the phase separation, see Fig.9. See also Exercise 3.17 for the conservative phase separation in the Cahn-Hilliard equation. ⌋

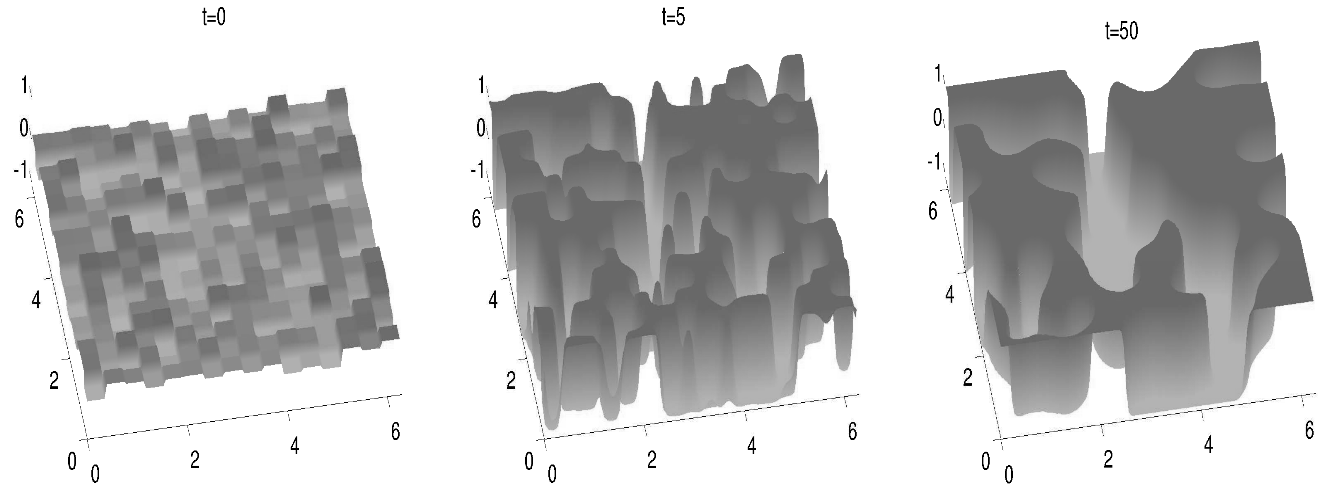

Figure 9: Phase separation in the 2D Allen-Cahn equation, $\nu = 0.001$, $n = 128$, random initial conditions near $u = 0$.

Exercise 3.17 a) Modify `ac2d.m` to study the 2D Swift-Hohenberg equation

$$\partial_t u = -(1-\Delta)^2 u + \alpha u - u^3. \tag{35}$$

b) Consider the Cahn-Hilliard equation

$$\partial_t u = -\Delta(\nu \Delta u + u - u^3), \tag{36}$$

which describes phase separation under conservation of the total phase $\displaystyle\int_\Omega u \,\mathrm{d}x$. Implement the scheme $\hat{u}^{m+1} = (1+h\nu|k|^4)^{-1}(\hat{u}^m + h|k|^2 \hat{f}_k^m)$ for (36). Thus obtain plots similar to Fig. 10. Also plot the $|\hat{u}_k(t)|$ and, as $t \to \infty$, compare this to the $|\hat{u}_k(t)|$ for the 2D Allen-Cahn equation. ⌋

Exercise 3.18 Study the models from Exercises 3.12-3.14 in 2D. For instance, for (30) in 2D there are self-replicating spots, see Fig. 11, which are fun to watch alive. ⌋

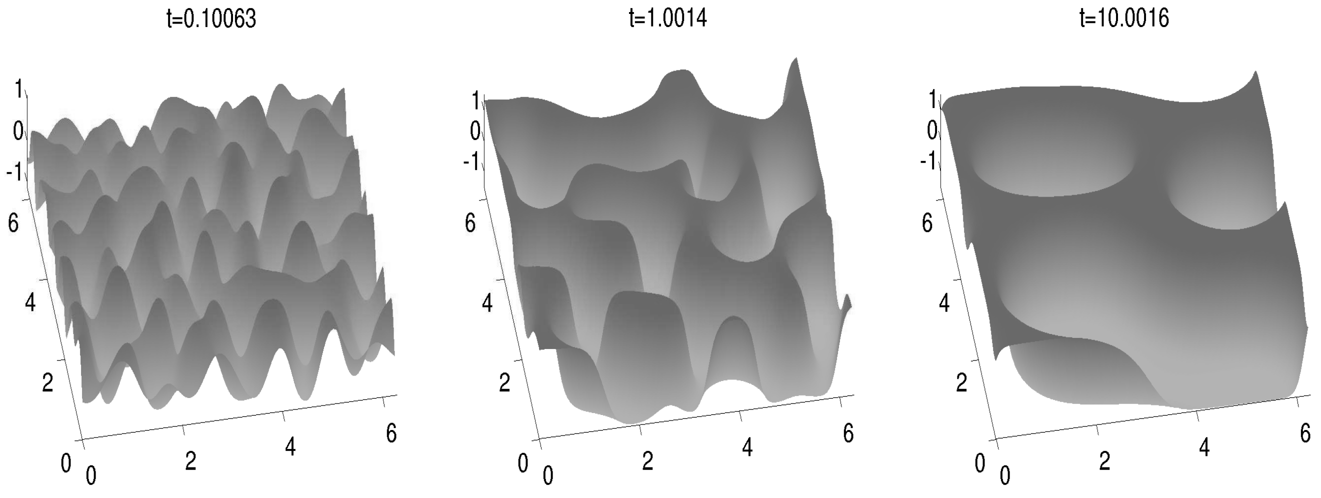

Figure 10: Phase separation in the 2D Cahn-Hilliard equation, $\nu = 0.01$, $n = 128$, random initial conditions near $u = 0$.

4 The Navier-Stokes equations

4.1 Introduction

The Navier-Stokes equations describe the velocity field of a fluid which is modeled as a continuum, i.e., we do not consider individual molecules of the fluid. Knowing the derivation of some model is useful to avoid applications which conflict with intrinsic limits of the theory. Thus, in Appendix B we recall the basic steps in the derivation, but, again, refer to the literature for more details.

The velocity at a position $x \in \mathbb{R}^d$, $d = 2$ or $d = 3$, at a time t is denoted with $u(x,t) \in \mathbb{R}^d$. The Navier-Stokes equations consist of a scalar equation for the conservation of mass and a d-dimensional system for the conservation of the momentum.

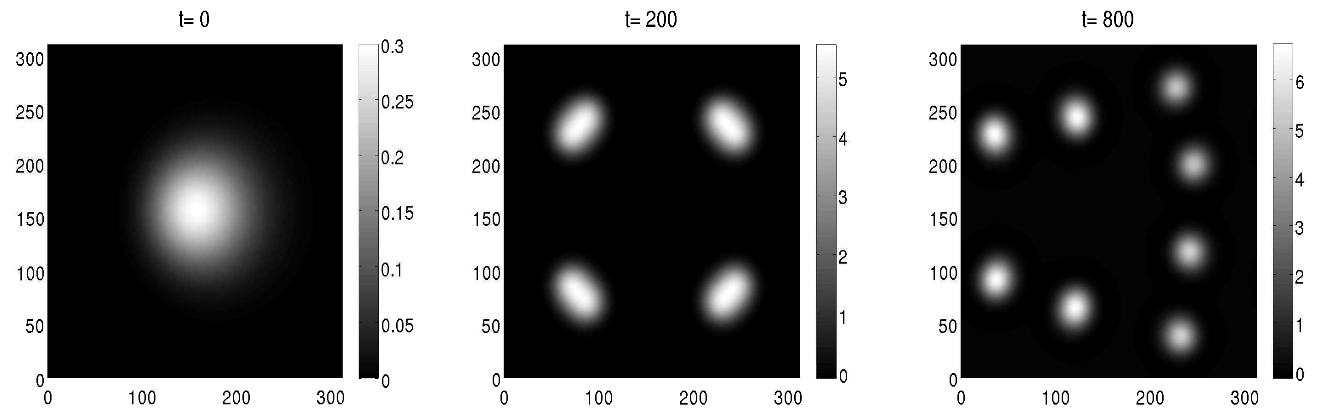

Figure 11: Self replicating spots in the 2D Gray-Scott model, v-component: $u_0(x,y) = 1 - \mathrm{e}^{-5(\tilde{x}^2+\tilde{y}^2)}, v_0(x,y) = 0.3\mathrm{e}^{-5(\tilde{x}^2+\tilde{y}^2)/(5+\tilde{x})}$, $(\tilde{x},\tilde{y}) = (x/50-\pi, y/50-\pi)$, $(d_1,d_2) = (0.1, 0.05)$, $(f,k) = (0.038, 0.06)$.

Here we focus on the so-called incompressible case, where the density of the fluid is constant in space and time. In dimensionless form the Navier-Stokes equations then become

$$\partial_t u + (u\cdot\nabla)u = -\nabla p + \frac{1}{\mathrm{R}}\Delta u + g, \qquad \nabla\cdot u = 0. \tag{37}$$

Here $\nabla p = (\partial_{x_1}p, \dots, \partial_{x_d}p)^T$ denotes the pressure gradient, $(u\cdot\nabla)u$ is called the convective nonlinearity and denotes the vector $\left(\sum_{j=1,\dots,d} u_j\partial_{x_j}u_1, \dots, \sum_{j=1,\dots,d} u_j\partial_{x_j}u_d\right)^T$, Δu is taken component wise, i.e., $\Delta u = \Big(\Delta u_1, \dots, \Delta u_d\Big)^T$, and $g = g(x,t) \in \mathbb{R}^d$ is an external volume force, e.g., gravity. Finally, the so-called Reynolds number $\mathrm{R} > 0$ measures the ratio between inertial and viscous forces. Large R roughly means fast flows or flows on a large scale, e.g. flows around a ship, while small R means slow flows, typically of very viscous fluids, for instance the motion of a drop of honey.

The equations (37) hold for $x \in \Omega \subset \mathbb{R}^d$ an open subset and for $t \geq 0$. Thus they have to be complemented by initial conditions and boundary conditions. For the latter we shall again content ourselves with periodic boundary conditions, which again have no physical meaning but allow to focus on the equations. Moreover, to treat (37) as an initial value problem there appears to be the problem that the second equation, $\nabla\cdot u$=0, is without time derivative, and that p appears without time derivative. This problem is solved by imposing the incompressibility $\nabla\cdot u$=0 as a condition on the function space for u; the term $-\nabla p$ in the first equation then has a natural interpretation as a projection P onto the divergence free vector fields, see Sec. 4.2.

Remark 4.1 In 2D, i.e., $d = 2$, for reasonable g the global existence and uniqueness of smooth solutions is well known, and, moreover, for $g \equiv 0$ all solutions of (37) decay to $u \equiv 0$, see, e.g., [Rob01] for a very readable exposition.

However, even for $g \equiv 0$, the global existence of smooth solutions of the 3D Navier-Stokes equations is one of the seven so called One Million Dollar or Millennium problems in mathematics presented by the Clay-Foundation in the year 2000. There are a number of reasons. On one hand, the Navier-Stokes equations describe the motion of fluids and the answer to this question would allow us to understand the world in a much better way. On the other hand, in mathematics the 3D Navier-Stokes equations are interesting partial differential equations, which so far resisted all attempts to prove the global existence of smooth solutions.

It is however a matter of debate whether answering the Millenium problem has consequences to applications. From a mathematical point of view, the convergence of numerical schemes is only guaranteed for smooth solutions. If the solutions become non-smooth in finite time then it is not clear that the numerical schemes describe reality. On the other hand, a proof of global existence of smooth solutions of the 3D Navier-Stokes equations which involve astronomically large bounds (on the order of, say, 10^{100}) will of course inhibit practical application, see, e.g. [HJ08, HJ07].

Classical books about local existence and uniqueness of the solutions of the Navier-Stokes- and Euler-equations are [Tem01, vW85]. A modern treatment of these equations as dynamical system and the existence theory of semilinear parabolic equations can be found for instance in [Hen81, Rob01]. Background on the derivation and applications of the Navier-Stokes equations and related equations can also be found in [Fow97]. Finally, turbulence in the Navier-Stokes equations is discussed from a mathematical point of view in [Cho94, DG95, FRMT01]. $\rfloor$

4.2 The linearized homogeneous equations

To understand the impact of the imcompressibility condition $\nabla \cdot u = 0$ and of the pressure gradient ∇p we first solve the linearized equations with $g = 0$, which is also the first step towards the construction of a numerical solver.

4.2.1 Explicit solution in Fourier space

To solve the linear system $\partial_t u = -\nabla p + \frac{1}{\mathrm{R}}\Delta u$, $\nabla \cdot u = 0$, for notational simplicity we first restrict ourselves to the two-dimensional case, i.e. $x \in \mathbb{R}^2$ and for our purposes w.l.o.g. to $\mathrm{R} = 1$. Then $u = (u_1, u_2)$ satisfies

$$\begin{aligned} \partial_t u_1 &= \partial_{x_1}^2 u_1 + \partial_{x_2}^2 u_1 - \partial_{x_1} p, \quad \partial_t u_2 = \partial_{x_1}^2 u_2 + \partial_{x_2}^2 u_2 - \partial_{x_2} p, \\ 0 &= \partial_{x_1} u_1 + \partial_{x_2} u_2. \end{aligned}$$

For periodic boundary conditions the solutions can again be written as Fourier series, i.e.

$$u_1(x,t) = \sum_{k\in\mathbb{Z}^2} \hat{u}_{1,k}(t) e^{ik\cdot x}, \quad u_2(x,t) = \sum_{k\in\mathbb{Z}^2} \hat{u}_{2,k}(t) e^{ik\cdot x}, \quad p(x,t) = \sum_{k\in\mathbb{Z}^2} \hat{p}_k(t) e^{ik\cdot x},$$

with $\hat{u}_{1,k}$, $\hat{u}_{2,k}$ and $\hat{p}_k$ complex-valued coefficients only depending on time, satisfying

$$\begin{aligned}\partial_t \hat{u}_{1,k} &= -(k_1^2+k_2^2)\hat{u}_{1,k} - ik_1\hat{p}_k, \quad \partial_t \hat{u}_{2,k} = -(k_1^2+k_2^2)\hat{u}_{2,k} - ik_2\hat{p}_k,\\ 0 &= ik_1\hat{u}_{1,k} + ik_2\hat{u}_{2,k}.\end{aligned}$$

This is a system of ordinary differential equations on the linear subspace of $\mathbb{C}^3$ defined by the third equation, and hence can be solved explicitly. We set

$$\begin{pmatrix}\hat{u}_{1,k}\\ \hat{u}_{2,k}\\ \hat{p}_k\end{pmatrix} = \begin{pmatrix}c_1\\ c_2\\ c_3\end{pmatrix} e^{\lambda t}$$

and look for a λ, such that

$$\det\begin{pmatrix}-\lambda-|k|^2 & 0 & -ik_1\\ 0 & -\lambda-|k|^2 & -ik_2\\ ik_1 & ik_2 & 0\end{pmatrix} = 0$$

with $|k|^2 = k_1^2+k_2^2$. For $|k|\neq 0$ the determinant vanishes for $\lambda = -|k|^2$ and then

$$\begin{pmatrix}c_1\\ c_2\\ c_3\end{pmatrix} = \frac{c_k}{|k|}\begin{pmatrix}-k_2\\ k_1\\ 0\end{pmatrix},$$

with arbitrary $c_k \in \mathbb{C}$ and where the normalization by $|k|$ turns out to be convenient later. Hence, the general solution u of the linearized equations is then given by

$$u(x,t{=}\sum_{k\in\mathbb{Z}^2}\frac{c_k}{|k|}e^{-t|k|^2}\begin{pmatrix}-k_2\\ k_1\end{pmatrix}\mathrm{e}^{\mathrm{i}k\cdot x}{=}\sum_{k\in\mathbb{Z}^2}\hat{u}_k(t)\mathrm{e}^{\mathrm{i}k\cdot x}, \quad \hat{u}_k(t){=}\frac{c_k}{|k|}e^{-t|k|^2}\begin{pmatrix}-k_2\\ k_1\end{pmatrix}.$$

Since we consider a real-valued problem we must have $\hat{u}_{-k}(t) = \overline{\hat{u}_k(t)}$ for $k \neq 0$, i.e. $c_{-k} = -\overline{c_k}$. The subspace $k=0$ will be considered later on in more detail. Note that in this calculus the pressure disappeared completely. However, it will play a role when considering the nonlinear problem, thus we next need to deal with it.

4.2.2 Dealing with the pressure gradient as a projection

In x-space we have, using integration by parts,

$$\int_\Omega u\cdot\nabla p\,\mathrm{d}x = -\int_\Omega p\nabla\cdot u\,\mathrm{d}x = 0,$$

where the boundary terms vanish for periodic boundary conditions, i.e. ∇p is L^2 orthogonal on the space of divergence free vector fields $\{u \in L^2 \ : \ \nabla \cdot u = 0\}$. In k space we have

$$\hat{u}_k \in U_k = \{\hat{u}_k \in \mathbb{C}^2 \ : \ k \cdot \hat{u}_k = 0\},$$

due to $\nabla \cdot u = 0$. The pressure gradient ∇p defines in $\mathbb{C}^2$ a vector $\mathrm{i}\hat{p}_k k$ with $\hat{p}_k \in \mathbb{C}$, orthogonal to U_k. Therefore, ∇p can be interpreted as orthogonal projection onto this subspace.

For the inhomogeneous problem or the nonlinear problem we need to calculate this projection P. Thus we consider

$$u + \nabla p = f, \quad \nabla \cdot u = 0, \tag{38}$$

in order to map a function f with $\nabla \cdot f \neq 0$ via the pressure gradient ∇p to a function u with $\nabla \cdot u = 0$. Again we restrict ourselves to the case $x \in \mathbb{R}^2$ with 2π-periodic boundary conditions. As above we make the ansatz

$$u_j(x) = \sum_{k \in \mathbb{Z}^2} \hat{u}_{j,k} e^{\mathrm{i}k \cdot x}, \quad p(x) = \sum_{k \in \mathbb{Z}^2} \hat{p}_k e^{\mathrm{i}k \cdot x}, \quad f_j(x) = \sum_{k \in \mathbb{Z}^2} \hat{f}_{j,k} e^{\mathrm{i} \cdot x},$$

with $\hat{u}_{j,k}$, $\hat{u}_{j,k}$, $\hat{f}_{j,k}$, $j = 1, 2$, and $\hat{p}_k$ complex-valued coefficients. Plugging into (38) yields

$$\hat{f}_{1,k} = \hat{u}_{1,k} + ik_1 \hat{p}_k, \quad \hat{f}_{2,k} = \hat{u}_{2,k} + ik_2 \hat{p}_k, \quad 0 = ik_1 \hat{u}_{1,k} + ik_2 \hat{u}_{2,k}. \tag{39}$$

In the case $|k| \neq 0$, we find the solution

$$\begin{pmatrix} \hat{u}_{1,k} \\ \hat{u}_{2,k} \\ \hat{p}_k \end{pmatrix} = \frac{1}{k_1^2 + k_2^2} \begin{pmatrix} k_2^2 & -k_1 k_2 & -ik_1 \\ -k_1 k_2 & k_1^2 & -ik_2 \\ -ik_1 & -ik_2 & 1 \end{pmatrix} \begin{pmatrix} \hat{f}_{1,k} \\ \hat{f}_{2,k} \\ 0 \end{pmatrix}.$$

For the subspace $k_1 = k_2 = 0$ there are two possibilities.
i) We prescribe the periodicity of the pressure p. Then $\hat{u}_{j,0} = \hat{f}_{j,0}$ in (39), which means that for $\hat{f}_{j,0} \neq 0$ we have a mean flow (or rather mean force) in direction x_j, and $\hat{p}_0$ is arbitrary, which is fine, since only ∇p plays a role.
ii) We require that the mean flows (forces) $\int_{[0,2\pi]^2} u_j(x_1, x_2) dx$, $j = 1, 2$, vanish, i.e. $\hat{u}_{1,0} = \hat{u}_{2,0} = 0$, and set

$$p(x,t) = \sum_{j=1}^{2} \alpha_j x_j + \tilde{p}(x,t),$$

where $\tilde{p}(x,t)$ is 2π periodic in the x_j. Then $\partial_{x_j} p(x,t) = \alpha_j + \partial_{x_j} \tilde{p}(x,t)$ and so

$$\hat{u}_{j,0} + \alpha_j = \hat{f}_{j,0}.$$

Thus, to a $\hat{f}_{j,0}$ we always find an α_j, so that $\hat{u}_{j,0} = 0$.

To illustrate the difference between i) and ii), in Fig. 12 we consider the example

$$f(x,y) = \mathrm{e}^{-2(x-\pi)^2-2(y-\pi)^2} \begin{pmatrix} 2+\tanh(y-\pi) \\ 0 \end{pmatrix}, \tag{40}$$

which corresponds to a slightly asymmetric volume force kicking to the right in the middle of the domain.

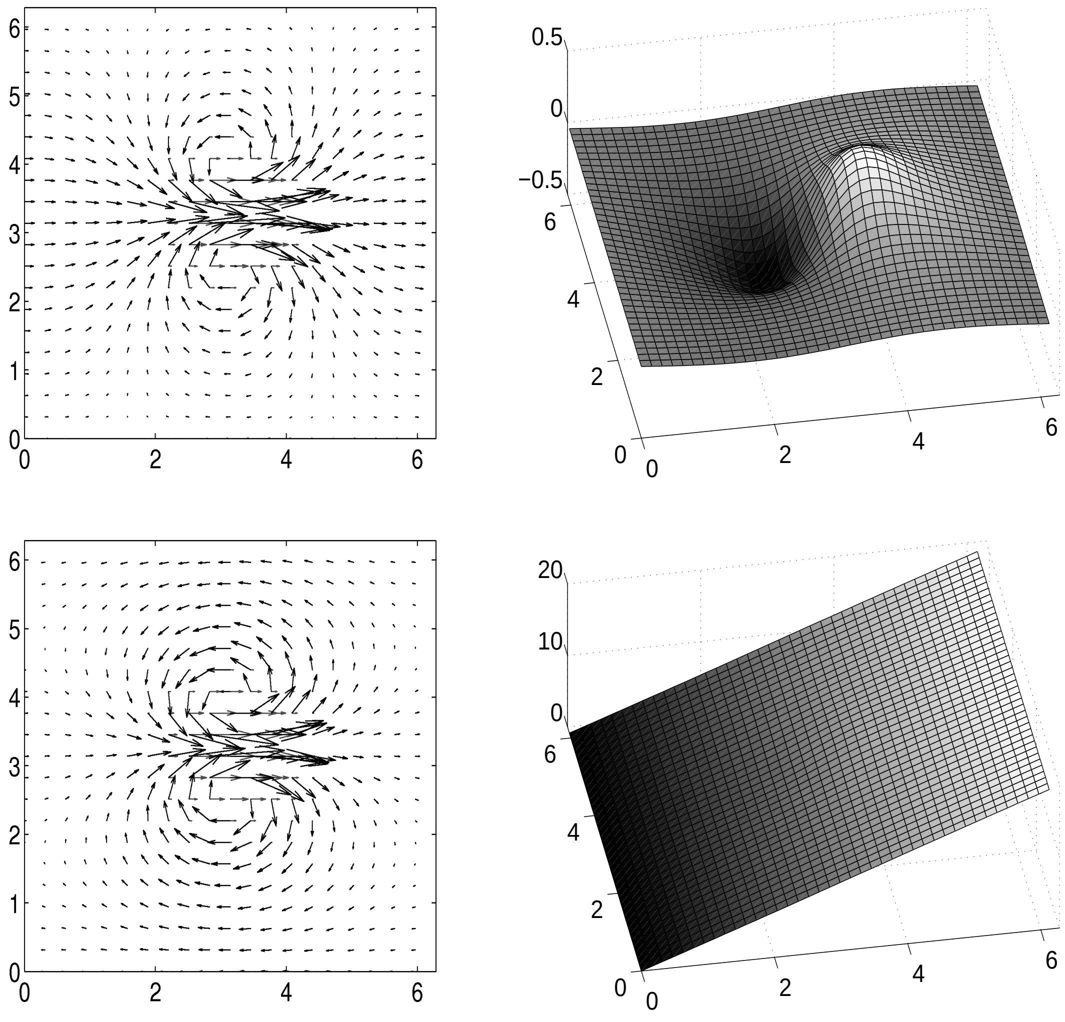

Figure 12: Illustration of the difference between cases i) and ii) concerning the boundary conditions for the pressure in the projection $u = Pf$, f from (40). Left: f and Pf, right: p, where in the top row we require a periodic pressure, giving a mean flow in Pf, while in the bottom row we require zero mean flow, giving a linear growth of the pressure.

Thus, choosing between i) and ii) is a question of modeling. i) has the disadvantage that a (constant) mean force leads to unbounded growth of (laminar) mean flows. Therefore, in the following we opt for ii), i.e. $\hat{u}_{j,0} = 0$ for $j = 1, 2$ and define the projection $\hat{P}$ as direct sum of the projections $\hat{P}_k$, i.e. $\hat{u}_k = (\hat{P}\hat{f})_k = \hat{P}_k \hat{f}_k$,

$$\hat{u}_0 = 0 \text{ and } \begin{pmatrix} \hat{u}_{1,k} \\ \hat{u}_{2,k} \end{pmatrix} = \frac{1}{k_1^2 + k_2^2} \begin{pmatrix} k_2^2 & -k_1 k_2 \\ -k_1 k_2 & k_1^2 \end{pmatrix} \begin{pmatrix} \hat{f}_{1,k} \\ \hat{f}_{2,k} \end{pmatrix} \text{ for } k \neq 0. \tag{41}$$

In x-space we define P by $P = \mathcal{F}^{-1}\hat{P}\mathcal{F}$.

Remark 4.2 The operator P is also called Stokes projector. In the general case of some arbitrary domain Ω one proceeds as follows to solve (38) and hence define P. Taking the divergence of (38) yields the Poisson equation

$$\Delta p = \nabla \cdot f, \tag{42}$$

and then $u = f - \nabla p$. The difference between i) and ii) becomes the question of appropriate boundary conditions for p in (42). Typically one uses homogeneous Neumann boundary conditions $\partial_n p = 0$ if u is prescribed on the boundary, for instance if $u|_{\partial\Omega} = 0$. Since $\Delta \hat{u}_k \mathrm{e}^{\mathrm{i}k\cdot x} = -|k|^2 \hat{u}_k \mathrm{e}^{\mathrm{i}k\cdot x}$, in a periodic box we have that P and Δ commute. In general (i.e., general domains) this is not true; then the operator $P\Delta$ is called the Stokes operator. ⌋

4.3 The numerical scheme

In Fourier space, the Navier-Stokes equations now read

$$\partial_t \hat{u}_k = -\frac{1}{\mathrm{R}}|k|^2 \hat{u}_k + \hat{P}_k \hat{f}_k \tag{43}$$

where we use the incompressibility $\nabla \cdot u = 0$ to rewrite the nonlinearity in an algorithmically friendly way as

$$(u \cdot \nabla)u = ((u_1^2)_x + (u_1 u_2)_y, (u_1 u_2)_x + (u_2^2)_y),$$

which in Fourier space becomes

$$\hat{f}_k = \mathcal{F}((-u \cdot \nabla)u + g)_k = \begin{pmatrix} -\mathrm{i}k_1 \mathcal{F}(u_1^2)_k - \mathrm{i}k_2 \mathcal{F}(u_1 u_2)_k + \hat{g}_{1,k} \\ -\mathrm{i}k_1 \mathcal{F}(u_1 u_2)_k - \mathrm{i}k_2 \mathcal{F}(u_2^2)_k + \hat{g}_{2,k} \end{pmatrix}. \tag{44}$$

Applying our usual linearly implicit nonlinearly explicit scheme we obtain

$$\frac{1}{h}(\hat{u}_k^{m+1} - \hat{u}_k^m) = -\frac{1}{\mathrm{R}}|k|^2 \hat{u}_k^{m+1} + \hat{P}_k \hat{f}_k^m, \tag{45}$$

i.e.

$$\hat{u}_k^{m+1} = \mu_k \big[\hat{u}_k^m + h\hat{P}_k \hat{f}_k^m\big] \quad \text{with} \quad \mu_k = \frac{1}{1 + h|k|^2/\mathrm{R}}. \tag{46}$$

To turn (46) into a program is now rather a question of organization, visualization and study of interesting flows, than of mathematical or algorithmic difficulty. As already said, the full program, including a number of interesting right hand sides g and various comments can be downloaded, see page 3. Here we only briefly discuss the main building blocks and then visualize some flows, including some further discussion of, e.g., the power spectrum and ideas related to turbulence.

The implementation Different from the `matlab` files so far we break the code into a script file `ns2d.m` and some function files `makemult.m, proj2d1.m, proj2d2.m, aap2.m, ns2dstep.m`.

Using a script `ns2d.m`, which contains the preparations like declaration of variables and the integration loop has the advantage that the variables can be inspected (e.g., replotted) from the command line, while using function files gives more structure to the code. Although the net `matlab` code is less than 60 lines, the files above are longer due to a number of options and comments, and therefore we do not print them completely but rather comment on the crucial parts.

For convenience we declare some matrices in `ns2d.m` as global, i.e.,

```
global av bv mm w1 w2 w3 w4 p1 p2 p3 p4 p5 g1h g2h;
```

where

`X,Y`	are the meshgrids,
`av,bv,mm`	are the vectors $(\mathrm{i}k_1) = \mathrm{i}(0\ 1\ \dots\ n/2-1\ \ -n/2\ \ -n/2+1\ \dots -1)$ and $(\mathrm{i}k_2) = \mathrm{i}(0\ 1\ \dots\ n/2-1\ -n/2\ -n/2+1\ \dots\ -1)^T$ used in the differentiation of the nonlinearity in (44), and the multiplier matrix (μ_{k_1,k_2}), set up for centered FFT and corrected by `fftshift` (see comments in `makemult.m`),
`p1,...,p5`	are the multiplier matrices for the projection step, (see `proj2d`),
`w1,...,w4`	are workspace matrices used to hold the nonlinearities (see `ns2dstep.m`),
`g1h,g2h`	hold the forcing $(\hat{g}_1, \hat{g}_2)$, which can be used for speed-up in the case of stationary forcing.

In `g.m` we define the forcing (g_1, g_2), for instance the "kick in the middle" (see Fig. 12)

```
Xt=X-pi; Yt=Y-pi;dec=exp(-4*Xt.^2-4*Yt.^2); g1=dec.*(2+tanh(Yt)); g2=0.*Y;
```

Next, in `ns2d.m` there are options to plot g, its projection Pg and the associated p (which were used to generate Fig. 12). The projection is done in `proj2d1` and `proj2d2`, resp. The difference is that the former does not return or calculate p, but the latter does.

As initial data we take $u \equiv 0$ and then go into the main integration loop, where the actual integration is done by `ns2dstep` which implements (46). There are three auxiliary functions, namely `red2d` which "resamples" a matrix (i.e., a meshgrid) for nicer `quiver` plots, `vof(u1,u2)` which calculates the vorticity

$$\omega = \partial_{x_2} u_1 - \partial_{x_1} u_2 = \mathcal{F}^{-1}(\mathrm{i}k_2 \hat{u}_1 - \mathrm{i}k_1 \hat{u}_2), \tag{47}$$

which is a good mathematical and visual diagnostic of the plot, and hence plotted in the integration loop, and `aap2` which returns an anti-aliased product. For this and for the structure of the integration loop and its various plot options see the comments in the respective files.

4.4 Let it flow

4.4.1 A stationary kick

The first flow we consider belongs to the stationary forcing (40), slightly changed, i.e.

$$g(x,y) = \mathrm{e}^{-4(\tilde{x}^2+\tilde{y}^2)} \begin{pmatrix} 2+\tanh\tilde{y} \\ 0 \end{pmatrix} \quad \text{where } (\tilde{x},\tilde{y}) = (x-\pi, y-\pi). \tag{48}$$

This might be considered as an (asymmetric) waterwheel which kicks the fluid locally in the middle of the flow domain. The asymmetry, given by $2+\tanh(y)$, is useful since without it the flow would be symmetric around $y=0$ and hence less interesting.

Starting from zero initial conditions, for small R the flow converges to a stationary, i.e. time independent one: the fluid flows to the right in the middle of the domain, and, due to the vanishing mean flow, see ii) on page 27, to the left at the top and the bottom. The left panel of Fig. 13 shows the flow field. As already said, a good

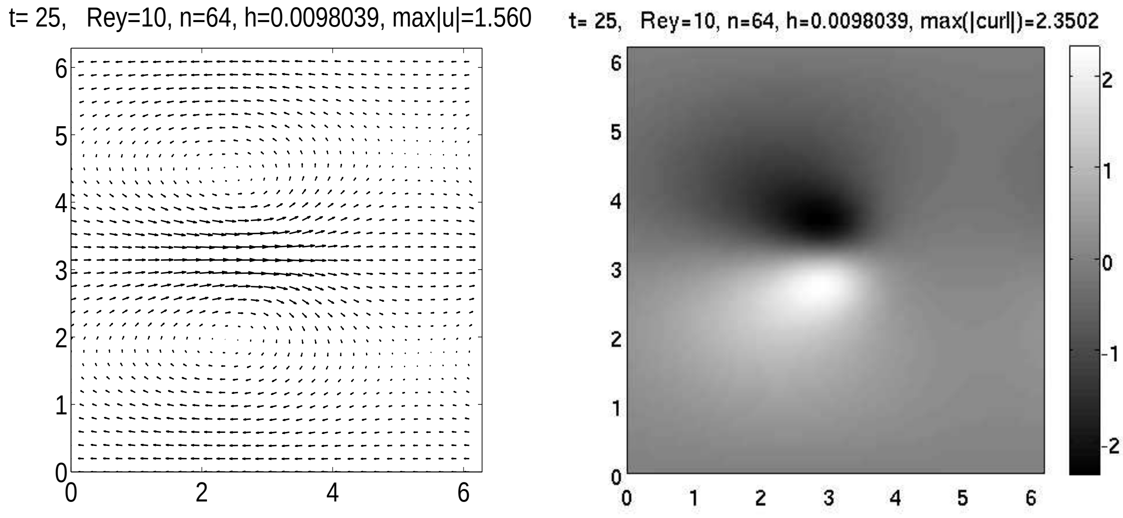

Figure 13: The stationary flow obtained for (48) at R = 10, and the associated vorticity.

visual diagnostic of the flow is the vorticity ω defined in (47) and plotted in the right panel, which illustrates that (for well-defined vortices also called eddies) $\omega > 0$ ($\omega < 0$) corresponds to a vortex with clockwise (counterclockwise) flow. This in fact can be taken as a general intuitive idea, even if proper vortices (i.e., flows with roughly circular closed streamlines) are not present. In the following we illustrate flows mainly by plotting the vorticity.

For larger R the flow no longer becomes stationary but develops a more and more (with increasing R) complicated time (and space) dependence. As an example consider

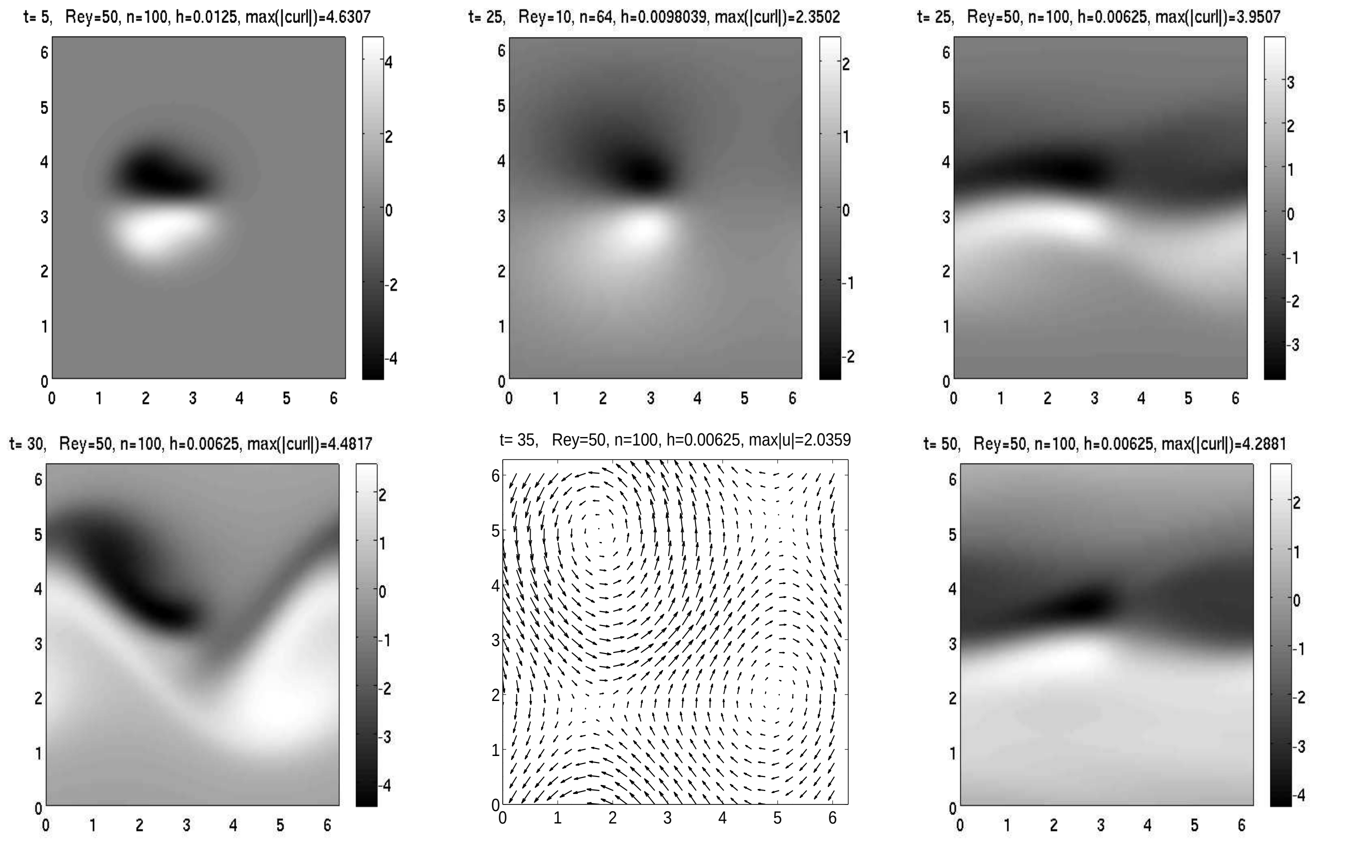

Figure 14: Non stationary flow obtained for (48) at R = 50.

Fig. 14 with R = 50. Initially, i.e., up to $t = 5$, say, the flow looks qualitatively similar to the one for R = 10 in Fig. 13 and seems to converge to a stationary flow u_s consisting of two vortices. However, u_s is unstable, i.e., a flow close to u_s moves away from u_s. This physical instability of u_s leads to the formation of a so-called vortex-sheet which bends and oscillates in time.

4.4.2 Shearing and stirring, and steps towards turbulence

As a second example we choose, for no particular reason, the forcing

$$g(x,y,t) = \mathrm{e}^{-2(\tilde{x}-1)^4-2(\tilde{y}-1)^4}\cos(t)\begin{pmatrix}\tilde{y}-1\\ -(\tilde{x}-1)\end{pmatrix} + \mathrm{e}^{-2(\tilde{x}+1)^4-(\tilde{y}+1/2)^4}\cos(t/5)\begin{pmatrix}0\\ \tilde{x}+1\end{pmatrix}, \tag{49}$$

where again $(\tilde{x},\tilde{y}) = (x-\pi, y-\pi)$. At $t = 0$ (49) corresponds to a so-called shearing localized around $(\tilde{x},\tilde{y}) = (-1,-1/2)$ and to some stirring localized around $(\tilde{x},\tilde{y}) = (1,1)$, see Fig.15 for $Pg(0)$. A time dependent forcing of course always gives a time dependent flow. However, for small R, i.e., R < 40, the flow essentially follows (with some delay) the forcing, see again Fig. 15 for an example.

For larger R the flow belonging to (49) becomes again more and more complicated, and for R ≥ 500, say, it is tempting to call the flow "turbulent". We strongly

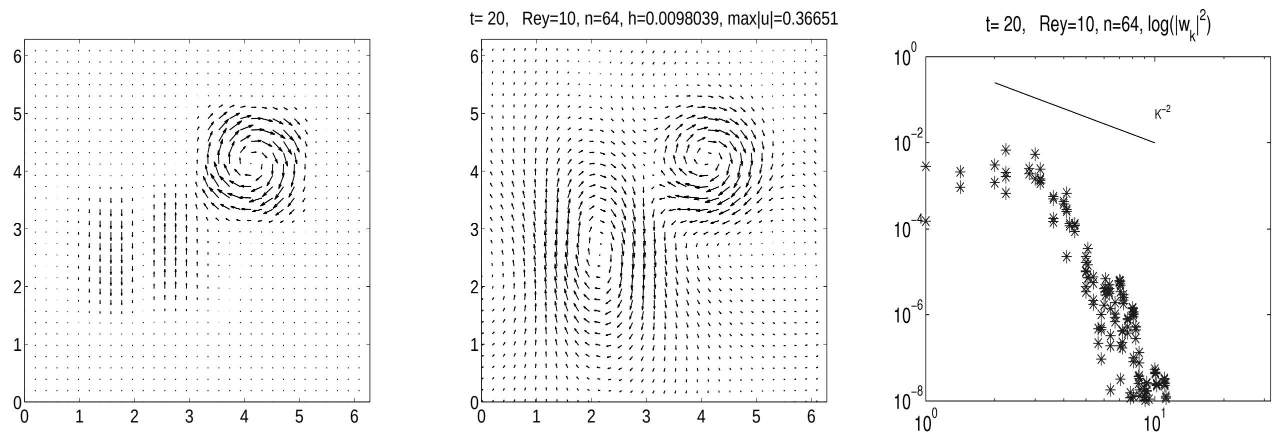

Figure 15: $Pg(0)$ with g from (49), a typical snapshot of the flow at small R, and the associated vorticity spectrum for comparison with Fig. 16.

encourage the reader to first explore this interactively. Concerning turbulence we start with the following words of caution: Despite more than 100 years of research in turbulence, it is not (yet) clear what turbulence is, i.e., how it should be defined mathematically. Physically it involves the idea of very complicated flows which in particular cannot be completely calculated or predicted and hence should be described using some statistical quantities like (time and space) averages.

That said, it should be clear that we do not claim that we now study turbulence. Rather we want to illustrate some very basic concepts. A set of quantities which play a big role in the description of turbulence are so called power spectra. Given some function $f : \mathbb{R}^d \to \mathbb{R}$, its power spectrum $E_f : \mathbb{R}_+ \to \mathbb{R}$ is defined as

$$E_f(\kappa) = \begin{cases} |\hat{f}(\kappa)|^2 & \text{if } d = 1, \\ \int_{|k|=\kappa} |\hat{f}(k)|^2 \, \mathrm{d}k & \text{if } d \geq 2, \end{cases} \tag{50}$$

where $\hat{f}(k)=(2\pi)^{-d/2} \int_{x\in\mathbb{R}^d} \mathrm{e}^{\mathrm{i}k\cdot x} f(x) \, \mathrm{d}x$ is the (proper) Fourier transform of f. The name power spectrum is based on Parsevals equality $\|f\|_{L^2}=\|\hat{f}\|_{L^2}$ which shows that the "energy" $\frac{1}{2}\|f\|_{L^2}^2$ of f can be expressed as the integral over the energy contained in all Fourier modes. Consequently $E_f(\kappa)$ denotes the energy (or power) contained in all Fourier modes with wave vectors of modulus κ.

Kolmogorov's theory of (stationary, homogeneous and isotropic[5]) turbulence, see, e.g., [Cho94, §3], amongst other things states that for turbulent flows the power spectrum in the "inertial range" (see below) has some universal (inverse) power law behaviour, independent of the flow details. One particular prediction is that for 3D turbulent flows the energy spectrum behaves (on time average) like

$$E_u(\kappa) = c\varepsilon^{2/3}\kappa^{-5/3}, \tag{51}$$

which is called Kolmogorovs 5/3 law and was essentially derived using dimensional

[5]on average, flow characteristics do not depend on time, position or direction

considerations. Here c is a numerical constant, and ε is the energy dissipation rate per unit volume, which again is assumed to be statistically stationary.

For 2D turbulent flows similar considerations yield that the power spectrum $W_\omega(\kappa)$ of the vorticity, aka enstrophy spectrum, behaves like

$$H_\omega(\kappa) \sim \beta^{2/3}\kappa^{-1}, \tag{52}$$

where $\beta = \frac{\mathrm{d}}{\mathrm{d}t}\int \|\omega\|^2 \,\mathrm{d}x$ is called the enstrophy injection rate, again assumed to be statistically stationary. Consequently, since $|\hat{u}(k)|^2 = |\hat{\omega}(k)|^2/|k|^2$, the energy spectrum behaves like

$$E_u(\kappa) \sim \beta\kappa^{-3}. \tag{53}$$

There is considerable numerical and experimental evidence for these power laws, see, e.g.[Sch00] and [KG02] for recent 2D compilations.

Figure 16 shows snapshots of the flow for a low-pass filtered version of (49) for $\mathrm{R} = 500$, in three different coordinates, all produced by `ns2d.m`. This low pass is given by

$$g_f(x) = \mathcal{F}^{-1}(\chi_\kappa \hat{u})(x), \tag{54}$$

where $\chi_\kappa(k) = 1$ if $\|k\|_\infty \le \kappa$ and zero else, see `lowp2d.m` for the implementation. Although g from (49) is an analytic vector field and hence $\hat{g}_k$ decays exponentially in k, the idea is to let the forcing act only on low order modes in a well controlled manner. This is useful to study in detail the energy transport to higher $|k|$, and in Fig. 16 we used $\kappa = 6$.

The vorticity plots in a), d) give impressions how the flow looks, b) show the decay of $|\hat{\omega}_k|^2$, while c), e) attempt to illustrate (52). Calculating the integral in (50) in some simulation is not straightforward since we would need to interpolate the circle $|k| = \kappa$ from the discrete wave numbers $(k_1, k_2) \in \mathbb{Z}^2$. Therefore, in c), e) we simply plot $|\hat{\omega}_k|^2$ over $\kappa = |k|$, in doubly logarithmic scale.

We then find (with a bit of good-will) a scaling κ^{-2} for the upper envelope, in the inertial range which here ranges from about $\kappa = 2$ to $\kappa = 10$. Beyond $\kappa \approx 10$ the so-called dissipation range starts where $|\hat{\omega}_k|^2$ decays exponentially (at least faster than any inverse power of κ). The scattered data in c), e) are due to the fact that these are just snapshots at some fixed time. If we average over time, then $k \mapsto |\hat{\omega}_k|^2$ in b) becomes more radially symmetric, and hence $|\hat{\omega}_k|^2$ over κ becomes less scattered. Then the enstrophy spectrum can be obtained from the envelope κ^{-2} by multiplication by $2\pi\kappa$ (the length of the circle with diameter κ in (50)), which indeed confirms $H(\kappa) \sim \kappa^{-1}$ in the inertial range. This should be compared to Fig. 15 where at $\mathrm{R} = 10$ there is no inertial range in the above sense.

Exercise 4.3 Implement the averaging $\left\langle |\hat{\omega}_k|^2 \right\rangle$ into `ns2d.m` $\rfloor$

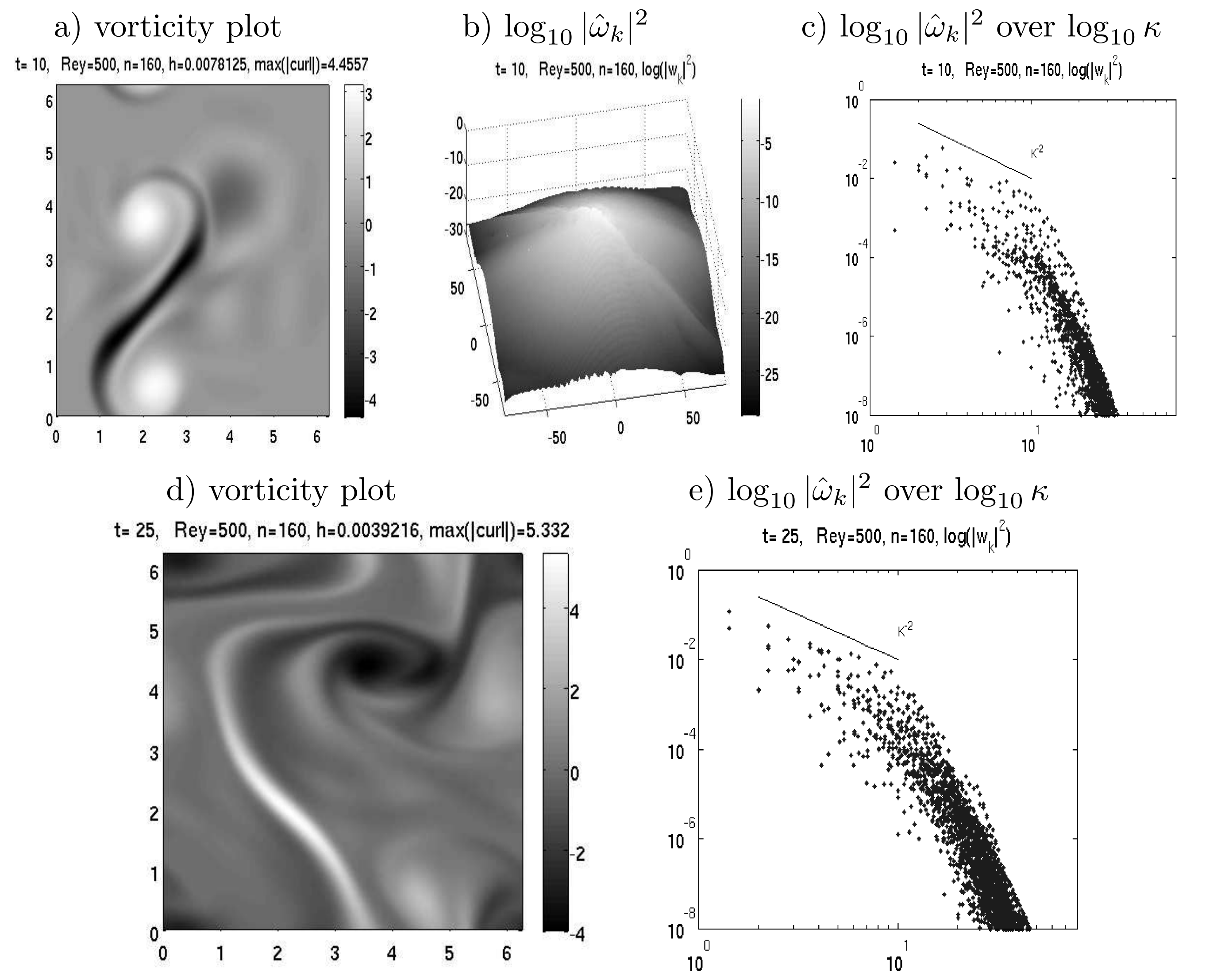

Figure 16: Towards turbulence: snapshots of the flow for forcing (49) with R = 500, calculation time 400s on a laptop computer. If we average the scatterplots in c), e) over time then we obtain the "universal scaling" $H(\kappa) \sim \kappa^{-1}$ in the inertial range $\kappa \in (2, 10)$, say. This becomes more evident at yet higher R. Then, however, we also need to increase n which ultimately leads to much longer calculation times and the simulations cannot be called "interactive" anymore (on standard laptop computers).

A Fourier series, DFT, FFT, and aliasing

Definition A.1 *A series of the form*

$$u(x) = \sum_{k\in\mathbb{Z}^d} \hat u_k e^{\mathrm{i}k\cdot x}, \tag{A.1}$$

is called Fourier series, and $\hat u_k$ is called the k^{th} Fourier coefficient.

The question is if and in what sense a function can be represented by its Fourier series, or, equivalently, in which norm Fourier series converge. The basic result is (see any textbook on Analysis)

Theorem A.2 *a) For $u \in L^2((0,2\pi)^d)$ we have L^2-convergence of the Fourier series, i.e., $\|u - \sum_{|k|\le N} \hat{u}_k e^{ik\cdot x}\|_{L^2} \to 0$ where*

$$\hat{u}_k = \langle e^{ik\cdot x}, u\rangle = \frac{1}{(2\pi)^d}\int u(x)e^{-\mathrm{i}k\cdot x}\,\mathrm{d}x. \tag{A.2}$$

b) We have Parsevals equality $\|\hat{u}\|_{l^2}^2 = \sum_{k\in\mathbb{Z}^d}|\hat{u}_k|^2 = \frac{1}{(2\pi)^d}\|u\|_{L^2}^2$.
c) There exists a $c > 0$ such that if $u \in C^m$ is 2π-periodic in each direction, then $|\hat{u}_k| \le c(1+|k|)^{-m}$.

Pointwise convergence of Fourier series is a rather delicate issue. For instance, the Fourier series of $u \in L^1$ may diverge almost everywhere (Kolmogorov 1921), while for $u \in L^2$ we have convergence almost everywhere (Carleson 1966). In summary, no necessary *and* sufficient conditions are known that the Fourier series of a function u converges pointwise to $u(x)$. There are however various sufficient conditions in the literature, see for instance the Dini criterion.

Moreover, there is a very helpful connection between so-called Sobolev spaces of functions and weighted ℓ^2 spaces for their Fourier coefficients. Sobolev spaces are a standard tool in the analysis of PDE; for instance the global existence and uniqueness of (smooth) solutions of the 2D Navier-Stokes can *easily* be proved using the space $H^1_{\text{per}}((0,2\pi)^2)$, see, e.g. [Rob01].

Numerically we can only deal with finitely many Fourier coefficients, functions $u(x)$ are only known in a discrete sense, and the integrals (A.2) in general have to be evaluated numerically. The basic idea is to use the trapezoidal rule, which leads to the so-called discrete Fourier transform DFT. We focus on 1D, consider $x \in [0,2\pi]$, and for reference give the main formulas, see, e.g., [CHQZ88, §2.1.2] or [Coo98, §6.5, including comments on `matlab`] for more details.

For $n \in \mathbb{N}$ let u be given at n equally spaced points $x_j = 2\pi j/n$, $j = 0,\dots,n-1$. Then the discrete Fourier coefficients $\tilde{u}_k$ are defined as

$$\tilde{u}_k = \frac{1}{n}\sum_{j=0}^{n-1} u(x_j)\mathrm{e}^{-\mathrm{i}kx_j}, \quad -n/2 \le k \le n/2-1, \tag{A.3}$$

and from the orthogonality

$$\frac{1}{n}\sum_{j=0}^{n-1}\mathrm{e}^{-\mathrm{i}px_j} = \begin{cases} 1 & \text{if } p = nl, l \in \mathbb{Z} \\ 0 & \text{else}\end{cases} \tag{A.4}$$

we immediately find the inversion formula

$$u(x) = \sum_{k=-n/2}^{n/2-1} \tilde{u}_k \mathrm{e}^{\mathrm{i}kx_j}. \tag{A.5}$$

In (A.3) we need $\mathcal{O}(n^2)$ operations to calculate the DFT of u, and similar in (A.5). The FFT exploits symmetries of the DFT to calculate the DFT in $\mathcal{O}(n\log n)$ operations. It works best for n some power of 2. The `matlab` command `fft` computes (A.3) without the factor $1/n$ but instead puts it in front of the inverse transform, called as `ifft`. Also note that `fft` calculates a vector `d` with a shifted indexing compared to (A.3), see (20).

For our purposes the most important fact is that if the Fourier series of u converges pointwise at each x_j, then the $\tilde{u}_k$ approximate the $\hat{u}_k$ via

$$\tilde{u}_k = \hat{u}_k + e_k, \quad k = -n/2, \dots, n/2-1, \quad \text{where} \quad e_k = \sum_{l\in\mathbb{Z}\setminus\{0\}} \hat{u}_{k+nl}. \tag{A.6}$$

The e_k are called the aliasing errors. However, if $u \in C^m$, then $|\hat{u}_k| \le c(1+|k|)^{-m}$ by Theorem A.2c), such that for fixed k the aliasing error e_k goes to zero as $n \to \infty$. Moreover, if u is band limited, i.e., if there exists some $M \in \mathbb{N}$ such that $\hat{u}_k = 0$ for $|k| > M$, then

$$e_k = 0 \text{ if } n > 2M, \tag{A.7}$$

i.e., if the sampling frequency is high enough. This is called the Nyquist-Shannon sampling theorem. In particular, a band limited function u can be reconstructed *exactly* from its finite DFT with $n > 2M$. As a further application we mention the CD: for CDs, musical signals are treated as band limited to 20kHz and therefore sampled at 44.1 kHz, allowing some margin for technical issues like the operation of filters, which technically cannot give an ideal sharp cut-off in Fourier space similar to the mathematical low-pass in (54).

B Derivation of the Navier-Stokes equations

The fluid is modeled as a continuum, i.e. we do not consider the molecules of the fluid separately. The velocity field at a position $x \in \mathbb{R}^d$ at a time t is denoted by $u(x,t) \in \mathbb{R}^d$ for $d = 2, 3$, and the density by $\rho = \rho(x,t) \in \mathbb{R}$. In general by the internal friction of the fluid heat will be produced which leads to a coupling of the Navier-Stokes equations with a heat equation. We neglect this aspect and ignore the coupling of the Navier-Stokes equations with heat. Also, we focus on the so-called incompressible case where the density is constant in space and time.

Conservation of mass. We consider a fixed test volume V with surface S. The mass in V can only change by the flow through the boundary S, i.e.

$$\partial_t \int_V \rho dV = -\int_S \rho u \cdot n dS = -\int_V \nabla \cdot (\rho u) dV$$

using Gauss' integral theorem, where $n(x) = (n_1, \dots, n_d)(x)$ is the outer unit normal in the point x at the boundary S. Since this relation holds for all test volumes V the integrands must be equal, i.e.

$$\partial_t \rho + \operatorname{div}(\rho u) = 0. \tag{B.1}$$

For imcompressible fluids ρ is constant in space and time such that (B.1) reduces to $\nabla \cdot u = 0$.

Conservation of momentum. Similarly the momentum $\int_V \rho u \, \mathrm{d}V$ of a test volume V can only change by flow through the boundary and forces on the volume. Internal forces, especially friction, are modeled by forces f_i on the surface of the test volume. In order to do so we assume the existence of a matrix $\sigma = (\sigma_{ij})_{i,j=1,\ldots,d}$, the stress tensor, such that

$$f_i = \sum_{j=1,\ldots,d} \sigma_{ij} n_j.$$

Application of the integral theorem of Gauss and the above arguments yield

$$\rho[\partial_t u_i + (u \cdot \nabla) u_i] = \nabla \cdot \sigma_{i\cdot} + g_i$$

where g_i models external volume forces, or, in coordinates,

$$\rho \left[\partial_t u_i + \sum_{j=1,\ldots,d} (u_j \cdot \partial_{x_j}) u_i \right] = \sum_{j=1,\ldots,d} \partial_{x_j} \sigma_{ij}.$$

In order to obtain a closed set of equations we need relations between the stress tensor σ and (u, ρ). These depend on the fluid, e.g. the function $\sigma = \sigma(u, \rho)$ differs strongly between water and honey. It is possible that σ also depends on the past. Such a relation is called *constitutive law.*

A fluid with internal friction is called *viscous.* The constitutive law is then given by

$$\sigma_{ij} = -p\delta_{ij} + \tau_{ij}, \quad \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & \text{else} \end{cases}.$$

Here p denotes the pressure and τ_{ij} models internal friction also called viscous stress. It can be shown that σ must always be symmetric. Moreover, the simplest assumption is that the viscous stress is proportional to the strains $\partial_{x_i} u_j$. For incompressible fluids this leads to the definition

$$\tau_{ij} = \mu(\partial_{x_j} u_i + \partial_{x_i} u_j).$$

where μ is called dynamic viscosity.[6] Using the kinematic viscosity $\nu = \mu/\rho$, the Navier-Stokes equations are then given by

$$\begin{aligned} \partial_t u + (u \cdot \nabla) u &= -\frac{1}{\rho} \nabla p + \nu \Delta u + \frac{1}{\rho} g, \\ \nabla \cdot u &= 0. \end{aligned} \tag{B.2}$$

[6] This is really a modeling issue: apart from simplicity there is no reason why τ and hence σ should only depend on the first derivatives of u, and only linearly.

The Reynolds number. In order to eliminate the physical units from the Navier-Stokes equations, let U be a typical velocity and l be a typical length of the flow. We set

$$u = Uu^*, \quad x = lx^*, \quad p = \rho U^2 p^*, \quad t = lt^*/U, \quad g^* = \frac{l}{\rho U^2} g,$$

and, after dropping the *, obtain the dimensionless Navier-Stokes equations (37) with Reynolds number $\mathrm{R} = Ul/\nu$. Thus R is large if any of the following holds (or any two, or all three together): "fast flow" (U large) or "large scale" (l large) or small viscosity (ν small). In nature, Reynolds numbers vary over several orders of magnitude. Although these values depend on exact definitions of typical lengths and velocities, here are some typical values from http://en.wikipedia.org/wiki/Reynolds_number: Spermatozoa: 10^{-4}, blood flow in aorta: 10^3, person swimming: 10^6, large ship 10^9. Reynolds numbers for atmospheric flows can easily exceed 10^{12}. In the limit $\mathrm{R} \to \infty$ we obtain the Euler equations which in the incompressible case correspond to (B.2) with $\nu = 0$.

The inverse Reynolds number roughly sets the linear decay rate $\mathrm{e}^{-t|k|^2/\mathrm{R}}$ of modes with wave number k. Thus, large R means that even high wave number (small scale) modes decay slowly. The forcing f and the nonlinearity $(u \cdot \nabla)u$ may then create "turbulent flows" by transfering energy from low modes (small $|k|$) to intermediate modes (intermediate $|k|$) in a so-called energy cascade (inertial range). Modes with very large $|k|$ are however always exponentially small (dissipation range), see also the discussion in Sec. 4.4. Depending on further characteristics like the geometry of some flow, the onset of turbulence in engineering applications typically is around $\mathrm{R} \approx 10^4$ to $\mathrm{R} \approx 10^6$. A rule of thumb then says that for the numerical discretization with a uniform grid one needs about $n = \mathrm{R}^{3/4}$ points or equivalently $n = \mathrm{R}^{3/4}$ Fourier modes in each space direction.

References

[BK00] J. Billingham and A. C. King. *Wave motion.* Cambridge University Press, Cambridge, 2000.

[Cho94] A. J. Chorin. *Vorticity and turbulence.* Springer-Verlag., 1994.

[CHQZ88] C. Canuto, M.Y. Hussaini, A. Quarteroni, and T.A. Zang. *Spectral Methods in Fluid Dynamics.* Springer, 1988.

[Coo98] J. Cooper. *Introduction to partial differential equations with MATLAB.* Birkhäuser, 1998.

[CT65] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comput.*, 19:297–301, 1965.

[DB02] P. Deuflhard and F. Bornemann. *Scientific computing with ordinary differential equations.* Springer-Verlag, New York, 2002.

[DG95] Ch. R. Doering and J.D. Gibbon. *Applied analysis of the Navier-Stokes equations.* Cambridge Univ. Press, 1995.

[Fow97] A. C. Fowler. *Mathematical models in the applied sciences.* Cambridge University Press, 1997.

[FRMT01] C. Foias, R. Rosa, O. Manley, and R. Temam. *Navier-Stokes equations and turbulence.* Cambridge University Press , 2001.

[GS83] P. Gray and S. K. Scott. Autocatalytic reactions in the isothermal, continuous stirred tank reactor. *Chemical Engineering Science*, 38(1):29–43, 1983.

[Hen81] D. Henry. *Geometric Theory of Semilinear Parabolic Equations.* Springer Lecture Notes in Mathematics, Vol. 840, 1981.

[HJ07] J. Hoffman and Cl. Johnson. *Applied mathematics: Body and soul (Volume 4). Computational turbulent incompressible flow.* Springer, 2007.

[HJ08] J. Hoffman and Cl. Johnson. Blow up of incompressible Euler solutions. *BIT*, 48(2):285–307, 2008.

[HNW93] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I.* Springer, 1993.

[HW96] E. Hairer and G. Wanner. *Solving ordinary differential equations. II: Stiff and differential-algebraic problems.* Springer, 1996.

[KG02] H. Kellay and W. Goldburg. Two-dimensional turbulence: a review of some recent experiments. *Reports on Progress in Physics*, 65:845–894, 2002.

[Mur89] J. D. Murray. *Mathematical biology*, volume 19 of *Biomathematics.* Springer-Verlag, Berlin, 1989.

[Rob01] J. C. Robinson. *Infinite-dimensional dynamical systems.* Cambridge University Press, 2001.

[Sch79] J. Schnakenberg. Simple chemical reaction systems with limit cycle behaviour. *J. Theoret. Biol.*, 81(3):389–400, 1979.

[Sch00] N. Schorghofer. Energy spectra of steady two–dimensional turbulent flows. *PRE*, 61(6):6572–6577, 2000.

[Str92] W. A. Strauss. *Partial differential equations – An Introduction.* John Wiley & Sons Inc., New York, 1992.

[Tem97] R. Temam. *Infinite-dimensional dynamical systems in mechanics and physics.* Springer-Verlag, New York, 1997.

[Tem01] R. Temam. *Navier-Stokes equations.* AMS, Providence, RI, 2001.

[vW85] W. von Wahl. *The equations of Navier-Stokes and abstract parabolic equations.* Vieweg, 1985.