The Kuramoto-Sivashinsky equation in Fourier and real space

Andreas Handel

Center for Nonlinear Science and School of Physics,

Georgia Institute of Technology, Atlanta, Georgia 30332-0430

(Dated: December 11, 2001)

Abstract

I investigate the Kuramoto-Sivashinsky equation in Fourier and real space. I give evidence that even though the two representations are completely different, invariant topological quantities such as periodic orbits stay the same and can be extracted from both representations.

I. INTRODUCTION

This project deals with the Kuramoto-Sivashinsky equation. I simulate and investigate this PDE in real space as well as in Fourier space. I give some evidence that both representations lead to the same results regarding topological invariants like periodic orbits - and therefore in each case periodic orbit theory [2] could be used to calculate physical interesting quantities.

For the part of the project that is done in Fourier space, I closely follow [1] and reproduce several of the crucial results presented in that reference. I also correct some inconsistency found in [1]. I carry out the same investigation for the real part representation, and find evidence for periodic orbits that correspond to the periodic orbits found in Fourier space.

II. THE KURAMOTO-SIVASHINSKY EQUATION

The Kuramoto-Sivashinsky (KS) equation is one of the most simple PDEs that allow for chaotic solutions. It is therefore widely studied as a prototype of nonlinear PDEs. It also has physical meaning in its own right, describing interfacial instabilities like flame fronts. There exist several versions of this equation. The one I study here, following [1], has the form

$$u_t = -\nu u_{xxxx} - u_{xx} + (u^2)_x \qquad x \in [-\pi, \pi] \tag{1}$$

Throughout this investigation, $u(x = -\pi, t) = u(x = \pi, t) \; \forall \; t$ holds. Equation (1) has ν as its only parameter, representing a damping factor. One can rescale the system such that $\nu = 1$ and the system size L becomes the free parameter. The relation between ν and L is $\nu = (\frac{2\pi}{L})^2$. Small ν therefore correspond to large system sizes which makes it obvious that the system becomes more turbulent as ν decreases. Another way of noticing that decreasing ν leads to more chaotic/turbulent behaviour is by simply performing linear stability analysis and noticing that a smaller ν leads to more unstable eigenmodes of the system. Following [1], I investigate the system for $\nu = 0.02991$. For this value, the system exhibits weak turbulence.

III. NUMERICS

A. Fourier Mode Approach

One way to simulate the system numerically is to write the PDE as an infinte sum of Fourier modes of the form

$$u(x,t) = \sum_{k=-\infty}^{\infty} b_k(t) e^{ikx} \tag{2}$$

Substituting (2) into (1), we get

$$\dot{b}_k = (k^2 - \nu k^4) b_k - \imath k \sum_{m=-\infty}^{\infty} b_m b_{k-m} \tag{3}$$

So far we have $b_k \in \mathbb{C}$. Following [1] we set $b_k = \imath a_k$, $a_k \in \mathbb{R}$. This chooses the subspace of odd solutions $u(x,t) = -u(-x,t)$ and we get

$$\dot{a}_k = (k^2 - \nu k^4) a_k - k \sum_{m=-\infty}^{\infty} a_m a_{k-m} \tag{4}$$

Since the mean drift a_0 is constant, we can set it to zero $\forall$ t. To simulate the system, we only keep a certain number N of Fourier modes. This leads to the system

$$\dot{a}_k = v_k = (k^2 - \nu k^4) a_k - k \{ \sum_{m=1}^{k} a_m a_{k-m} - \sum_{m=k+1}^{N} a_m a_{m-k} - \sum_{m=1}^{N-k} a_m a_{m+k} \} \tag{5}$$

Where we have also used the fact that $a_k = -a_{-k}$ holds. In [1] the authors checked that $N = 16$ modes suffice to accuratly simulate a system with $\nu = 0.029910$. Since we use the same value for ν, we keep the same number of Fourier modes.

To find periodic orbits, one also needs to integrate the Jacobian J of the system according to the evolution equation

$$\dot{J} = AJ \tag{6}$$

where $A_{kj} = \frac{\partial v_k}{\partial a_j}$ is called (by Predrag) the matrix of variations. From (5) one finds

$$A_{kj} = (k^2 - \nu k^4) \delta_{kj} - 2k (a_{k-j} - a_{k+j}) \tag{7}$$

Again $a_{-k} = -a_k$ holds. The integration of the system and the Jacobian is done by an explicit 4th order Runge Kutta scheme with fixed timesteps of magnitude 1e-5 and 1e-6.

B. Space Discretization Approach

The second approach used to simulate the system is to perform a space discretization and then integrate the resulting system of coupled ODEs. Requiring second order accuracy in space leads to the following discretizations:

$$u_{xxxx} \longrightarrow \frac{1}{(\Delta x)^4}(u_{i+2} - 4u_{i+1} + 6u_i - 4u_{i-1} + u_{i-2}) \tag{8}$$

$$u_{xx} \longrightarrow \frac{1}{(\Delta x)^2}(u_{i+1} - 2u_i + u_{i-1}) \tag{9}$$

$$u_x \longrightarrow \frac{1}{(2\Delta x)}(u_{i+1} - u_{i-1}) \tag{10}$$

Using these discretizations in (1) leads to

$$\frac{\partial u_i}{\partial t} = -\frac{\nu}{\Delta x^4}(u_{i+2} + u_{i-2}) + (\frac{4\nu}{\Delta x^4} - \frac{1}{\Delta x^2})(u_{i+1} + u_{i-1}) + (\frac{2}{\Delta x^2} - \frac{6\nu}{\Delta x^4})u_i + \frac{1}{2\Delta x}(u_{i+1}^2 - u_{i-1}^2) \tag{11}$$

We can write this more compactly as

$$\frac{\partial u_i}{\partial t} = v_i = (Mu)_i + \frac{1}{2\Delta x}(u_{i+1}^2 - u_{i-1}^2) \tag{12}$$

$$u = \begin{pmatrix} u_1 \\ \vdots \\ u_i \\ \vdots \\ u_N \end{pmatrix}, M = \begin{pmatrix} c & b & a & 0 & \ldots & \ldots & 0 & a & b \\ b & c & b & a & 0 & \ldots & \ldots & 0 & a \\ a & b & c & b & a & 0 & \ldots & \ldots & 0 \\ 0 & a & b & c & b & a & 0 & \ldots & \ldots \\ 0 & 0 & a & b & c & b & a & \ldots & \ldots \end{pmatrix} \tag{13}$$

with $a = -\frac{\nu}{dx^4}$, $b = \frac{4\nu}{dx^4} - \frac{1}{dx^2}$ and $c = \frac{2}{dx^2} - \frac{6\nu}{dx^4}$. Again to get the periodic orbits one needs to integrate the Jacobian J of the system according to

$$\dot{J} = AJ. \tag{14}$$

The matrix of variations $A_{kj} = \frac{\partial v_k}{\partial u_j}$ in this case is

$$A = M + \frac{1}{\Delta x}(u_{i+1}\delta_{j,i+1} - u_{i-1}\delta_{j,i-1}) \tag{15}$$

We used a space discretization of $N = 256$ points which corresponds to a $\Delta x = 2\pi/N \approx 0.025$. This means mapping the real space $x = [-\pi, \pi]$ to the discretized space $i = 1 \ldots 256$.

Integrating this system with an explicit method would require very small time steps. An estimate gave that using 4th order Runge Kutta to integrate the system (12), (14) for a time span T≈1sec (the time of the shortest periodic orbit) would take about 4 days.

In order to allow for reasonably large timesteps, we choose to use an fully implicit integrator. This involves using a Newton solver at each timestep. We also tried several semi-implicit methods. We found that a 2nd order semi-implict integrator performed almost as good as the fully implicit one. It unfortunately turned out too late that none of the different integrators tried was able to produce results accurate enough to get a good quantitative result for the periodic orbits (see remarks in IV B).

To show how the equations look like for the semi-implicit 2nd order integrator, we list them here. For the linear part of the euqation we used an implicit second order Adams-Moulton scheme and for the nonlinear part we used an explicit second order Adams-Bashford scheme. The equations to be integrated have the following form

$$u_i^{t+1} = u_i^t + dt(\frac{1}{2}L_i^{t+1} + \frac{1}{2}L_i^t + \frac{3}{2}N_i^t - \frac{1}{2}N_i^{t-1}) \tag{16}$$

and for the Jacobian

$$J^{t+1} = J^t + dt(\frac{1}{2}MJ^{t+1} + \frac{1}{2}MJ^t + \frac{3}{2}N_J^t - \frac{1}{2}N_J^{t-1}) \tag{17}$$

where

$$L_i = (Mu)_i \quad N_i = \frac{1}{2\Delta x}(u_{i+1}^2 - u_{i-1}^2) \quad N_J^t = \frac{1}{\Delta x}(u_{i+1}\delta_{j,i+1} - u_{i-1}\delta_{j,i-1})J^t \tag{18}$$

C. Some Numerics Remarks

Since the whole project was basically about how to set up PDE simulations in a fast and efficient way, I want to make some remarks about numerics which might be helpful for further generations of students who are silly enough to pick a PDE as their object of study.

I started the project using C and MATLAB but then decided to solely use MATLAB. If one tries to code in MATLAB the way one is used to code in C, one notices very soon that MATLAB is much slower. One has to optimize code for MATLAB. To give an example, equation (5), which is the one to be integrated/simulated, can be expressed in matrix/vector notation. The two first sum terms in (5) are then represented by a Toeplitz matrix while the last sum term is a Hankel matrix. In MATLAB one can exploit these special structures of the matrices by making use of the built-in functions 'toeplitz' and 'hankel'. Writing the code this way instead of using standard C type loops, one can gain about a factor of 5 in speed. Also note that matrix (7) is very similar to the matrix one gets on the RHS of (5) if one uses above mentioned matrix/vector notation. Efficient computation reusing the matrix calculated for the flow is therefore possible and again helps to speed up the simulation.

In simulating the equations (16) and (17) for the real space investigation, one can (and has to in order to be able to run the code in a reasonable amount of time) use the sparsity properties of the involved matrices by using MATLABs sparse-matrix functionality.

Even with all this optimization done, MATLAB is somewhat slower than using an optimized Fortran or C routine. But I noticed that the difference is not that huge. For example when simulating the real space equations with C and the additional "matrix library package" PETSc [4], I did not notice a considerable change in execution time. In addition, one can use functionality provided by MATLAB to convert MATLAB code into C code. While I have no experience with this and therefore did not try to use it for this project, I read that this will make the code run at about the speed of a well written C code.

On the other hand, the coding complexity by going from C/PETSc to MATLAB reduces significantly. MATLAB is much easier to code and is therefore the best choice if one has serious time constraints and can't afford spending weeks on debugging code. While I acknowledge that MATLAB might not be suitable for specific large and complex research problems, I am more and more convinced that it is the best overall choice for 'small to medium' type problems.

IV. RESULTS

A. Results for Fourier space simulation

To make sure our Fourier space simlulation works properly, we first reproduce a few figures from reference [1]. Figure 1 shows the result of our code for the same choice of parameteres and variables as in [1]. The figure is the same as figure (2) in [1].

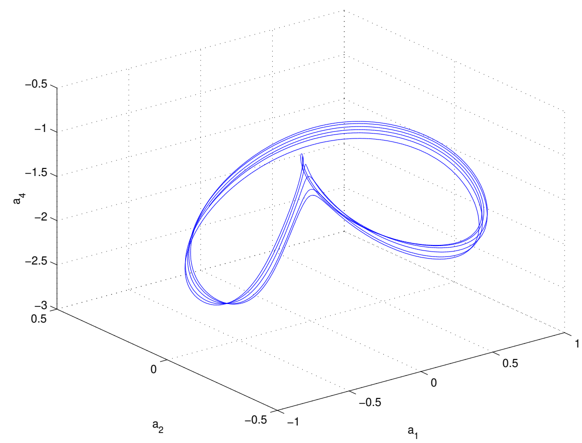

FIG. 1: Fourier modes (a_1, a_2, a_4), $\nu = 0.29910$, $N = 16$

Next we plot the Poincare section $a_1 = 0$ and, as expected, find that it agrees with figure (3) in [1]

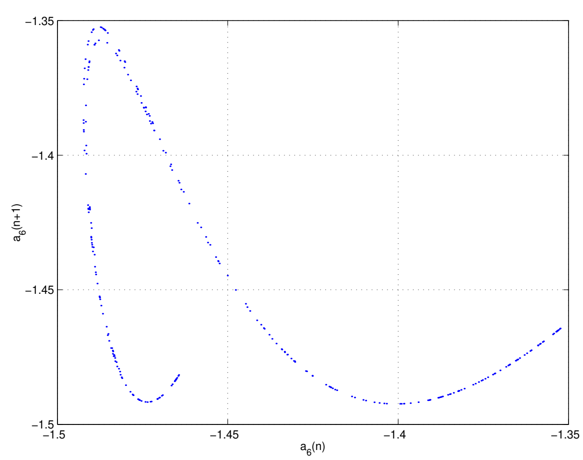

FIG. 2: Poincare section $a_1 = 0$, $\nu = 0.29910$, $N = 16$

Last, we reproduce the Feigenbaum diagram, figure (1) in [1]. Here we find a significant difference between our results and the reference. A closer look at the reference shows that

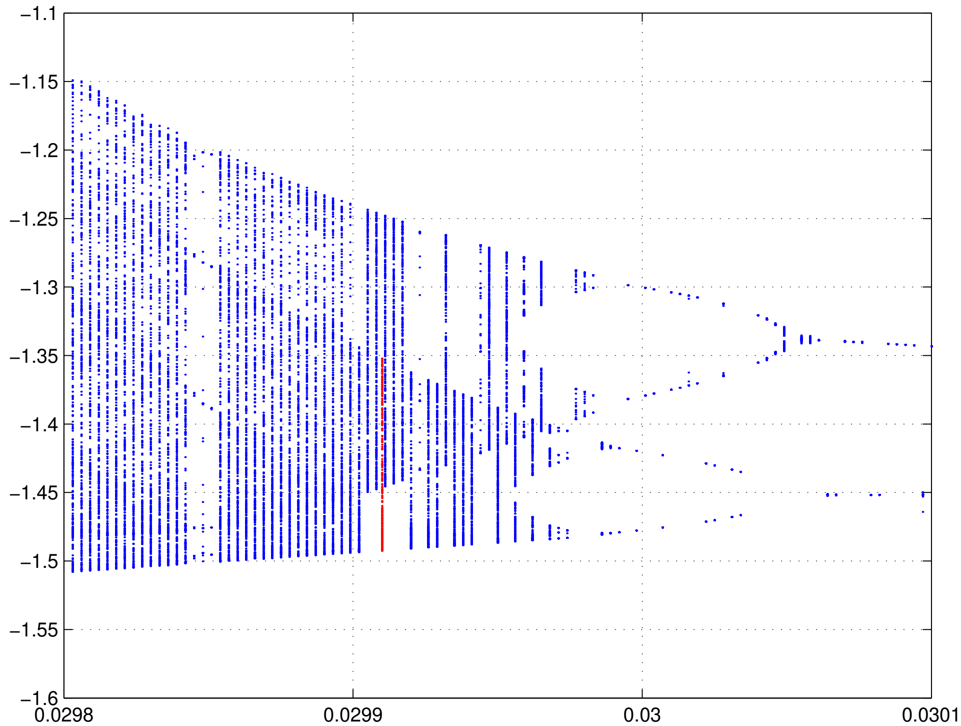

FIG. 3: Feigenbaum diagram in Fourier space for $a_6(\nu)$, $a_1 = 0$, $\nu = 0.29910$, $N = 16$

there is inconsistency between the Feigenbaum diagram (figure 1) and the Poincare section (figure 3). If one looks at $\nu = 0.02991$, given by the left upper arrow in figure (1) of [1], one sees that the values for a_6 lie between about -1.45 and -1.25. But if we now look at figure (2) of our work or correspondingly at figure (3) in the reference, we see that the range of a_6 is approximately between -1.49 and -1.35. This shows that the two figures (1) and (3) of [1] do not agree. The reason for this discrepancy becomes clear if one looks at the Feigenbaum diagram shown in figure (3) above. Different to the technique used by the authors of [1] to get the Feigenbaum diagram ('adiabatic' scanning of parameter space), we choose random initial conditions for each value of ν and then waited until the transients relaxed. This method gives both existing attractors for the system, as can be seen in figure (3). One can see that the upper attractor corresponds to the one in the reference, while the lower attractor is not shown in the reference but it would have been the appropriate one to show since it is the one that contains the Poincare section shown in figure (3) of the reference. Plotting both attractors also shows clearly the merging discussed in the reference. For values of $\nu \gtrapprox 0.0299$ depending on the intial conditions the system will either settle to the upper or lower attractor. At $\nu \approx 0.0299$ the two attractors combine and the system now lives on the attractor which is a combination of the two earlier separate ones.

As a last part of looking at the Fourier representation, we reproduce some of the periodic orbits found in [1]. Since the calculation/simulation to get periodic orbits involves integration of the Jacobian and therefore takes rather long, we have to limit our investigation to finding the periodic orbits with the lowest period. These are the fixed points of the Poincare map. The following table shows the periodic points we found.

TABLE I:

Orbit Symbol	Time	a_6	Λ_1	$\Lambda_2 - 1$	Λ_3	Λ_4
0	0.897654	-1.454001	3.298176	1.126302e-05	-2.793070e-03	-2.793070e-03
1	0.870730	-1.448315	-2.014317	-1.390773e-06	6.579647e-03	-3.653644e-04

The results agree with the ones in [1], the difference in the numbers coming purely from the fact that due to time constraints we did not perform the calculation with as high an accuracy as done in the reference. This is clear when one looks at the values for $\Lambda_2 - 1$ which should be strictly zero. The fact that it is not shows that the calculation done is not highly accurate. Nevertheless it is obvious that we find the same orbits as found in [1]. Provided more time would be available, we could simply increase the accuracy of our calculations and exactly reproduce the values found in table (1) of [1].

This concludes the first part of our investigation. We were able to reproduce some of the crucial results from [1]. With more time at hand it should be rather straightforward to extend the calculations to get longer periodic orbits using the symbolic dynamics described in [1] for initial guesses. This would allow us to completely reproduce [1] (minus the errors found ☺).

B. Results for Real space simulation

Instead of spending more time reproducing the rest of the results from [1], we turn now to the - numerically even more time intensive - part of simulating the system in real space.

The authors of [1] showed that while changing the number of Fourier modes in the simulation does not change the structure of the Feigenbaum diagram, it changes the values ν for which one finds chaos, period three windows, etc. Since our real space simulation is fundamentally different than the Fourier space representation, we first need to scan the ν-space in order to determine which value for ν will approximately correspond to the value of ν for the Fourier space simluations. Therefore we start out by plotting Feigenbaum diagrams. We do that for u_i at different values of the space variable i.

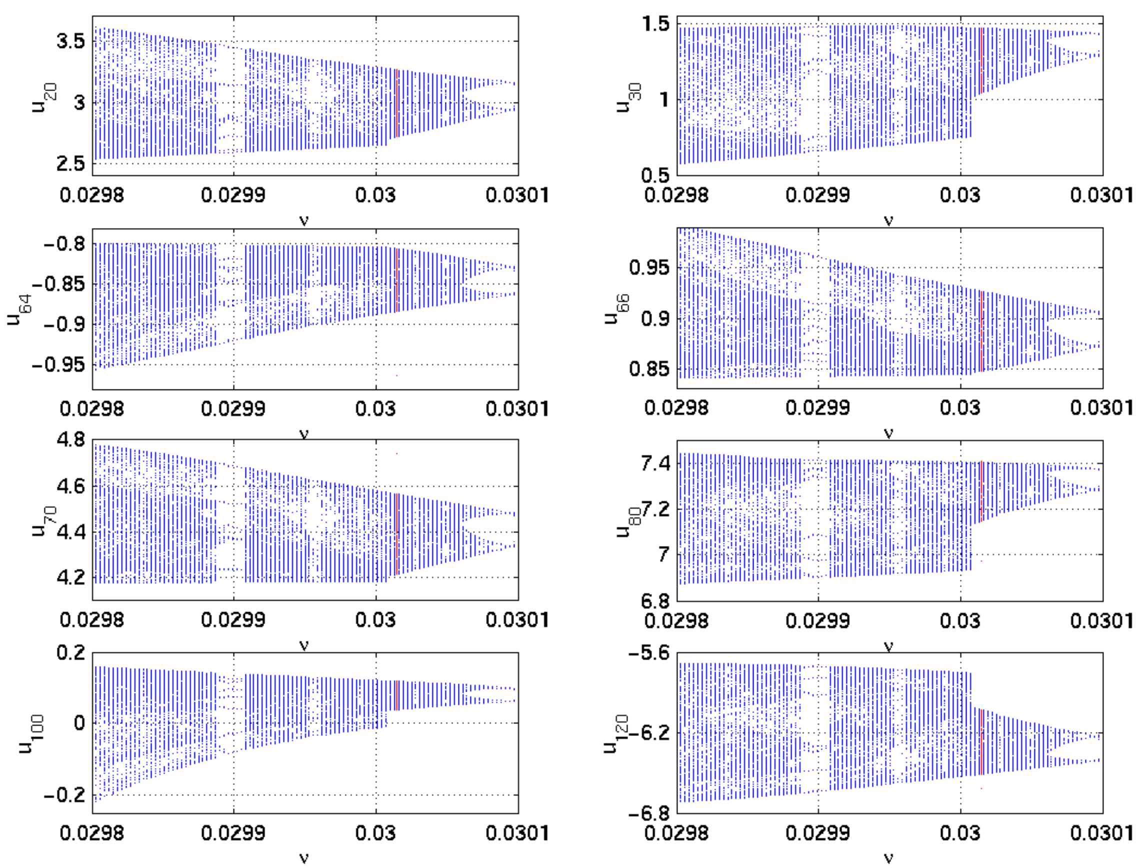

FIG. 4: Feigenbaum diagrams for $u_{i=65} = 0$ in real space

The first thing one notices is that before the two attractors merge, the system is either on the upper or lower attractor but does not jump. This is simply due to the fact that for this simulation, to ensure that we are always in an antisymmetric state, the initial condition is always the same (a sin function) for each value of ν. This means a given variable u_i either starts on the upper or lower attractor for every ν and stays there. At $\nu \approx 0.030005$ the two attractors merge and the variables now live on the combined attractor.

One also notices that the structure of the diagrams corresponds to figure (3), but the ν axis is shifted. The overlap of the two attractors happens for the real space simulation at $\nu \approx 0.030005$ compared to $\nu \approx 0.0299$ for the Fourier space simulation. To achieve consistency with the results gained via the Fourier space investigation at $\nu = 0.02991$, we therefore need to chose a diffeent ν in real space. Choosing $\nu 0.030015$ ensures that the distance from the attractor merging point is about the same as it is for the Fourier space scenario. For that choice of ν we thus hope to find almost the same results for the periodic orbits using the real space simulation. The next figure shows the evolution of the system in real space. Since we pick the subspace $u(x) = -u(-x)$ as described in section III A, we only plot here the system in the interval $i = 1..128$ which corresponds to $x = [-\pi, 0]$. The other half $x = [0, \pi]$ is simply the negative of the shown plot.

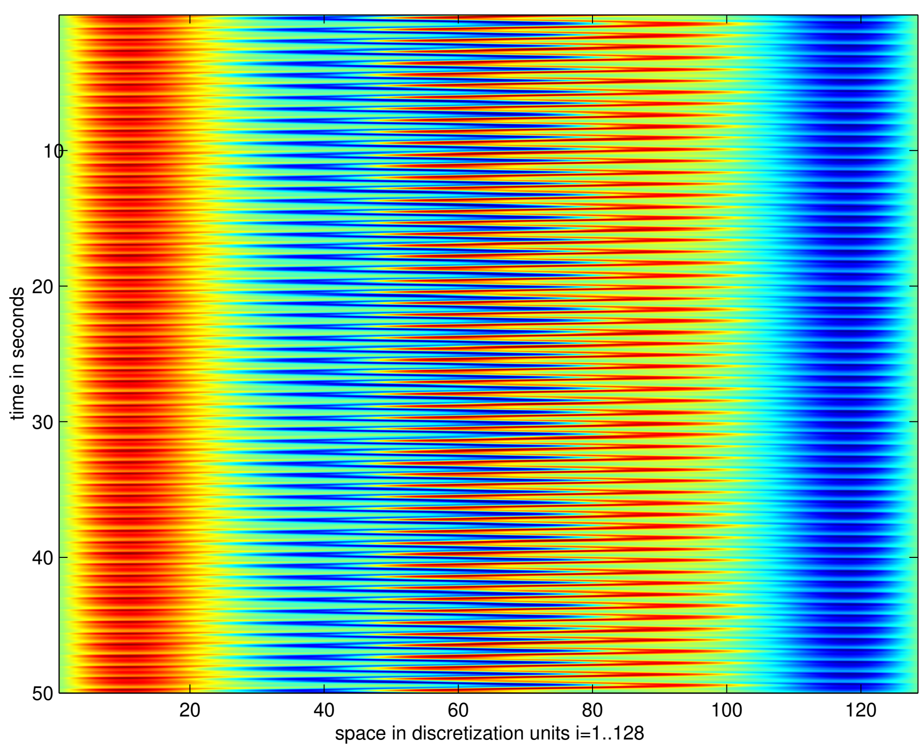

FIG. 5: Evolution of equation for $\nu = 0.030015$

Next we would like to find a sensible Poincare section in real space, similar to the one obtained in Fourier space. As condition for our Poincare section we arbitrarily choose $u_{i=65} = 0$. We then plot the Poincare sections u_i^{t+1}, u_i^t for different values of i.

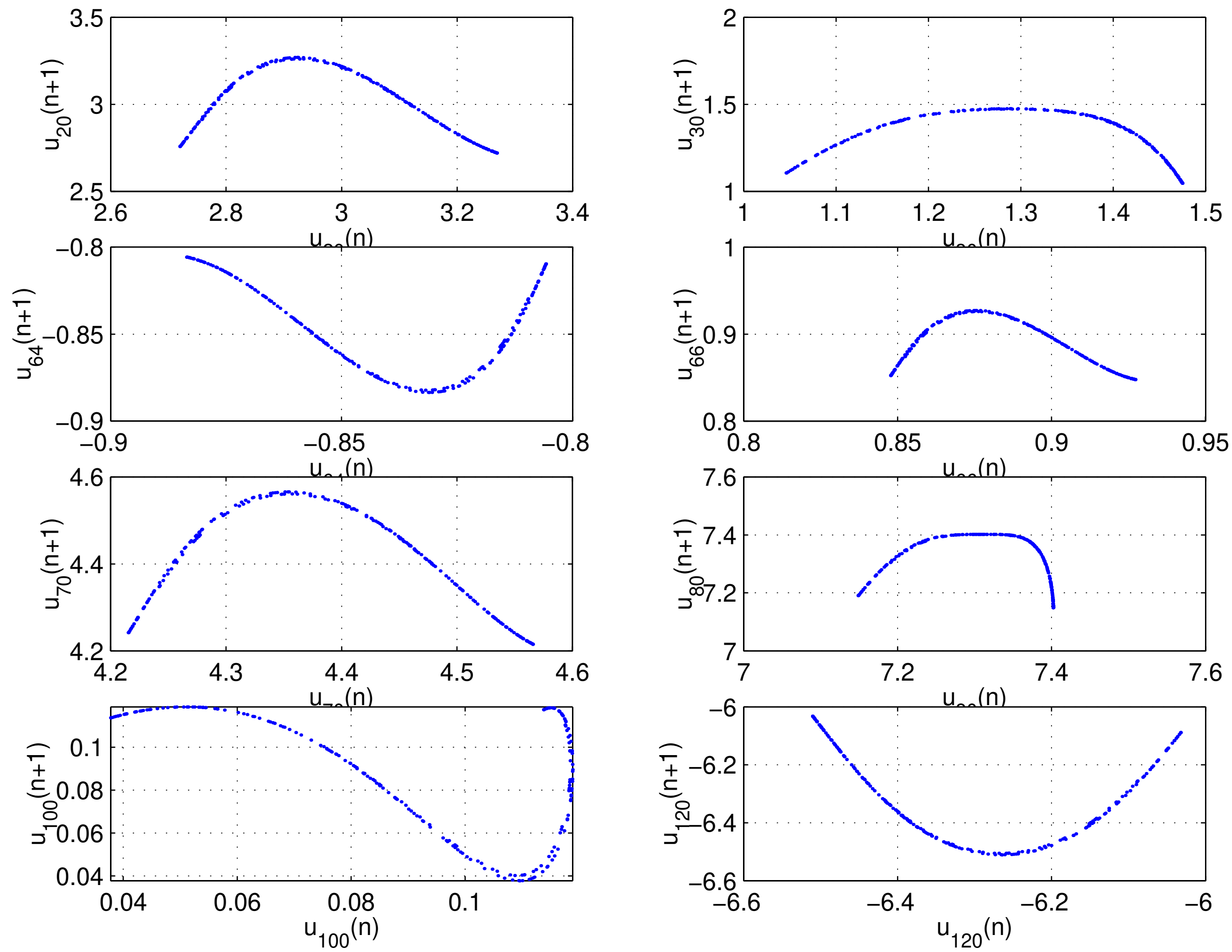

FIG. 6: Poincare sections for $i = 20, 30, 64, 66, 70, 80, 100, 120$, $\nu = 0.030015$

Although in detail different, the overall structure of the Poincare sections can be compared to the one found in Fourier space.

As a last step, we try to calculate the fixed points for this Poincare map, corresponding to the lowest periodic orbits. One expects the values for the Jacobian to be almost the same as the one found in Fourier space, since periodic orbits are topologically invariant and should give the same values in either representation.

Unfortunately, numerics struck once more. Since evolving the Jacobian in real space involves constantly multiplying matrices of the size 256×256, the code is very slow. We were therefore not able to perform accurate integrations to allow a good convergence to even the lowest periodic orbits corresponding to fixed points of the Poincare map. While we noticed that we start to converge to a periodic point, we were not able to achieve a satisfactory accuracy. This comes from the fact that the integration steps have to be taken rather large and therefore even though the Newton iteration brings us close to the fixed point, the following integration drives us again away. Depending on the integration accuracy one will find some kind of 'equilibrium' for the error. In our case time constraints (only several hours before project deadline) did only allow for a rather crude integration scheme. Therefore we never really converged close enough to the fixed point. While we were able to get the norm of the difference between old and new state-space point of the Newton-Raphson iteration down to around 0.001, this value turned out to be still too inaccurate to get any good results for the eigenvalues of the Jacobian. Since the Jacobian is of dimension 256×256, relatively small inaccuracies in its values can result in big differences for the numerically found eigenvalues.

The following result should therefore not be regarded as a useful quantitative result but rather be taken as a general 'proof of concept' that one is indeed able to reconstruct the same periodic orbits in real space that one found in Fourier space. More time/higher accuracy should then give results that agree much better with the one found in Fourier space. While I believe that it is purely a matter of numerical accuracy/available time to get exactly the same results in Fourier space as in real space, I admit that the result shown below leaves enough room for doubt. In the time at hand I was unfortunately not able to give more convincing numerical evidence.

TABLE II:

Orbit Symbol	Return Time	Λ_1	$\Lambda_2 - 1$	Λ_3	Λ_4
1	0.84	-2.07	0.1	N/A	N/A

V. CONCLUSIONS AND OUTLOOK

I have investigated the Kuramoto-Sivashinsky equation in Fourier space and real space. In the first part of the project, I was able to reproduce several of the results and correct an inconsistency found in [1]. Lack of time did not allow for a full reproduction of [1] but since I was able to reproduce the crucial parts, the other ones would be rather straightforward, provided enough time would be available.

In the second part, I investigated the real space representation of the KS equation. I was able to at least provide some evidence that one can extract information about topological invariants, specifically periodic orbits, from this representation as well. This result is expected and shows that either representation is valid and that for a given promblem one is allowed to chose the representation that is most convenient. I admit that further and more accurate numerical investigations need to be done to put this claim on a firmer "numerical/experimental" ground. (Apart from the in my view much firmer basis that theory tells us topological invariants don't change under linear transformations). While I did not learn much physics by doing this project, I think I can by now claim to be almost an expert with MATLAB. I also gained a lot of experience in coding all kinds of integrators for ODEs, PDEs, etc. (Most of them are not listed in this project due to the fact that I only listed the one that worked best. Many other ones that failed to do the calculations in a reasonable time are not mentioned...) It seems to me that the project would have been more appropriate as my project for the course 'Numerical Methods for PDEs' which I happened to take parallel to this course. (Ironically the project I did for the PDE course was much less computational, did not involve any PDE and had more (nonlinear) physics to it).

I have learned to never try and work on a nonlinear PDE when one has only a few weeks at hand. The numerics is simply too time consuming. To give a flavor for the numerical times involved, I just mention that I usually used five computers *at the same time* (a relatively fast Alpha workstation and four computers of the type Pentium III, 800Mhz, 512MB RAM) to perform various parts of the calculations/simulations. Producing the Feigenbaum diagrams for example took about two days each on the Alpha workstation.

I freely admit that I'm not very excited about the project. It was nice to learn all kind of nifty numerical stuff and on the way correct a mistake found in [1]. But the fact that the

the project consisted almost entirely of coding and did hardly involve any physics reduced my excitement factor by a lot.

My original plan was to decrease ν, making the system more turbulent/interesting and study its behaviour in that regime - something I think would be more interesting from a physical point of view. Further possible and interesting investigations would be to find a good real space symbolic dynamics for the ν value studied above and a symbolic dynamics for both or one of the spaces if ν is further decreased.

Unfortunately due to the lack of time and my own stupidity in choosing a PDE for the project, the computational/coding part and numerically setting up the system required so much time that there was not enough to do any further investigations than the ones done.

In any case, I think I now deserve to tackle part 11 of the list of tasks mentioned in [3]!

---

[1] *Spatiotemporal chaos in terms of unstable recurrent patterns*
F. Christiansen, P. Cvitanovic', V. Putkaradze, Nonlinearity **10**, 55 (1997)

[2] *Classical and Quantum Chaos*
P. Cvitanović, R. Artuso, R. Mainieri, G. Tanner and G. Vattay
`www.nbi.dk/ChaosBook/`, Niels Bohr Institute (Copenhagen 2001)

[3] *Spatiotemporally periodic solutions by variational methods - a term project*
P. Cvitanovic', R. Mainieri
http://www.nbi.dk/ChaosBook/projects/variational.ps.gz

[4] *PETSc home page*
http://www.mcs.anl.gov/petsc