university of groningen
faculty of mathematics and natural sciences

Burgers' equation: numerical models and filtering

R. Bosma

Johannes Martinus Burgers (1895-1981) [9]

Master Thesis in Applied Mathematics

Supervisor: R.W.C.P. Verstappen

August 2008

Burgers' equation: numerical models and filtering

R. Bosma

Supervisor(s):
R.W.C.P. Verstappen
Institute of Mathematics and Computing Science
P.O. Box 407
9700 AK Groningen
The Netherlands

Preface

> A little boy and his mother were walking in the local supermarket. From isle to isle they walked, searching for the neccesary groceries. After a while the shopping basket was filled and together they went to the check-out. Whilst unloading the basket the boy quitetly spoke to his mother, telling her exactly how much she had to pay...

Mathematics has always been an important part of my life and I have always been fascinated by this versatile subject. Therefore it isn't strange that in high-school my favourite subject, along with music, was mathematics. Several of my classmates didn't see the purpose of it and therefore disliked it. Purpose or no purpose, math was fun. Learning all about graphs and "magic" numbers proved to be not only interesting, but also useful. Mathematics really helped me, as well as many other people that walk this earth, to explore and sharpen my level of abstract thinking. Abstract thinking helps one to sum up problematic situations very quickly and learn how to deal with them accordingly. Apart from learning basic arithmetic skills, the true purpose of high-school mathematics in my opinion is to develop a greater sense of abstract thinking.
Thus I chose to take up mathematics at the university of Groningen ("Universiteit van Groningen" or R*u*G for short). Although I hadn't really given much thought on what my future would look like afterwards, I still chose to study mathematics because that's what my heart told me to do. My choosing to go to the R*u*G was based on the fact that the R*u*G appeared to be a kindhearted and warm university as well as it being the nearest university.
It was only at the R*u*G that I got to know more about numerical mathematics and technical mechanics. Aerodynamics and Computational Fluid Dynamics really got my attention and I started to develop a profound love for these subjects. I wanted to know more about boundary layers and the aerodynamics of cars, cyclists and since badminton is one of my hobbys: shuttlecocks.
With this in mind I went to see professor Arthur Veldman to talk about my thesis. A swift calculation showed that a badminton shuttle is too large and moves too fast for the computers available at the R*u*G. Therefore professor Veldman suggested I should contact Dr. Roel Verstappen, my supervisor, about Burger's equation. Dr. Verstappen has already done research in DNS methods for solving the Navier-Stokes equations. My challenge lies in applying his DNS approach to Burgers' equation and extend it to two dimensions.
This report is the result of 2 years of hard work and has by far been the largest project I have ever worked on. During this period I have had the support of several people that are dear to me. I hereby want to thank them in no particular order

- Aaltje Lubbers. My girlfriend who kept faith in me and supported me in the most loving manner.
- My father and mother. For their continuous support during my entire study.

- Roel Verstappen. My supervisor, a very kind, gentle and patient person who with his friendly smile really, probably unknowingly, motivated me in difficult times.
- Joop Helder. For sharing his knowledge and time at the end of my thesis.
- Everyone I have forgotten to mention...

I might have completed my study in more than the advised time, yet I do not regret this at all. During my college years I have participated in a different variety of activities through which I have improved my social skills and my knowledge of people. Things I find equally important as sheer knowledge.
I have had a great time at the R*u*G with a lot of great memories. But now it is time for me to take the next step in my life. The time has come for me to enter the "working world".

Emmen, August 2008 **Ronald Bosma**

Contents

1 Introduction

Direct Numerical Simulation is of great interest to the world of Computational Science due to its proven precision. There are people who state that everything has a drawback and Direct Numerical Simulation (DNS) is no exception to this statement. The drawback of this method in combination with computers nowadays lies in speed. This method is rather slow compared to other methods. If we want to keep the precision of the method we have to come up with a way to decrease the computation time of DNS. In this paper we will apply four different DNS methods to Burgers' equation, two Lagrangian methods and two Spectro-Consistent methods. We want to determine which of these methods is most suitable to use for Burgers' equation.
First we will look at a simple one-dimensional flow problem, without time, in order to show which type of discretization method ought to be used to discretize both convective as well as diffusive terms of the Navier-Stokes equations. Part of this proces is to find out whether the fourth-order versions of these methods outclass the second-order versions.
Based on these one-dimensional results we will investigate how our programs handles turbulence by looking at Burgers' equation, still in one dimension, hence adding time. From this investigation of different numerical methods we will see that the Spectro-Consistent methods are always stable as opposed to the Lagrangian methods, which are not always stable.
The second part of this paper will focus on applying filtering to the convective term of Burgers' equation. Reason for this is to be able to compute less small scales of motion and therefore use coarser grids. We will first take a look at filtering in physical space and finally take a look at filtering in spectral space. In spectral space we will take a look the energy for different wavenumbers.
The structure of this document can be summarized by the following scheme:

- Decide what discretization methods to use for convective as well as diffusive terms based on a one-dimensional flow problem;
- Decide to choose either second- or fourth order discretization methods;
- Apply a filter in physical space to the chosen discretization method;
- Convert Burgers' equation to spectral space and apply filtering;
- Extend Burgers' equation to two dimensions and apply a DNS method;
- And finally report our findings and conclude whether or not filtering should be applied to these kind of problems and how filtering can efficiently used to speed up DNS methods.

To obtain a complete image of this entire process, the problems encountered during the process will be mentioned.

2 A one-dimensional convection-diffusion problem

In order to get some understanding of Direct Numerical Simulation we will try to solve the one-dimensional Burgers' equation. At first we will take a look at a simplified version of Burgers' equation in order to compare four different methods: a second and a fourth order Lagrangian method and a second and a fourth order Spectro-Consistent method as proposed by Verstappen en Veldman in [3].
In the final part of this section we will try and solve Burgers' equation using the Spectro-Consistent methods.
The one-dimensional Burgers' equation reads:

$$\frac{\partial u}{\partial t} + u_c \frac{\partial u}{\partial x} - k \frac{\partial u^2}{\partial^2 x} = f(x,t), \tag{1}$$

where $u(x,t)$ is the velocity in x-direction. t is time, k is the diffusive coefficent. The right-hand side of the equation is $f(x,t)$. In Burgers' equation $u_c = u$, creating a non-linear convective term. However in the first part of our investigation we have chosen u_c to be constant, which is why we chose this particular notation. In another section we will see the $k = 1/Re$, where Re is Reynolds' number.

2.1 Simple case: Lagrangian or Spectro-Consistent?

At first we will rid ourselves of several terms of this equation. The velocity $u(x,t)$ is taken constant in time and u_c is taken constant and equal to one. Furthermore we take $u(0) = 0$ and $u(1) = 1$. Also $f(x,t)$ is taken equal to zero. This leaves us the following boundary problem:

$$\frac{\partial u}{\partial x} - k \frac{\partial u^2}{\partial^2 x} = 0,\ u(0) = 0,\ u(1) = 1. \tag{2}$$

Eq. (2) is discretized using two second-order methods, from now on referred to as C2 and L2, where C2 stands for second-order Spectro-Consistent and L2 stands for second-order Lagrangian. Likewise the equation is also discretized using two fourth-order methods, C4 and L4. C4 being a fourth-order Spectro-Consistent and L4 a fourth-order Lagrangian method. These methods are described in the following section.
The naming of the spectro-consistent methods is chosen as proposed in [8].

2.1.1 In depth mathematics

The main difference between the Spectro-Consistent and the Lagrangian methods is merely a different way of discretising the convective term $\frac{\partial u}{\partial x}$. In accordance with the naming of the different methods the convective term is discretized using second- and fourth-order Spectro-Consistent and Lagrangian methods. Whereas the diffusive term, $-k\frac{\partial u^2}{\partial^2 x}$ is in both the C2 and C4 as well as in the L2 and L4 methods discretized using a Lagrangian scheme of the accessory order. In the fourth-order methods also Richardson extrapolation is used with the equivalent second-order discretizations applied to a

larger-sized grid.
As was mentioned before u_c is taken constant in the convection-diffusion equation Eq. (1). The spatial discretization of Eq. (1) can now be given in matrix-vector form. This results in:

$$\boldsymbol{\Omega}_0 \frac{\mathrm{d}\boldsymbol{u}_h}{\mathrm{d}t} + \boldsymbol{C}_0(u_c)\boldsymbol{u}_h + \boldsymbol{D}_0\boldsymbol{u}_h = \boldsymbol{f}, \tag{3}$$

where the vector $\boldsymbol{u}_h$ is built of the discrete velocities $u_i(= u(x_i))$ and $\boldsymbol{\Omega}_0$ is a diagonal matrix with the spacings of the mesh as its entries:
$(\boldsymbol{\Omega}_0)_{i,i} = \frac{1}{2}(x_{i+1} - x_{i-1})$. The tridiagonal matrix $\boldsymbol{C}_0$ represents the convective operator whereas $\boldsymbol{D}_0$, which is also tridiagonal, represents the diffusive operator. $\boldsymbol{f}$ is the vector which, in analogy to $\boldsymbol{u}_h$, is the discrete representation of the right-hand side $f(x,t)$ of Eq. (1).
In this discrete form of Eq. (1) time is still included, while we wanted to look at the case whitout time. Removing time must be done with care because of the term $\boldsymbol{\Omega}_0$ preceeding $\frac{\mathrm{d}\boldsymbol{u}_h}{\mathrm{d}t}$. Multiplying the entire equation Eq. (3) with $\boldsymbol{\Omega}_0^{-1}$ before taking out time, gives us the correct numerical form of Eq. (2)

$$\boldsymbol{\Omega}_0^{-1}\boldsymbol{C}_0(u_c)\boldsymbol{u}_h + \boldsymbol{\Omega}_0^{-1}\boldsymbol{D}_0\boldsymbol{u}_h = \boldsymbol{0} \tag{4}$$

In Eq. (4) $\boldsymbol{f}$ is taken zero, since in Eq. (2) $f(x,t)$ is also zero.

2.1.2 L2 and C2

In the L2 and C2 methods $\boldsymbol{D}_0$, is the same. Truncating Taylor-series in a smart way leads to

$$\boldsymbol{D}_0 = k\boldsymbol{\Delta}_0^*\boldsymbol{\Lambda}_0^{-1}\boldsymbol{\Delta}_0,$$

where $\boldsymbol{\Delta}_0$, the difference matrix, is defined by $(\boldsymbol{\Delta}_0\boldsymbol{u}_h)_i = u_i - u_{i-1}$ and $\boldsymbol{\Lambda}_0$, a diagonal matrix, is defined by $(\boldsymbol{\Lambda}_0)_{i,i} = \delta x_i$, with $\delta x_i = x_i - x_{i-1}$. The convective term however is treated differently. In the L2 method the convective term is found in the same manner as the diffusive term. Taylor-series expansions lead to

$$\frac{\partial u}{\partial x}(x_i) \approx \frac{\delta x_i^2 u_{i+1} + (\delta x_{i+1}^2 - \delta x_i^2)u_i - \delta x_{i+1}^2 u_{i-1}}{\delta x_{i+1}\delta x_i(\delta x_{i+1} + \delta x_i)} \tag{5}$$

Note that equation Eq. (5) can also be obtained by constructing a second-order polynom, a parabola, through the three points (x_{i-1}, u_{i-1}), (x_i, u_i) and (x_{i+1}, u_{i+1}) and differentiating that parabola at $x = x_i$.
Intuitively this appears to be the best approximation of the derivative constructed from three given points, and regarding the local truncation error it is. The backdraw of this approach lies in the fact that it doesn't yield the physical property of energy conservation and other properties the continuous problem conserves. In problems where turbulence is involved energy conservation is, as we shall see, particularly important because of the subtle interaction between convective transport and physical dissipation

also at small scales of motion.
As can be seen in [3], $\boldsymbol{C}_0(u_c)$ has to be skew-symmetric in order to establish discrete energy conservation:

$$\boldsymbol{C}_0(u_c) + \boldsymbol{C}_0^*(u_c) = \boldsymbol{0} \tag{6}$$

The L2 method described in Eq. (5) will not lead to a skew-symmetric $\boldsymbol{C}_0(u_c)$ since, for nonuniform grids, it will have nonzero diagonal entries.

The C2 method however, is constructed on physical grounds:

$$u_c \frac{\partial u}{\partial x}(x_i) \approx u_c \frac{u_{i+1} - u_{i-1}}{x_{i+1} - x_{i-1}} = \left(\boldsymbol{\Omega}_0^{-1} \boldsymbol{C}_0(u_c) \boldsymbol{u}_h\right)_i, \tag{7}$$

where the entries of the tridiagonal matrix $\boldsymbol{C}_0(u_c)$ are given by $\boldsymbol{C}_0(u_c)_{i,i-1} = -\frac{1}{2}u_c$, $\boldsymbol{C}_0(u_c)_{i,i} = 0$ and $\boldsymbol{C}_0(u_c)_{i,i+1} = \frac{1}{2}u_c$, resulting in a skew-symmetric $\boldsymbol{C}_0(u_c)$.
This could also have been obtained by constructing a straight line through the points (x_{i-1}, u_{i-1}) and (x_{i+1}, u_{i+1}) and taking its derivative at $x = x_i$.
The discussion regarding these two methods will be performed after the introduction of the fourth-order methods.

2.1.3 L4 and C4

Higher-order methods usually give better results due to smaller truncation errors opposed to lower-order methods. This also holds for DNS methods.
Therefore Eq. (3) is going to be transformed into a fourth-order method. In order to achieve this, a similar equation will be constructed using a two times larger control volume:

$$\boldsymbol{\Omega}_2 \frac{\mathrm{d}\boldsymbol{u}_h}{\mathrm{d}t} + \boldsymbol{C}_2(u_c)\boldsymbol{u}_h + \boldsymbol{D}_2\boldsymbol{u}_h = \boldsymbol{0}, \tag{8}$$

where $\boldsymbol{\Omega}_2$ is a diagonal matrix with $(\boldsymbol{\Omega}_2)_{i,i} = \frac{1}{2}(x_{i+2} - x_{i-2})$. $\boldsymbol{C}_2(u_c)$, the convective term, is given by $(\boldsymbol{C}_2(u_c)\boldsymbol{u}_h)_i = \frac{1}{2}u_c(u_{i+2} - u_{i-2})$. The diffusive term $\boldsymbol{D}_2 = k\boldsymbol{\Delta}_2^*\boldsymbol{\Lambda}_2^{-1}\boldsymbol{\Delta}_2$ is given by $(\boldsymbol{\Delta}_2\boldsymbol{u}_h)_i = u_{i+1} - u_{i-1}$ and $(\boldsymbol{\Lambda}_2)_{i,i} = x_{i+1} - x_{i-1}$. Note that since we have taken $\boldsymbol{f}$ to be $\boldsymbol{0}$ in this example, the right-hand side of Eq. (8) equals $\boldsymbol{0}$.
The leading term in the discretization error can be removed if Richardson extrapolation is applied to Eq. (3) and Eq. (8). The errors in these expressions are third-order errors. On a uniform grid we would now have to take $2^3\times$ Eq. (3) $-$ Eq. (8). On a non-uniform grid the same weights are taken in order for us not to break the symmetry. This leads to the following system:

$$(8\boldsymbol{\Omega}_0 - \boldsymbol{\Omega}_2)\frac{\mathrm{d}\boldsymbol{u}_h}{\mathrm{d}t} + (8\boldsymbol{C}_0(u_c) - \boldsymbol{C}_2(u_c))\boldsymbol{u}_h + (8\boldsymbol{D}_0 - \boldsymbol{D}_2)\boldsymbol{u}_h = \boldsymbol{0} \tag{9}$$

Taking out time in the same manner as above gives:

$$(8\boldsymbol{\Omega}_0 - \boldsymbol{\Omega}_2)^{-1}(8\boldsymbol{C}_0(u_c) - \boldsymbol{C}_2(u_c))\boldsymbol{u}_h + (8\boldsymbol{\Omega}_0 - \boldsymbol{\Omega}_2)^{-1}(8\boldsymbol{D}_0 - \boldsymbol{D}_2)\boldsymbol{u}_h = \boldsymbol{0} \tag{10}$$

2.1.4 Starting the battle: error analysis

The boundary problem Eq. (2) can be solved analytically, therefore the solutions found using the four methods mentioned above, can be compared to the exact solution of the problem. We can define the error as the norm of the difference between the exact solution computed at the grid-points and the numerical approximation of the solution at the grid-points using the different methods.
The grid is chosen by dividing the interval $[0, 1]$ into two different intervals $[0, 1-d]$ and $[1-d, 1]$ with $0 < d < 1$. Both intervals will contain an equal amount of gridpoints.
If we vary the total number of gridpoints N, we can construct an error curve from the errors we get for each individual number of gridpoints by simply connecting the points. Doing so for all four methods with $k = 0.001$, $N = 8, 16, 32, 64, 128$ and $d = 10k = 0.01$, the following figure can be made:

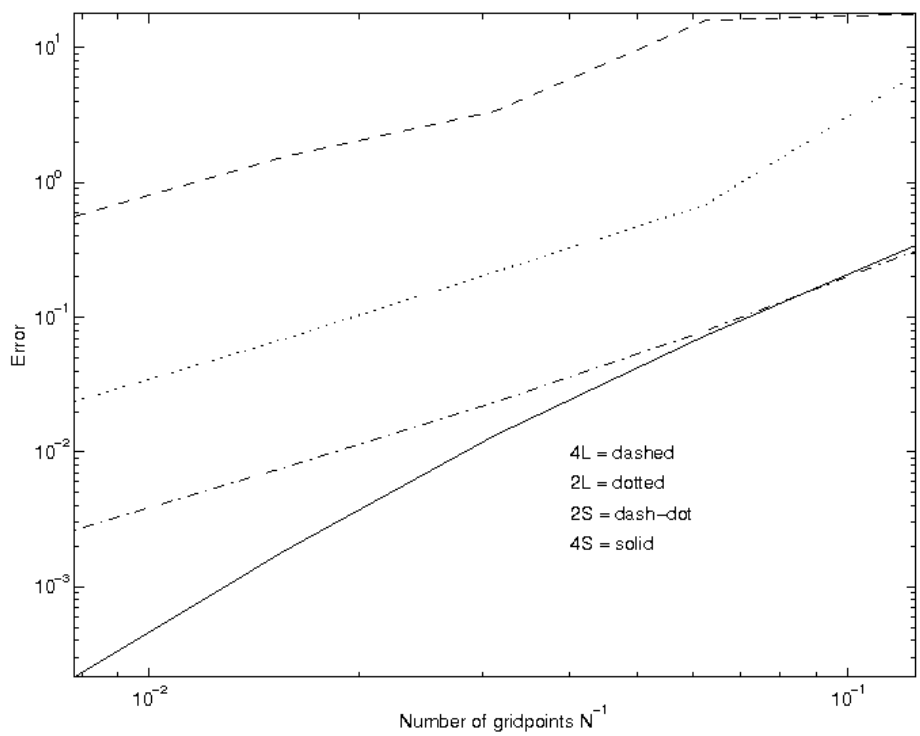

Figure 1: Comparing second- and fourth-order Spectro-Consistent and Lagrangian methods.

Looking closely at Figure 1 it can be seen that the errors for the Lagrangian methods become substantially large on the coarser grids. The Spectro-Consistent methods give better results and the C4 method appears to be superior to the C2 method, as had been predicted.

2.1.5 Round two: C2 vs C4

Since L2 and L4 are out of the league of C2 and C4, we will now focus on whether or not C4 is indeed a better method than C2. A more interesting and challenging problem is found when u_c is not taken constant:

$$u\frac{\partial u}{\partial x} - k\frac{\partial u^2}{\partial^2 x} = 0,\ u(0) = 0,\ u(1) = 1. \tag{11}$$

In the absence of an exact solution, we create an approximation of the exact solution by simulating the solution on a dense grid, $N \geq 600$, using the C4 method.
The same grid-structure and values of k, N and d as were used before, leads us to the following error-figure:

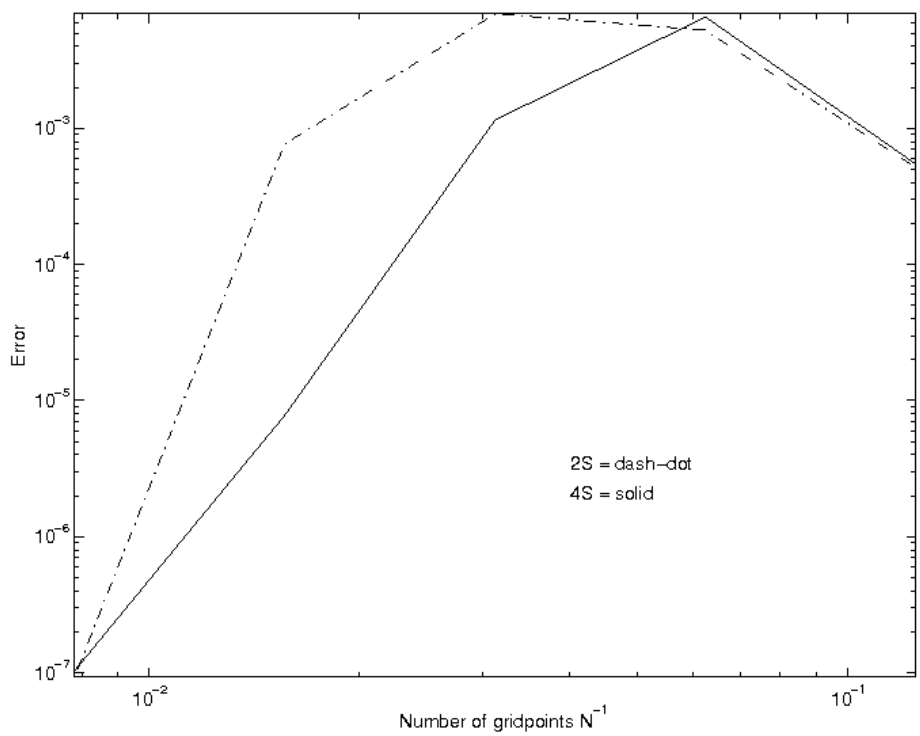

Figure 2: Comparing second- and fourth-order Spectro-Consistent methods.

First of all note that on the horizontal axis $1/N$ represents the gridsize. The C4 method is the better method in general, as can be seen in the figure. However for small grids the C2 method tends to be equally good if not better than the C4 method. This can be explained by the fact that the inner part of these smaller grids consist of a number of points of the same order as the number of points on the boundary. Since the boundary is treated equally in both the C2 and C4 method and the inner part of the grid is of the same order, the overall performance of the methods will be approximately the same.
For large grids both methods appear to be equal as well. As was mentioned above,

the exact solution is merely an approximation on a dense grid. The largest grid in Figure 2 consists of 128 gridpoints which is approximately a quarter of the number of gridpoints used for the "exact" solution. Therefore the error of both the "exact" and the approximated solution are almost the same. This holds for both methods and the error therefore is of the same order. Combining the above we can conclude that the C4 method outclasses the C2 method.

2.1.6 Outcome of the battle

In [1] Veldman and Verstappen show us that the eigenvalues of both the L2 and L4 methods lie in the instable halfplane causing this instability. The Spectro-Consistent methods however will never have instable eigenvalues because of the way these methods are constructed.
In Figure 1 these results are confirmed and although the Lagrangian methods at first, by construction, appear to be superior to the Spectro-Consistent methods, research and theory have proven the opposite. Therefore from now on we will merely investigate the Spectro-Consistent methods. Furthermore the results show us that overall the C4 method gives better results than the C2 method.

2.2 Adding time

In general we can state that the spatial discretization of Eq. (1) is of the following form, regardless of the method being used being second order or fourth order:

$$\boldsymbol{\Omega}\frac{\mathrm{d}\boldsymbol{u}_h}{\mathrm{d}t} + \boldsymbol{C}(u_c)\boldsymbol{u}_h + \boldsymbol{D}\boldsymbol{u}_h = \boldsymbol{f} \tag{12}$$

Until this point we have considered the convection-diffusion equation Eq. (1) without time, or in mathematical terms with $\frac{\partial u}{\partial t} = 0$. Due to both the absence of time and the fact that u_c was either taken constant or equal to u, the Jacobian matrix of the left-hand side of Eq. (4) could be computed. Therefore the equation could be solved using Newton's method as described on pages 108-109 of [4].
To determine whether the Lagrangian method or Spectro Consistent method is better, Newton's method delivered pleasing results and was fast. The boundary problem Eq. (2) has been used to show that the Spectro Consistent approach is always stable and the Lagrangian approach is not.
In order to be able to simulate real-life problems, we have to take turbulence into account. Turbulence however is a time dependent phenomena. Therefore in order for us to get a feeling for turbulence, we have to add time, or in mathematical terms we have to take $\frac{\partial u}{\partial t} \neq 0$.
Adding time leads to the problem that the Jacobian matrix cannot be determined analytically anymore and hence Newton's method cannot be used to solve the differential equation. Thus another method has to be used to solve the problem.
This calls for a finite difference method like Euler's method. This method is described

in [4] at page 341. Other more sophisticated methods like Adam's method could of course also be used, but we will limit ourselves to the slower but easier implementable Euler method.
In the next section we would like to apply filtering to the convective term. Using Euler's method we will be able to compute the solution to the boundary problem Eq. (1) and still be able to integrate a filter to control the convective term.
Before we will look at the time we will first verify the correctness of the program using Euler's method giving us more insight in the convection-diffusion problem.

2.2.1 Verification of the correctness of the program using Euler's method

As has been mentioned we will verify whether or not the program, which now uses Euler's method instead of Newton's method, provides us with correct answers. With Euler's method we can not only compute the solution of the system in time for a given time-interval, but we can also compute a steady-state solution which means a solution with $\frac{\partial u}{\partial t} = 0$. We will now shortly explain the Euler method and how this method can be used to create a steady-state solution.
The first step we have to take is to rewrite our system Eq. (1) in the following form:

$$\frac{\partial u}{\partial t} = h(u, x, t), \tag{13}$$

with $h(u, x, t) = -u_c \frac{\partial u}{\partial x} + k \frac{\partial u^2}{\partial^2 x} + f(x, t)$.
This is exactly the type of equation the Euler method can be used for. If we take $\frac{u^{(m+1)} - u^{(m)}}{\triangle t}$ to be the approximation for $\frac{\partial u}{\partial t}$, then the Euler method is nothing more than the following:

$$u^{(m+1)} = u^{(m)} + \triangle t \cdot h(u^{(m)}, x, t), \tag{14}$$

where $\triangle t$ is the chosen timestep, $u^{(m+1)}$ and $u^{(m)}$ are two subsequent approximated solutions in the time-dimension and $h(u^{(m)}, x, t)$ is defined as above.
If given a start vector u^0 this method is computed for sufficiently large m, $u^{(m+1)}$ and $u^{(m)}$ will be close to each other, meaning $u^{(m+1)} - u^{(m)} = 0$ and therefore $\frac{\partial u}{\partial t} = 0$. In other words, a steady-state solution has been reached. In this way it is clear that it is possible to generate steady state solutions of Eq. (1).
Knowing this we can proceed to the actual verification. We will verify two things: first we will verify whether the boundary conditions are implemented properly and second we will verify whether the correct solution is computed. In the following two subsections these verifications will be discussed seperately.

2.2.2 Verification: boundary conditions

In order to verify the correctness of the solution of the system Eq. (13), two different problems will be solved. At first we will solve the system $h(u, x, t) = 0$ with $u(0) = 1$ and $u(1) = 2$. This will produce the solution u. Next we will solve the system $h(v + 1, x, t) = 0$ with $v(0) = 0$ and $v(1) = 1$, producing the corresponding solution v.

This substitution $u = v + 1$ leads to the following relations: $\frac{\partial v}{\partial x} = \frac{\partial u}{\partial x}$ and $\frac{\partial v^2}{\partial^2 x} = \frac{\partial u^2}{\partial^2 x}$. Thus the only point at which the systems differ is at the convective term. In the first case it is $u\frac{\partial u}{\partial x}$ and in the second it is $(v+1)\frac{\partial u}{\partial x}$.
For several values of N this comparison has been made and the outcome can be seen in Table 1.

N	C2: $\|\|u-(v+1)\|\|_\infty$	C4: $\|\|u-(v+1)\|\|_\infty$
16	$4.8850{\cdot}10^{-15}$	$2.3759{\cdot}10^{-14}$
32	$1.0103{\cdot}10^{-14}$	$1.7319{\cdot}10^{-14}$
48	$1.0880{\cdot}10^{-14}$	$2.1538{\cdot}10^{-14}$
64	$7.9936{\cdot}10^{-15}$	$5.6621{\cdot}10^{-14}$

Table 1: Verification of adjustable boundary values.

From Table 1 we see that all computations lead to the same solution: The differences are all in the order of 10^{-14}. Since boundaries are not the easiest part of a differential equation to implement, this result is satisfying.

2.2.3 Verification: correctness of the solution

After verifying that the boundaries are treated correctly it is now time to verify the correctness of the entire solution. In order to do this, we will create a test-case that gives us the opputunity to compare the computed solution to a known analytical solution. As has been mentioned before, the solution to Eq. (2) can be determined analytically. If we now can generate a right-hand side such that the steady-state solution to Eq. (12) will also be that very same analytical solution, we can compare the two and verify the correctness of the computed solution.
We can now construct a right-hand side f such that the steady-state solution of Eq. (1) is known. We will do this by chosing f as follows:

$$f(x,t) = f(x) = u_a \frac{\partial u_a}{\partial x} - k\frac{\partial u_a^2}{\partial^2 x}, \tag{15}$$

with u_a the analytical solution to Eq. (2):

$$u_a(x) = \frac{u_N - u_0}{1 - e^{\frac{x_0 - x_N}{k}}} e^{\frac{x - x_N}{k}} + u_0 - e^{\frac{x_0 - x_N}{k}} \frac{u_N - u_0}{1 - e^{\frac{x_0 - x_N}{k}}} \tag{16}$$

In this example we can simplify this equation, since we already know that $x_0 = u_0 = 0$ and $x_N = u_N = 1$. This leads to:

$$u_a(x) = \frac{e^{\frac{x-1}{k}}}{1 - e^{\frac{-1}{k}}} - \frac{e^{\frac{-1}{k}}}{1 - e^{\frac{-1}{k}}} = \frac{e^{\frac{x-1}{k}} - e^{\frac{-1}{k}}}{1 - e^{\frac{-1}{k}}} \tag{17}$$

For $N = 16, 32, 64, 128$ we have computed the solution to the resulting equation and compared the answers to the analytical solution. We have chosen $||u - u_{\mathrm{a}}||_\infty$ to measure the correctness of the solution. u_a is the analytical solution and u the computed solution. This leaves us the following results:

N	C2: $e(N) = ||u - u_a||_\infty$	C4: $e(N) = ||u - u_a||_\infty$
16	$4.0523 \cdot 10^{-2}$	$1.6053 \cdot 10^{-2}$
32	$5.7864 \cdot 10^{-3}$	$2.7521 \cdot 10^{-4}$
64	$9.3933 \cdot 10^{-5}$	$1.9362 \cdot 10^{-5}$
128	$2.8848 \cdot 10^{-5}$	$2.9244 \cdot 10^{-6}$

Table 2: Verification of the correctness of the computed solution.

From this table we can conclude three things. First of all we can confirm what we already new: the C4 method is superior to the C2 method. Secondly we can state that denser grids provide us with more precise answers and third, last and most importantly we can conlude that our program computes correct answers. For the coarse grid of merely 16 gridpoints still a precision of 10^{-2} is obtained and for denser grids the precision increases.

3 Filtering

In the introduction we already mentioned the intention to use a smoothening filter. The reason for us to apply filtering is the following. In order to simulate an accurate solution using a DNS method, small scales of motion (and complementary energy) have to be computed. Hence a dense grid has to be used in order to be able to sufficiently compute small scales of motion.
Small scales of motion are not of interest in real-life examples, however they are necessary to produce an accurate solution. Therefore we propose to apply filtering to the convective term, in order to see whether we can accurately simulate larger scales of motion, whilst ignoring the smaller scales of motion. If we are able to do so, coarser grids can be used which allows us to save on computation time. Furthermore DNS methods applied to the non-linear model appear to be more accurate compared to the linear model.
We will now first suggest a filter, based on the Gauss-Jacobi method, and explain why this filter might be of interest.

3.1 Gauss-Jacobi filter: definition

The filter we are going to use is based on the iterative Gauss-Jacobi method for solving sytems of equations of the form $Ax = b$. This method is fully explained on page 545 of [4]. We will recite the method. First $Ax = b$ is rewritten in the following form:

$$x_i = \frac{1}{a_{ii}}\{b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j\},\ i = 1, 2, \ldots, n \tag{18}$$

All a_{ii} have to be nonzero. Now we can define the iteration as follows:

$$x_i^{(m+1)} = \frac{1}{a_{ii}}\{b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(m)}\},\ i = 1, \ldots, n \text{ and } m \geq 0 \tag{19}$$

Furthermore we have to assume that the starting values $x_i^{(0)}, i = 1, \ldots, n$ are given. This is merely the definition of the Gauss-Jacobi method and we still need to describe in what way we will use it with our filter.
In order to complete the filter the following differential equation is also needed:

$$\hat{u} + \varepsilon^2 \Delta \hat{u} = \tilde{u}, \tag{20}$$

where $\hat{u}$ is the solvant of this equation, $\tilde{u}$ its right-hand side and ε a parameter.
The numerical representation of Eq. (20) is:

$$(\boldsymbol{I} + \varepsilon^2 \boldsymbol{\Omega^{-1}} \boldsymbol{D})\hat{u} = \boldsymbol{u}\ , \tag{21}$$

where $\boldsymbol{u}$ is the discete solution of Eq. (12). Eq. (21) is now solved using Eq. (19). This can be represented in a numerical way as follows:

$$\hat{u}^{(m+1)} = \frac{\boldsymbol{u} - ((\boldsymbol{I} + \varepsilon^2 \boldsymbol{\Omega^{-1}} \boldsymbol{D}) - \mathrm{diag}(\boldsymbol{I} + \varepsilon^2 \boldsymbol{\Omega^{-1}} \boldsymbol{D}))\hat{u}^{(m)}}{\mathrm{diag}(\boldsymbol{I} + \varepsilon^2 \boldsymbol{\Omega^{-1}} \boldsymbol{D})}, \tag{22}$$

where diag() is a function which constructs a matrix which only consists of the diagonal entries of its argument, other non-diagonal entries are zero.
Since $\text{diag}(\boldsymbol{I}) = \boldsymbol{I}$, this equation can be rewritten in a more simple form as follows:

$$\hat{u}^{(m+1)} = \frac{\boldsymbol{u} + \varepsilon^2(\text{diag}(\boldsymbol{\Omega^{-1}D}) - \boldsymbol{\Omega^{-1}D})\hat{u}^{(m)}}{\text{diag}(\boldsymbol{I} + \varepsilon^2\boldsymbol{\Omega^{-1}D})} \tag{23}$$

We have not discussed the parameter ε yet. This parameter makes the filter adjustable and hopefully we can get some insight in how to set this parameter for different gridsizes. This is one way to adjust the filter. The other way is to use a different number of iteration steps of the Gauss-Jacobi method.
Note that the filter is not applied locally, but to an entire vector.
Now that we have defined the filter, we can look at its behaviour.

3.2 Gauss-Jacobi filter: behaviour

After defining the filter we can now look at how the filter behaves. We will verify that the filter smoothens the convective term. In this section we will construct two different examples. The first one will show that the filter does not alter the identity map. And the second one will show whether or not the filter has the ability to make a solution smoother. We will also explain the meaning of "smoother".
In the Gauss-Jacobi filter $\boldsymbol{u}$ is the input and the output is $\hat{u}^{(m+1)}$ for a certain value of m. To show that the identity map is not altered by the filter, two figures have been constructed. In Figure 3 x has been inputted in the filter and the filter has been applied to x for a total of 10 iteration- or filtersteps. This filtered x, $\hat{x}^{(10)}$, is plotted against x itself. And finaly in Figure 4 $\hat{x}^{(100)}$ is plotted against x. Note that $\varepsilon = 0.5$.

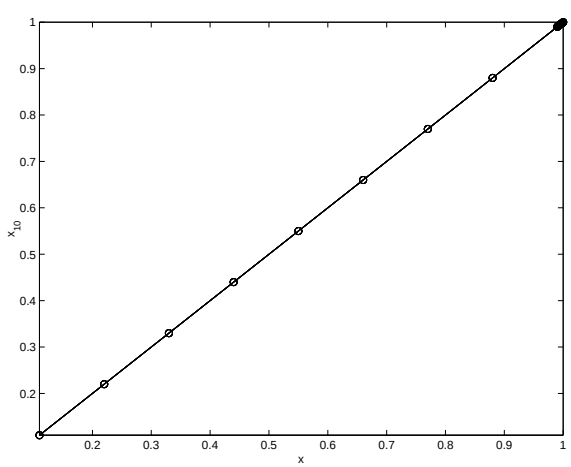

Figure 3: Gauss-Jacobi filter: $\hat{x}^{(10)}$ vs. x.

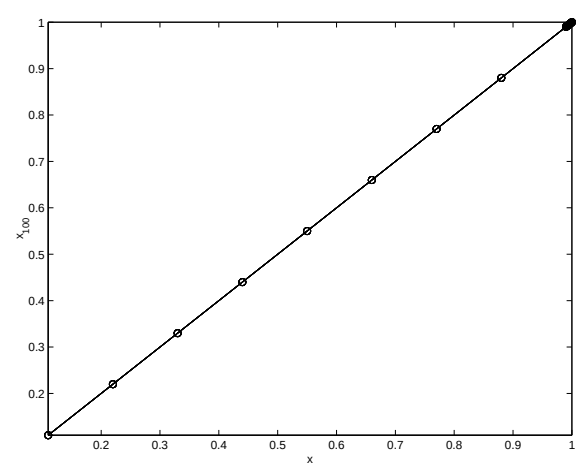

Figure 4: Gauss-Jacobi filter: $\hat{x}^{(100)}$ vs. x.

From these figures we can conclude that both $\hat{x}^{(10)}$ and $\hat{x}^{(100)}$ appear to be indentical. A

more mathematical and conclusive answer can be given by looking at both $||\hat{x}^{(10)} - x||_\infty$ and $||\hat{x}^{(100)} - x||_\infty$. These are both equal to $2.2204 \cdot 10^{-16}$ and therefore it can be concluded that the identity map is preserved by the filter. It is also important to note that as a result of how the boundary is treated and the choice of input, it makes no difference whether we use the C2 or the C4 version of the filter. In our example we have used the C2 version and in the next we will use this version as well.
In the second example we have constructed we will look at the solution u to Eq. (11). We choose to use the solution for $N = 32$ and $k = 0.001$. We will now create four figures. In the first figure we will merely plot u against x. The other three figures will consist of $\hat{u}^{(j)}$ with $j = 10, 100$ and 1000 plotted against x, where $\hat{u}^{(j)}$ is constructed from u in the same way as the $\hat{x}^{(j)}$ from above were constructed from x. To indicate the effect of the filter in the latter three figures also u against x is plotted (thin line). This results in the following figures:

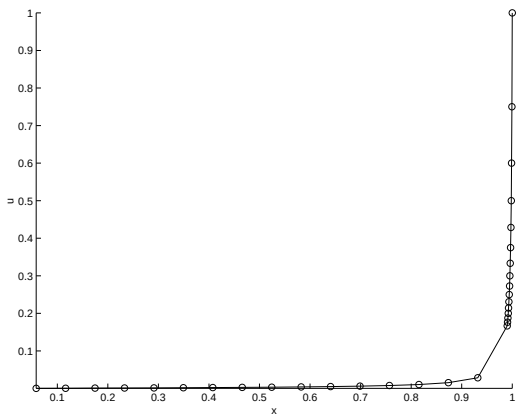

Figure 5: Gauss-Jacobi filter: u vs. x.

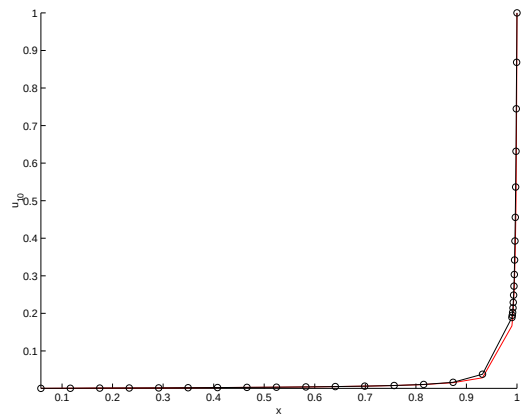

Figure 6: Gauss-Jacobi filter: $\hat{u}^{(10)}$ vs. x.

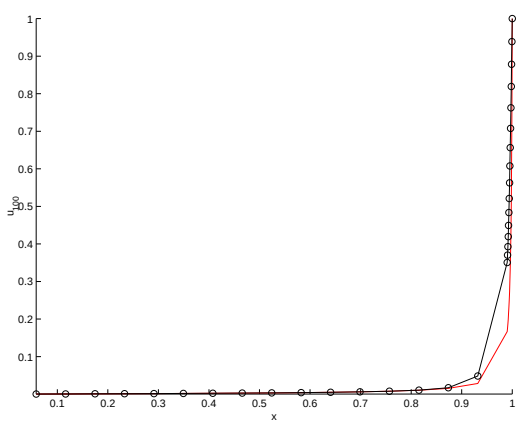

Figure 7: Gauss-Jacobi filter: $\hat{u}^{(100)}$ vs. x.

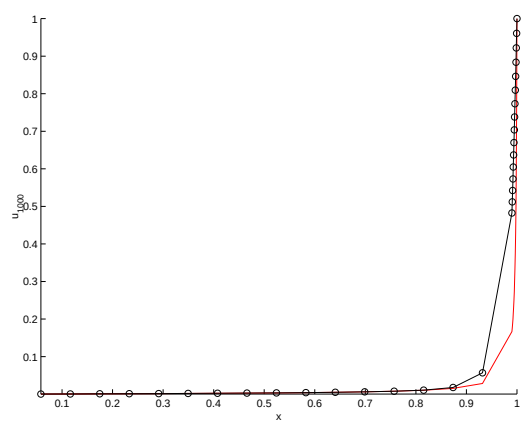

Figure 8: Gauss-Jacobi filter: $\hat{u}^{(1000)}$ vs. x.

It is clear that the more filter steps are applied to u, the "higher" the solution becomes, especially at the right boundary. But in this case higher also means smoother. The thin boundary that is created in the differential equation Eq. (11) at the right boundary x_N is now flattened out somewhat. In other words the slope of the right side of the figures becomes less steep and thus the solution becomes smoother.
An important note we have to make is the following. When we are filtering, in essence we are solving the differential equation Eq. (20) for a certain ε. This means that if we take a large enough number of filter steps, the solution of this equation will be reached and would not change any further if we were to take more filter steps. This means that there is a limit to the extent of how much we can smoothen the solution.
Another note that deserves mentioning is the fact that if ε increases, the filter will smoothen the solution further to the left. This is illustrated in Figure 9 and Figure 10 where ε is respectively chosen to be 2 and 100 and the number of filter steps in both cases is 10.

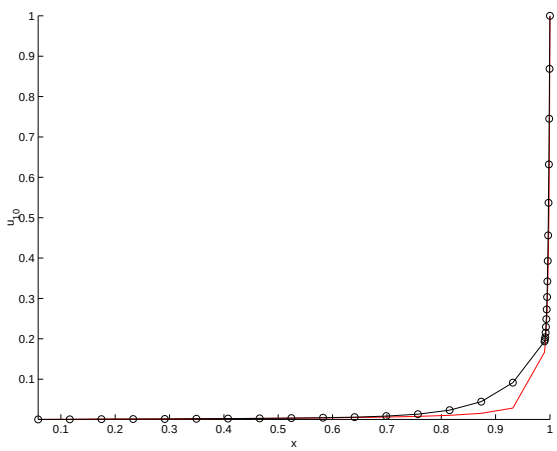

Figure 9: Gauss-Jacobi filter: $\hat{u}^{(10)}$, $\varepsilon = 2$

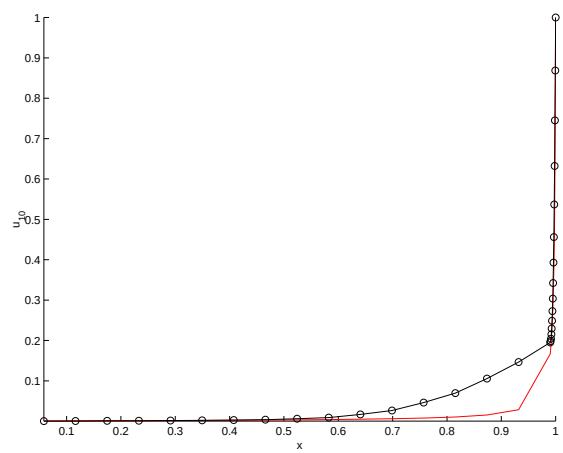

Figure 10: Gauss-Jacobi filter: $\hat{u}^{(10)}$, $\varepsilon = 100$

This behaviour of the solution smoothening more to the left for larger ε also has a limit as to the amount the smoothening travels to the left. This behaviour can be explained mathematically if we look more closely at the solution to Eq. (20).
It is easy to show that the solution $\hat{u}$ to Eq. (20), with boundary values $u(0) = 0$ and $u(1) = 1$ is as follows:

$$\hat{u} = \frac{\sin(\frac{x}{\varepsilon})}{\sin(\frac{1}{\varepsilon})} \tag{24}$$

If ε is chosen large enough $\sin(\frac{1}{\varepsilon}) \approx \frac{1}{\varepsilon}$. Hence the solution can be simplified to:

$$\hat{u} = \frac{\sin(\frac{x}{\varepsilon})}{\sin(\frac{1}{\varepsilon})} \approx \frac{\frac{x}{\varepsilon}}{\frac{1}{\varepsilon}} = x \tag{25}$$

For $\varepsilon > 1$ this effect can already be noticed. This implies that the solution evolves to a straight line. If we only use a small number of filter steps the solution has to move towards this straight line. In other words it has to become smoother on the entire grid. The larger we chose ε the faster this happens due to the sine approximation. This means that for larger ε the smoothening effect appears also more to the left of the grid in stead of merely in the steeper right boundary layer as opposed to smaller ε. The theory explains the smoothening effect as we have seen in the examples.

3.3 Steady-state filtering

In this section we will first of all look at the effect of filtering when we are trying to solve Eq. (2). We will solve this problem with our program using the aforementioned Euler-method. As a next step we will also try to solve Eq. (1) using the right-hand side Eq. (15) and as before we will look at the effect of filtering.
Since both problems generate the same, analytically known, solution, this comparison can easily be made. Note that the only real difference between these problems is the difference in the convective part of the equation. In the first case we merely have $\frac{\partial u}{\partial x}$, whereas in the second case the term complete convective term $u\frac{\partial u}{\partial x}$ is computed. Note that in the latter case u is not taken to be equal to 1 as opposed to the first case.

3.3.1 Steady-state filtering: $\frac{\partial u}{\partial x}$

For $N = 24$ we have ran the program using respectively 1, 2 and 10 filter-steps for both the C2 and C4 method. For ε we chose 0, 0.001, 0.003, 0.005, 0.01, 0.03, 0.05, 0.1, 0.3, 0.5, 1, 3 and 5. This results in the following 6 figures.

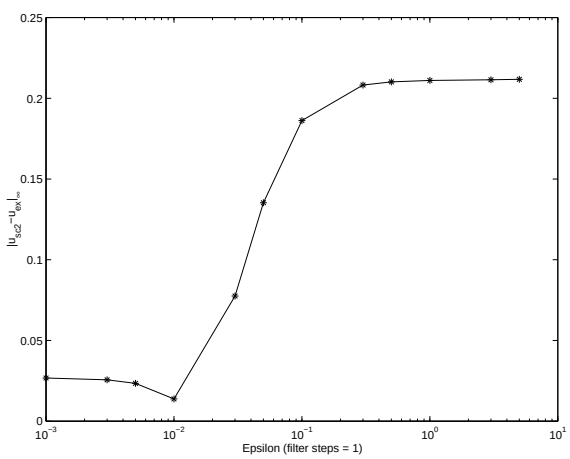

Figure 11: Steady-state filtering: error vs. ε; $N = 24$, C2 method, filter-steps = 1

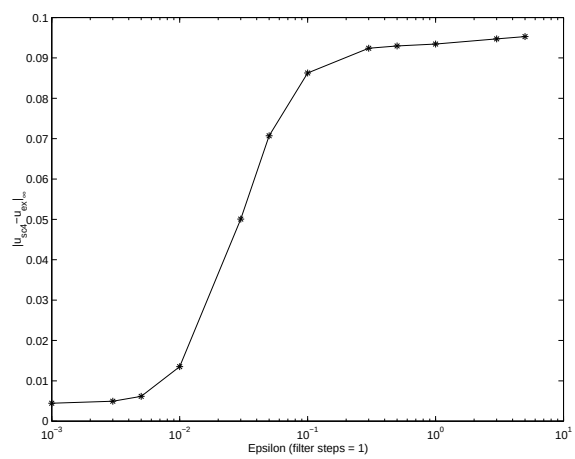

Figure 12: Steady-state filtering: error vs. ε; $N = 24$, C4 method, filter-steps = 1

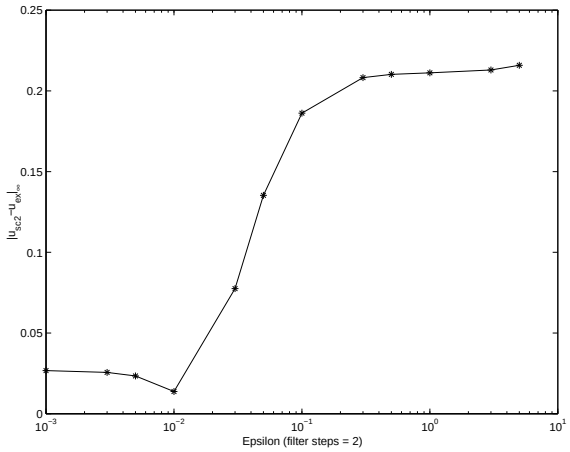

Figure 13: Steady-state filtering: error vs. ε; $N = 24$, C2 method, filter-steps = 2

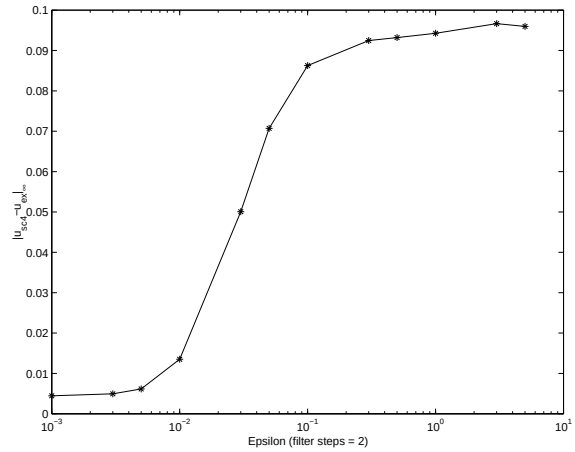

Figure 14: Steady-state filtering: error vs. ε; $N = 24$, C4 method, filter-steps = 2

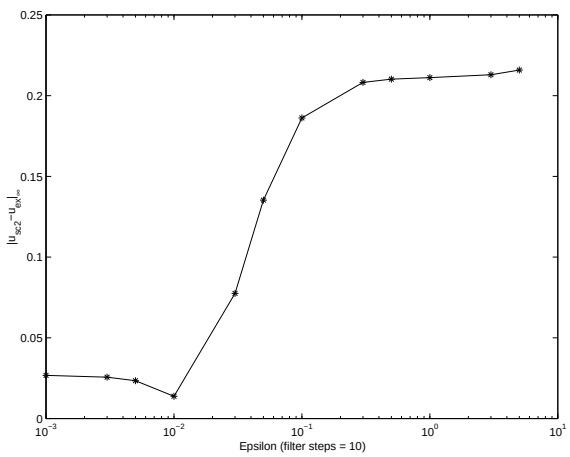

Figure 15: Steady-state filtering: error vs. ε; $N = 24$, C4 method, filter-steps = 10

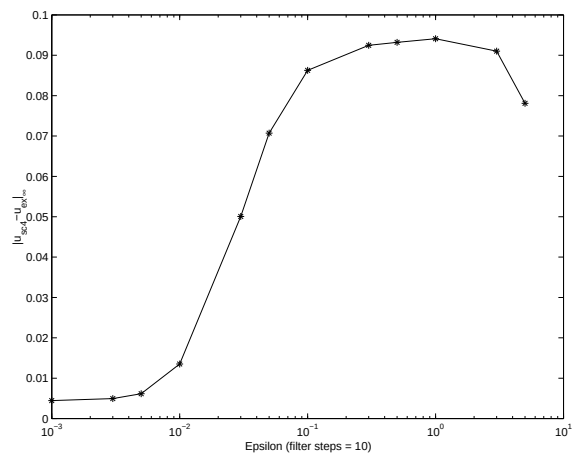

Figure 16: Steady-state filtering: error vs. ε; $N = 24$, C4 method, filter-steps = 10

From these figures we can see that filtering in general worsens the precision of both methods. For the C2 method we can see that a marginal improvement can be achieved if the right value is chosen, however this value is strongly dependent of the parameters of the problem.

3.3.2 Steady-state filtering: $u\frac{\partial u}{\partial x}$

Again the stead-state solution will be computed for $N = 24$ using both the C2 and C4 method. Using the same filter-steps as above, 1, 2 and 10, the steady-state solution has been computed only this time the convective term was not set to 1, but the right-hand

side has been been chosen in such a way that the exact solution to the problem is the same as though the convective term had been chosen equal to 1.
From this we can see if this approach, taking the convective term into account, leads to better results then setting omiting the convective term as we have done in the previous paragraph. The following figures have been created based on the computed solutions.

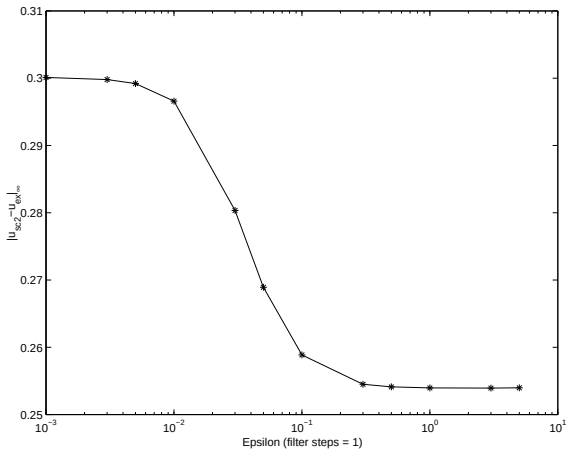

Figure 17: Steady-state filtering: error vs. ε; $N = 24$, C2 method, filter-steps = 1

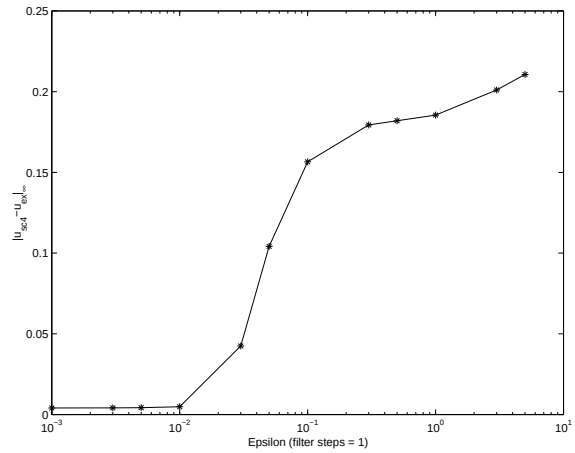

Figure 18: Steady-state filtering: error vs. ε; $N = 24$, C4 method, filter-steps = 1

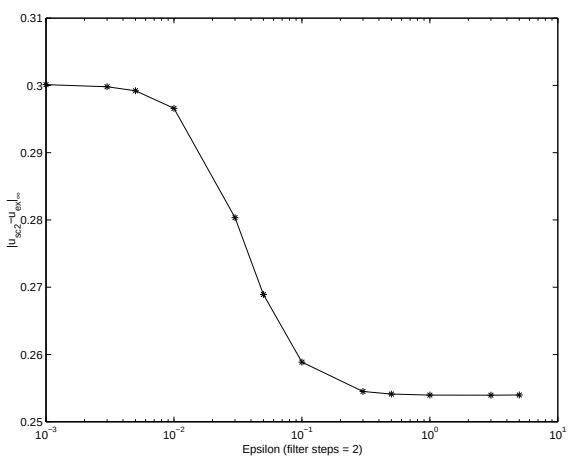

Figure 19: Steady-state filtering: error vs. ε; $N = 24$, C2 method, filter-steps = 2

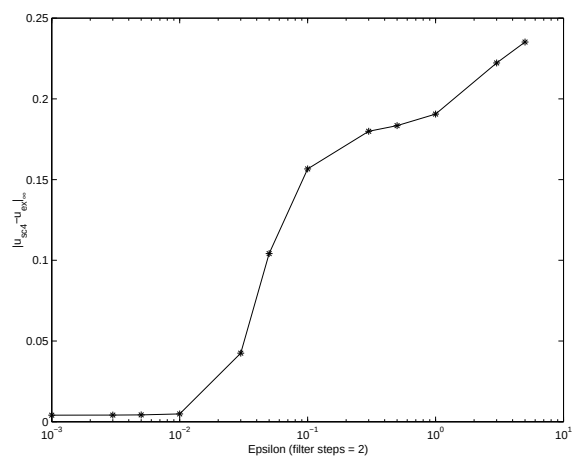

Figure 20: Steady-state filtering: error vs. ε; $N = 24$, C4 method, filter-steps = 2

From these figures we can conclude that the C2 method leads to different results then the C4 method. The latter produces results similar to the previous paragraph, whereas

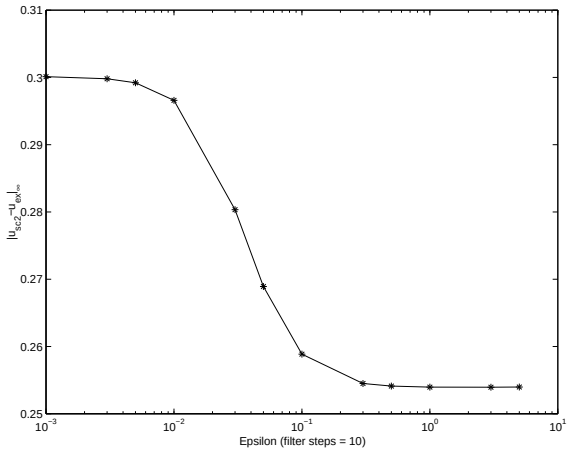

Figure 21: Steady-state filtering: error vs. ε; $N = 24$, C4 method, filter-steps = 10

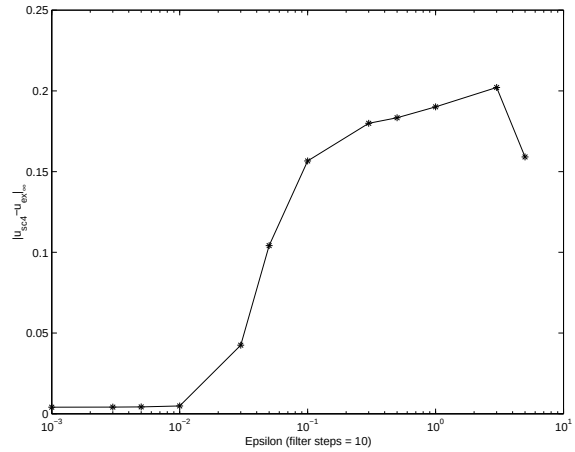

Figure 22: Steady-state filtering: error vs. ε; $N = 24$, C4 method, filter-steps = 10

the C2 method does seem to improve when applying filtering. However this improvement is still marginal and is highly dependent of the number of gridpoints and of the method used to compute the time-component, or in this case the steady-state component.

3.4 Steady-state filtering: conclusion

In computing the steady-state solution to our problem, filtering has little to no effect on the accuracy of the solution. In several cases one of the two methods we used, either C2 or C4, can produce slightly more accurate results, both no real beneficial or structural improvements can be observed.

As we noted, the filtering proces we chose involved filtering an entire vector. A different approach is to filter locally: to modify the value in a point by also taking the values of its neighbours into account. Furthermore, we can also look at Burgers' equation in spectral space. In this way we can look at the energy of different modes.

4 Restraining subgrid-scales in 1D Burgers' equation

As has been anounced at the end of the previous section, in this section we will look at Burgers' equation. More specifically we will look at the energy of Burgers equation in Fourier space and apply a filter in order to restrain the energy at subgrid-scales.
As has been stated in the previous section, we want to limit the computation of the smallest scales and still be able to simulate the energy caused by convection and diffusion accurately among the largest of the remaining scales.
We shall see that by filtering carefully, we can actually restrain subgrid-scales and compute larger scales accurately. Furthermore we will show that for different types of problems the energy will show a pattern.
At first the 1D Burgers' equation in spectral space and the filtering methods are introduced. Afterwards we will discuss the results.

4.1 1D Burgers' equation in spectral space

Since we are going to transform Burgers' equation to spectral space, we will introduce new notation to represent Burgers' equation. In one dimension Burgers' equation reads:

$$\frac{\partial u}{\partial t} + \mathcal{C}(u,u) = \mathcal{D}(u) \ , \tag{26}$$

where $\mathcal{C}(u,v) = u\frac{\partial v}{\partial x}$ and $\mathcal{D}(u) = \frac{\partial u^2}{\partial^2 x}/Re$ and Re is the Reynolds number. Eq. (26) is considered on an interval with periodic boundary conditions.

Henceforth in Fourier space Burgers' equation reads:

$$\frac{\partial \hat{u}_k}{\partial t} + \mathcal{C}_k(\hat{u},\hat{u}) = -(k^2/Re)\hat{u}_k + F_k \ , \tag{27}$$

where $\hat{u}_k$ is the k-th Fourier coefficient of $u(x,t)$.
F_k is a forcing term which has been added in order to obtain non-trivial solutions. The forcing term is only applied at the first scale $k=1$. For $k=1$ the forcing term is taken such that $\frac{\partial \hat{u}_1}{\partial t} = 0 \forall t$ and $F_k \equiv 0$ for $k > 1$.
Furthermore

$$\mathcal{C}_k(\hat{u},\hat{u}) = \sum_{\substack{p=1 \\ p+q=k}}^{k_c} \hat{u}_p i p \hat{u}_q \ , \tag{28}$$

where k_c is the cut-off wavenumber of the numerical solution. All interactions between modes $\hat{u}_p$ and $\hat{u}_q$ with $p+q=k$ and $p=1,\ldots,k_c$ are captured in this term.
For further details regarding the numerical representation of Burgers' equation in Fourier space we refer to [6] and [7].

4.1.1 Evolution of enstrophy

If ω denotes the enstrophy of Burgers' equation in physical space, Eq. (26), then in Fourier space according to [6] the evolution of the enstrophy is given by:

$$\frac{\partial \hat{\omega}_k \hat{\omega}_k^*}{\partial t} = -(2k^2/Re)\hat{\omega}_k\hat{\omega}_k^* - ik(\hat{\omega}_k^*\mathcal{C}_k(\hat{u},\hat{u}) - \hat{\omega}_k\mathcal{C}_k(\hat{u},\hat{u})^*) \ , \tag{29}$$

with $\hat{\omega}_k = ik\hat{u}_k$.

The convection-diffusion equation and Burgers' equation are both connected to the Navier-Stokes equations. As is stated in [7], for the Navier-Stokes equations, an increase in enstrophy will lead to the production of smaller and smaller scales of motion. This will lead to a point where these small scales can not be represented anymore on the chosen computational grid.
For Burgers' equation the evolution of enstrophy at the smallest scale is:

$$\frac{\partial \hat{\omega}_{k_c} \hat{\omega}_{k_c}^*}{\partial t} = -(2k_c^2/Re)\hat{\omega}_{k_c}\hat{\omega}_{k_c}^* - ik_c(\hat{\omega}_{k_c}^*\mathcal{C}_{k_c}(\hat{u},\hat{u}) - \hat{\omega}_{k_c}\mathcal{C}_{k_c}(\hat{u},\hat{u})^*) \ , \tag{30}$$

as has been stated, k_c is the cut-off wavenumber of the numerical solution. On a uniform grid with spacing h this means that $k_c = \pi/h$.
In order for the enstrophy not to grow

$$\frac{\partial \hat{\omega}_{k_c} \hat{\omega}_{k_c}^*}{\partial t} \leq 0 \tag{31}$$

is required.

In the next section we will take a look at a method to restrain the enstrophy, by applying a filter to the non-lineair term $\mathcal{C}_k(\hat{u},\hat{u})$.

4.2 Restraining method

In analogy to section 3 we will approximate $\mathcal{C}_k(\hat{u},\hat{u})$ by $\mathcal{C}_{4,k}(\hat{u},\hat{u})$:

$$\mathcal{C}_{4,k}(\hat{u},\hat{u}) = \sum_{\substack{p=1 \\ p+q=k}}^{k_c} f(\widehat{G}_k, \widehat{G}_p, \widehat{G}_q)\hat{u}_p ip\hat{u}_q \ , \tag{32}$$

where $\widehat{G}_k$ denotes the Fourier transform of the kernel of the convolution filter and

$$f(\widehat{G}_k, \widehat{G}_p, \widehat{G}_q) = \widehat{G}_k(\widehat{G}_p + \widehat{G}_q) + \widehat{G}_p\widehat{G}_q(1 - 2\widehat{G}_k) \ , \tag{33}$$

In accordance to [6] $f(\widehat{G}_k, \widehat{G}_p, \widehat{G}_q)$ is a monotone function of $\widehat{G}_k, \widehat{G}_p$ and $\widehat{G}_q$ and reduces every nonlinear interaction.
Since the value of $f(\widehat{G}_k, \widehat{G}_p, \widehat{G}_q)$ is dependent on p and q, the terms in the summation

in the right-hand side of Eq. (32) are damped differently. If a filter can be constructed which is almost independent of p and q (for $p+q=k_c$), the filter can be applied outside the summation which decreases the time necessary to compute the filter. The filter will be merely dependant of $\widehat{G}_{k_c}$, thus

$$\mathcal{C}_{4,k_c}(\hat{u},\hat{v}) \approx \tilde{f}(\widehat{G}_{k_c})\ \mathcal{C}_{k_c}(\hat{u},\hat{v}) \tag{34}$$

$\tilde{f}(\widehat{G}_{k_c})$ can be computed by setting the evolution of the enstrophy of mode k_c to 0, equation Eq. (30), and by replacing $\mathcal{C}_{k_c}$ by $\mathcal{C}_{4,k_c}$:

$$\tilde{f}(\widehat{G}_{k_c}) = \frac{2ik_c\hat{\omega}_{k_c}\hat{\omega}^*_{k_c}}{Re(\hat{\omega}^*_{k_c}\mathcal{C}_{k_c}(\hat{u},\hat{u}) - \hat{\omega}_{k_c}\mathcal{C}_{k_c}(\hat{u},\hat{u})^*)} \tag{35}$$

We will now discuss the choice of the filter for the restraining method.

4.2.1 Choice of the filter for the restraining method

In this section we will denote the filter which has been used for the restraining method. For the analysis on how the filter was chosen, we refer to [7]. After some calculation as proposed in [7] using the relation $\hat{\omega}_k = ik\hat{u}_k$, $\tilde{f}(\widehat{G}_{k_c})$ can be determined:

$$\tilde{f}(\widehat{G}_{k_c}) = 1 - \sqrt{c}\ , \tag{36}$$

with

$$c = \frac{2k_c^2\hat{u}_{k_c}\hat{u}^*_{k_c}}{Re(\hat{u}^*_{k_c}\mathcal{C}_{k_c} - \hat{u}_{k_c}\mathcal{C}^*_{k_c})} \tag{37}$$

Note that $c = \tilde{f}(\widehat{G}_{k_c})$, with $\tilde{f}(\widehat{G}_{k_c})$ as mentioned in Eq. (35).
Filtering is merely applied if $0 \leq c \leq 1$. This condition coincides exactly with the condition mentioned at Eq. (31). For $0 \leq c < \frac{1}{2}$ a three point filter in physical space is taken and for $\frac{1}{2} \leq c \leq 1$ a five point filter in physical space is taken. This transfers to the following filter in spectral (Fourier) space:

$$\tilde{f}(\widehat{G}_k) = -\widehat{G}_k^2 + 2\widehat{G}_k \qquad k = 1 \ldots k_c\ , \tag{38}$$

where $\widehat{G}_k = c_0 + 2c_1\cos(kh) + 2c_2\cos(2kh)$. In this formula h is the grid size on a uniform grid in physical space and $k_c = \frac{h}{\pi}$.
For the three point filter c_0, c_1 and c_2 are chosen as follows:

$$c_2 = 0 \tag{39}$$

$$c_1 = \frac{1-\widehat{G}_{k_c}}{4} \tag{40}$$

$$c_0 = \frac{1+\widehat{G}_{k_c}}{2}\ , \tag{41}$$

and for the five point filter as follows:

$$c_2 = \frac{1 - 3\widehat{G}_{k_c} + 2\widehat{G}_{k_c}^2}{16(1 + 2\widehat{G}_{k_c})} \tag{42}$$

$$c_1 = \frac{1 - \widehat{G}_{k_c}}{4} \tag{43}$$

$$c_0 = -2c_2 + \frac{1 + \widehat{G}_{k_c}}{2} \ , \tag{44}$$

4.2.2 Filter condition in discrete time

If we integrate Burgers' equation in time using a forward Euler scheme in analogy to [6], the discrete time evolution of the energy is given by

$$\frac{[\hat{u}_k]^{n+1}[\hat{u}_k^*]^{n+1} - [\hat{u}_k]^n[\hat{u}_k^*]^n}{\delta t} = [\hat{u}_k^*]^n[W_k]^n + [\hat{u}_k]^n[W_k^*]^n + \delta t[W_k]^n[W_k^*]^n \ , \tag{45}$$

where n and $n+1$ respectively denote the old and new time levels, δt the timestep and $W_k = -\mathcal{C}_{4,k}(\hat{u}, \hat{u}) - \frac{k^2}{Re}\hat{u}_k$. We propose to alter the filter condition using this discrete evolution.

$$c = \frac{-a_2 + \sqrt{a_2^2 - 4a_1a_3}}{2a_1} \ , \tag{46}$$

with

$$a_1 = \delta t \mathcal{C}_{k_c}\mathcal{C}_{k_c}^* \tag{47}$$

$$a_2 = (\delta_t \frac{k_c^2}{Re}) - 1)(\hat{u}_{k_c}\mathcal{C}_{k_c}^* + \hat{u}_{k_c}^*\mathcal{C}_{k_c}) \tag{48}$$

$$a_3 = -\frac{k_c^2}{Re}(2 - \delta t \frac{k_c^2}{Re})\hat{u}_{k_c}\hat{u}_{k_c}^* \ , \tag{49}$$

In the next section we will compare the results without filtering to both filtering methods.

4.3 Results

In this section we first will show and discuss whether or not filtering has a positive effect to determine the correct solution after which we will show and discuss the difference between the two different filtering methods as mentioned above.
In analogy to [7] we have chosen to solve Burgers' equation for $Re = 50$. As initial condition $\hat{u}_k = \frac{1}{k}$ and a time step $\delta t = 0.001$ have been chosen.

4.3.1 DNS vs regularization method

In this section we will show that the regularization method will conserve energy and henceforth will capture the physics correctly.

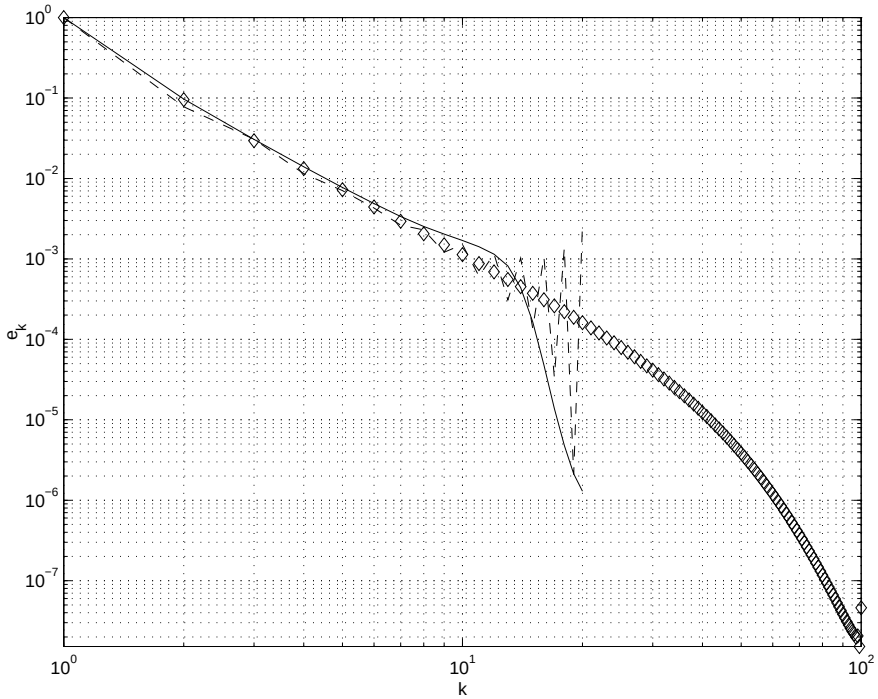

Figure 23: Energy spectrum of the steady-state solution of Burger's equation, with and without filtering, for $k_c = 20$ and $\delta t = 0.001$.

Figure 23 shows the energy spectrum of the steady state for $k_c = 20$ and $\delta t = 0.001$. A DNS spectrum with $k_c = 100$ and $\delta t = 0.0005$ has been added as a reference. It can be concluded that the direct simulation without filtering is not capable of capturing the physics correctly. The energy is not dissipated enough at the higher wavenumbers and is sent back towards lower wavenumbers, resulting in "wiggles".
The regularization model uses energy conservation. Therefore a small hump arises to compensate for the loss of energy near $k_c = 20$.

4.3.2 Continous condition vs discrete condition

In this section we will show the following: if we use the regularization method using c as proposed in Eq. (37) the energy may still grow due to the fact that the continuous condition is discretized which can lead to numerical errors. However the regularization method which uses c based on the evolution of the discrete energy as proposed in Eq. (46) fully conserves energy.

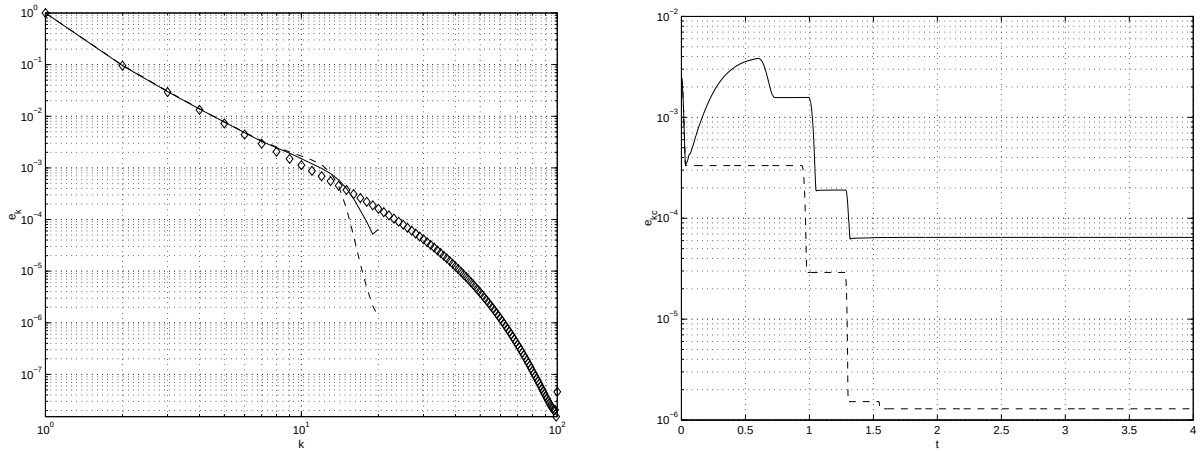

Figure 24: Results using the regularization method with both the continuous and discrete condition, for $k_c = 20$ and $\delta t = 0.001$. Left: see Figure 23. Right: evolution of the energy in time at the highest wavenumber e_{k_c}

The right figure in Figure 24 shows that the regularization method using the discrete condition fully conserves energy at the highest wavenumber k_c, whereas the energy at regularization method using the continuous condition grows in time for certain values of t.

In order to verify whether the regularization method using the discrete condition yields the same results for different values of k_c, the program has also been run for $k_c = 20, 30, 40$ and 50. This results in the following figure:

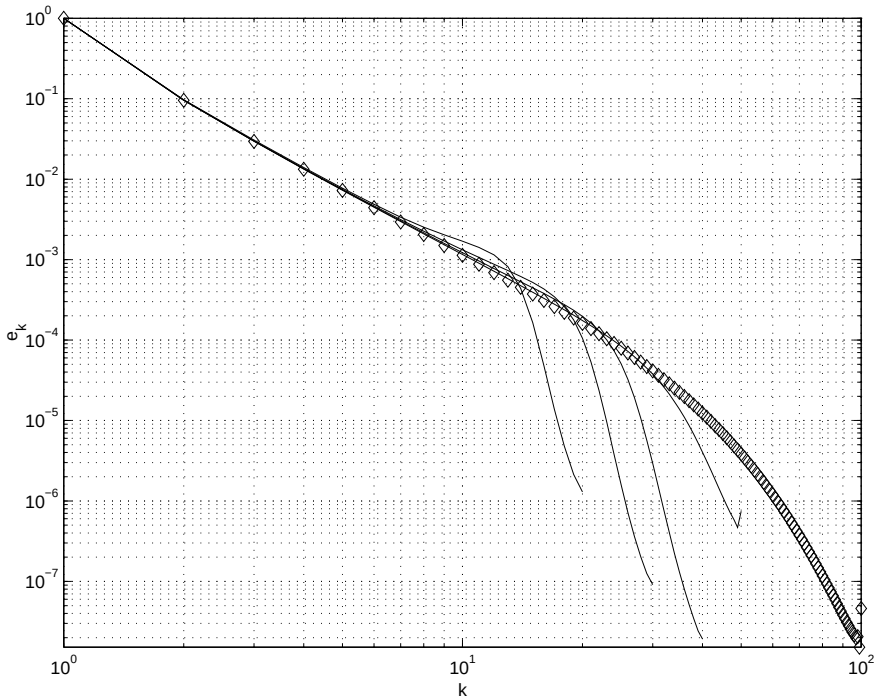

Figure 25: Energy spectrum of the steady-state solution of Burger's equation, with and without filtering, for $k_c = 20, 30, 40$ and 50 and $\delta t = 0.001$.

From this figure can be concluded that the chosen method shows the same pattern for each value of k_c.

4.3.3 Conclusion

From the above we can conclude that for the solution of the one dimensional Burgers' equation the regularization method using the numerical condition from the discrete evolution of energy is able to generate physically correct solutions, at the highest wavenumbers. For smaller wavenumbers the solution is not correct. However the energy shows the same pattern for different values of k_c: after a certain value of k, there is a small rise in energy after which there is a large drop in energy.
From this we can conclude that it might be possible to use the filtered solution to determine which part of this filtered solution we can use to approximate the real, physically correct solution.
In the next section we will take a look at Burgers' equation in two dimensions.

5 Burgers' equation in two dimensions

In analogy to the previous section we will now try to model the 2D version of Burgers' equation in the same manner as the 1D version by looking at the equation in spectral space.
Before we take a look at the equation in spectral space, we will recite the 2D equation in physical space:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \mathcal{C}(\boldsymbol{u}, \boldsymbol{u}) = \mathcal{D}(\boldsymbol{u}) \ , \tag{50}$$

where $\boldsymbol{u} = (u, v)$, $\mathcal{C}(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u} \cdot \nabla)\boldsymbol{v}$ and $\mathcal{D}(u) = \Delta \boldsymbol{u}/Re$. Our approach will be to transfer both the u-equation and the v-equation to spectral space and to compute them seperately. Note that the u-equation and v-equation are coupled through the convective term.
In 2D each dimension can have a different number of modes and the solution will be computed for each combination of these modes. For comparison we will try to obtain the same results in 2D as we did in 1D. Furthermore we will compute the solution for combinations of larger number of modes and look at the energy. We want to find out whether the 2D case is in accordance with the 1D case regarding the patterns the energy show. Furthermore we will discuss the difficulties we encountered in the modelling proces and discuss the results.

5.1 Spectral space

As has been mentioned above, there will be two equation, the u-equation and the v-equation. This will also be the case in spectral space. Furthermore, since we will compute in 2D, there will be modes in two dimensions as well. Henceforth Burgers' equation in 2D reads:

$$\frac{\partial \hat{u}_{k_1k_2}}{\partial t} + \mathcal{C}_{k_1k_2}(\hat{u}, \hat{u}, \partial x_1) + \mathcal{C}_{k_1k_2}(\hat{v}, \hat{u}, \partial x_2) = -\frac{k_1^2 + k_2^2}{Re}\hat{u}_{k_1k_2} + F_{k_1k_2} \tag{51}$$

$$\frac{\partial \hat{v}_{k_1k_2}}{\partial t} + \mathcal{C}_{k_1k_2}(\hat{u}, \hat{v}, \partial x_1) + \mathcal{C}_{k_1k_2}(\hat{v}, \hat{v}, \partial x_2) = -\frac{k_1^2 + k_2^2}{Re}\hat{u}_{k_1k_2} + F_{k_1k_2} \ , \tag{52}$$

where

$$\hat{u} = \sum_{k_1}^{k_{c_1}} \sum_{k_2}^{k_{c_2}} u_{k_1k_2} e^{ik_1x_1} e^{ik_2x_2} \ , \tag{53}$$

and $F_{k_1k_2}$ is a forcing term which has been added in order to obtain non-trivial solutions. The forcing term is only applied at the first scales $k_1 = 1$ and $k_2 = 1$. For $k_1 = k_2 = 1$ the forcing term is taken such that $\frac{\partial \hat{u}_1}{\partial t} = 0 \forall t$ and $F_{k_1k_2} \equiv 0$ for $k_1, k_2 > 1$.
Furthermore

$$\mathcal{C}_{k_1k_2}(\hat{u}, \hat{v}, \partial x_j) = \sum_{\substack{p_1=1 \\ p_1+q_1=k_1}}^{k_{c_1}} \sum_{\substack{p_2=1 \\ p_2+q_2=k_2}}^{k_{c_2}} \hat{u}_{p_1p_2} i p_j \hat{v}_{q_1q_2} \quad j = 1, 2 \ , \tag{54}$$

where k_{c_1} and k_{c_2} are the cut-off wavenumbers of the numerical solution. All interactions between modes $\hat{u}_{p_1 p_2}$ and $\hat{u}_{q_1 q_2}$ with $p_j + q_j = k_j$ and $p_j = 1, \ldots, k_{c_j}$ for $j = 1, 2$ are captured in this term.

One of the difficulties that arose is the following: suppose $k_1 < p_1 \leq k_{c_1}$ for a certain value of k_1. Then by definition $q_1 < 0$. This implies that in computing at time $t+1$, $u^{t+1}(k_1, k_2)$, we need $u^t(p_1, p_2)$ and $u^t(q_1, q_2)$ at the old time step t. If $q < 0$ in one dimension we would take the complex conjugate of $-q$, since

$$u(-q)^* \Rightarrow (e^{-iqk})^* = e^{iqk} \Rightarrow u(q) \ , \tag{55}$$

where * denotes the complex conjugate operation.
In two dimensions however if we take the complex conjugate of a fourier component, we get:

$$u(-q_1, q_2)^* \Rightarrow (e^{-iq_1x_1} e^{iq_2x_2})^* = e^{iq_1x_1} e^{-iq_2x_2} \Rightarrow u(q_1, -q_2) \tag{56}$$

If both q_1 and q_2 are negative this suffices, since in that case for q_2 we can also use $-q_2$. If however q_1 is negative and q_2 positive, we have to correct the left-hand side of Eq. (56) by $e^{2iq_2k_2}$ to compensate for the complex conjugate being taken over the product of both components:

$$u(-q_1, q_2)^* \Rightarrow e^{2iq_2x_2}(e^{-iq_1x_1} e^{-iq_2x_2})^* = e^{2iq_2} e^{iq_1x_1} e^{-iq_2x_2} = e^{iq_1x_1} e^{iq_1x_1} \Rightarrow u(q_1, q_2) \tag{57}$$

If q_1 is positive and q_2 negative an analogues correction has to be applied.

5.2 Results

In this section we will discuss the results which have been obtained solving the 2D Burgers' equation. At first a comparison is made between the 1D and 2D energy spectra for verification and finally a more in depth analysis of 2D energy spectra for various situations are displayed and discussed.

5.2.1 Results: comparison between 1D and 2D energy spectrum

As has been mentioned in the introduction of this section, we will first compare the 1D energy to the 2D energy to see whether we can obtain the same results with our 2D program as with our 1D program. In order to do so, we have set the parameters as follows $k_{c_1} = 0$, $k_{c_2} = 20$, $t = 4$ and $\delta t = 0.001$. In the following figure the diamonds are the values computed using the 1D program and the solid line is the solution computed by the 2D program.

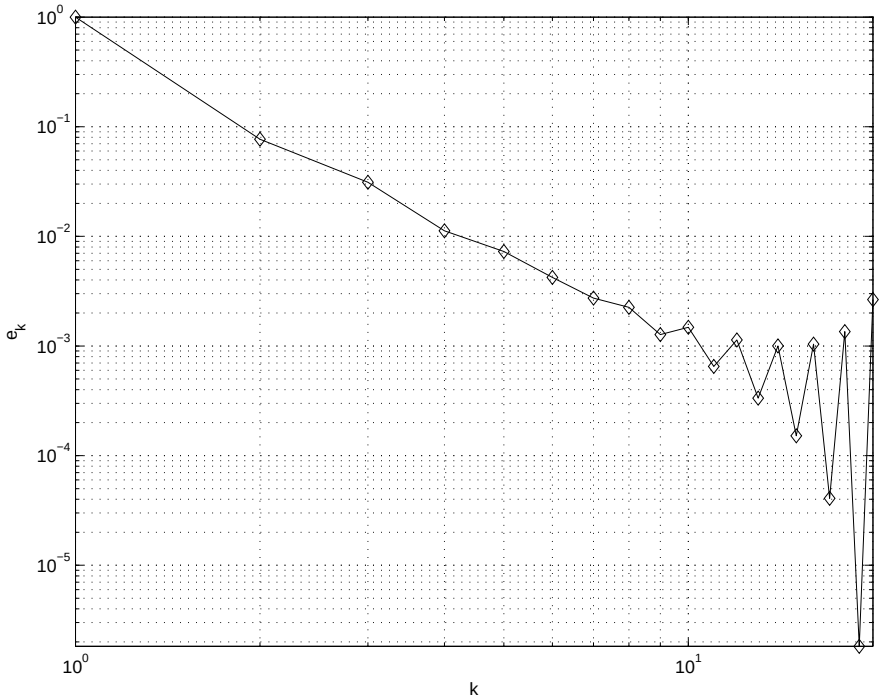

Figure 26: Comparison between the energy spectrum of the 1D and 2D steady-state solution of Burger's equation for $k_{c_1} = 0$ and $k_{c_2} = 20$ and $\delta t = 0.001$.

From these figures and the underlying data it can be concluded that the 2D program with computed in one dimension yields the same results as the 1D program, as expected. We will now look at several two dimensional problems where $k_{c_1} = k_{c_2} > 0$.

5.2.2 Results: 2D energy spectrum in depth

The next step is to compute the 2D program in two dimensions for different parameters and take a look at the energy spectrum. For the energy spectrum we will look at the energy per wavenumber instead of the energy per mode. Note that in one dimension these two are the same. The energy at a wavenumber $\tilde{k}_i$ for $i = 0, \ldots, k_1 + k_2 - 1$ is defined by:

$$e(\tilde{k}_i) = \sum_{k_1+k_2=\tilde{k}_i} e(k_1) + e(k_2) \tag{58}$$

We have computed the energy spectrum for $Re = 2$, $k_{c_1} = k_{c_2} = 10$, $\delta t = 0.001$. We have also tried to compute the energy spectrum for $Re = 4$, $k_{c_1} = k_{c_2} = 20$, $\delta t = 0.0005$. At the end of my research we found out that the 2D program had an error and therefore I aborted the computation of the energy spectrum for the latter set of parameters. From this we also have to conclude that the energy spectrum of the first set of parameters is not exactly the energy of the 2D burgers equation. We will now first show the figure and then discuss the error in the program.

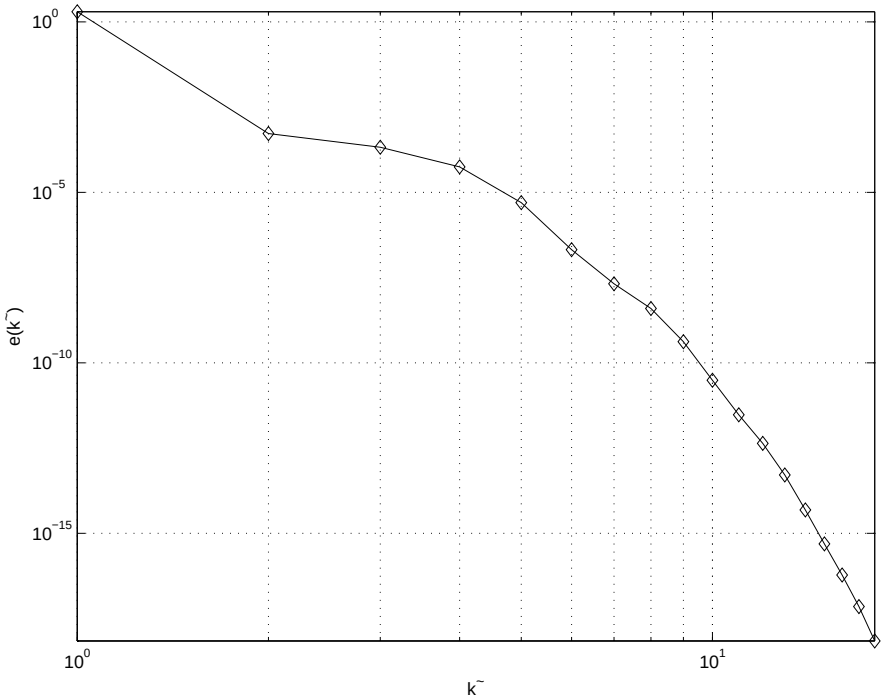

Figure 27: Energy spectrum for the steady-state solution of the 2D Burgers' equation for $Re = 2$, $k_{c_1} = k_{c_2} = 10$ and $\delta t = 0.001$.

We can see from the energy spectrum shown in Figure 27 that the energy is distrubeted amongst all wavenumbers. However we have also tried to compute different solutions for different parameters. For several of these sets of parameters we expected the energy spectrum to show something similar to the energy spectrum shown in Figure 27. These simulations of these energy spectra however were unstable and the solutions "blew up".

The error can be found in an errorneus implementation of the correction we proposed in Eq. (57). The correction has to applied in physical space, hence an inverse Fourier transformation has to be applied to the velocity in spectral space to compute the velocity in physical space. In physical space the correction has to be applied in the appropriate direction. Finally the velocity has to be transformed again to spectral space using a Fourier transformation.
In our case we have applied the correction to the velocity in spectral space. But since the velocity in spectral space is the sum of velocities in physical space, the wrong premultiplication is made. We are now essentially applying some kind of filtering without knowing exactly what it is we are filtering with.
In some cases this might lead to a stable solution, however, the solution is not correct.

5.3 Conclusion

Our 2D program is not able to correctly simulate energy spectra for different sets of parameters, due to an error. We have explained the error and have shown that in some cases the program can still produce stable solutions. The fact that the 2D can accurately simulate a 1D problem, lies within the fact that whilst computing the 1D problem the correction does not have to be applied. In the 1D problem q_1 will not be negative while q_2 is positive and vise versa.
Due to the fact we have not been able to compute a solution on a denser grid, we have not been able to look at convergence or give an impression of how well the 2D program works.

6 Conclusion

In this paper we have looked in several different ways at Burgers' equation. In the first part we have looked at four different Direct Numerical Simulation methods which we used to numerically solve Burgers' equation in one dimension. We have looked at two Lagrangian methods, second and fourth order, and we have looked at two Spectro-Consistent methods, also second and fourth order. One might think that the Lagrangian methods are better due to the way they are constructed, however we have shown that these methods can be instable and should therefore not be used.

On the other hand the Spectro-Sonsistent methods are always stable and preserve continuous proprties like the conservation of energy and mass. From this we can conclude that for solving Burgers' equation in physical space one of the Spectro-Consistent methods should be used.
By the addition of time turbulence was introduced. We have shown that due to the presence of turbulence the DNS methods require the computation of small scales in order to be able to generate an accurate solution. This requires the use of dense grids. In order to find out whether we can accurately compute larger scales without having to compute small scales, we have applied filtering.
At first we have to tried to apply a filter based on the iterative Gauss-Seidel method. Using this method, filtering is applied to an entire vector. While applying this filter, no structured improvements have been observed, neither in speed or in accuracy.
The next stage for us was to look at Burgers' equation in spectral space and by applying a different kind of filter. in spectral space we have applied a filter, a cosine, which in physical space coincides with a local three or five point filter. In computing the velocity at a certain point in space, the values of two or four neighbouring points are also taken into account.
Aplying this filter leads to satisfactory results. Not only are we able to restrain the production of subgrid scales, this method also shows a pattern for different types of problems. This pattern could be used in further research in order to obtain a filtercriterium which can be included in physical space. This could lead to the use of coarser grids whilst still accurately preserving energy at larger scales of motion.

Finally we have extended the problem to two dimensions. The difficulty in spectral lies in the proper handling of the non-linear convective term. The non-linearity leads to a correction which has to be applied. We have suggested a correction, however we have implemented this correction incorrectly in our program. We have discussed this and have concluded that in some cases our program still was able to produce stable energy spectra. However in several cases where we expected stable energy spectra, the program produced unstable solutions. Due to the fact that I ran out of time, I haven't been able to implemet a correct program and therefore I haven't been able to properly compare multiple 2D simulations.

All in all I think that a correct implementation of the 2D will lead to similar results as the 1D program. If it is possible to implement the filter in the same manner as in the 1D program, I feel it is possible to actually restrain the production of subgrid-scales also in two and three dimension. I will gladly leave this challenge for future research.
I am glad that in the end I have continued my research. It has taken a couple of years and I am glad I have finally been able to come up with a paper. Again I would like to thank Roel Verstappen and Joop Helder for assisting me in the final, most straining part of my study.

References

[1] R.W.C.P. Verstappen, A.E.P. Veldman. Turbulentie, golfballetjes en discrete afgeleiden. *NAW*, 5/2 nr. 4, December 2001.
Note: article is written in dutch.

[2] R.W.C.P. Verstappen, A.E.P. Veldman. Numerical Simulation of a Turbulent Flow in a Channel with Surface Mounted Cubes. *Applied Scientific Research*, 59: 395-408, 1998.

[3] R.W.C.P. Verstappen, A.E.P. Veldman. Symmetry-preserving discretization of turbulent flow. *Journal of Computational Physics*, 187: 343-368, 2003.

[4] Kendall E. Atkinson. *An Introduction to Numerical Analysis*, Second Edition. John Wiley & Sons, Inc., 1989.

[5] J. Qian. Numerical experiments on one-dimensional model of turbulence. *Phys. Fluids 27 (8)*, August 1984.

[6] J. A. Helder, R.W.C.P. Verstappen. On restraining subgrid-scale production in Brugers'equation. *International journal for numerical methods in fluids*, 56: 1289-1295, 2008.

[7] J. A. Helder, R.W.C.P. Verstappen. On restraining subgrid-scale production in Brugers'equation (elongated version), 2007.

[8] R.W.C.P. Verstappen. On restraining the production of small scales of motion in a turbulent channel flow. *Computers and Fluids*, doi:10.1016/j.compfluid.2007.01.013, 2007.

[9] A.J.Q. Alkemade. Burgers, Johannes Martinus (1895-1981).
`http://www.inghist.nl/Onderzoek/Projecten/BWN/lemmata/bwn5/burgers`
Biografisch Woordenboek van Nederland, March 2008