Technical University of Munich
Prof. Dr. Nils Thürey School of Computation, Information and Technology
Dr. Mario Lino, Benjamin Holzschuh, Patrick Schnell, Qiang Liu and Chengyun Wang Summer Semester 2025

Advanced Deep Learning for Physics (IN2298)
Exercise 10
Kuramoto-Sivashinsky Simulator

This exercise comprises the first of two parts of a comparative study between supervised and differentiable learning approaches based on the Kuramoto-Sivashinksy equation. The goal across both exercises is to demonstrate the challenges and benefits of these learning approaches in an application-oriented context. Refer to Lecture 3 for a thorough discussion.

This first part of the project is concerned with the implementation of a differentiable solver for the Kuramoto-Sivashinsky equation and with generating a dataset for training and testing prediction networks. As introduced in earlier exercises, differentiable physics solvers are not part of the common machine learning libraries, but are available through research-focused extensions such as Φ_{Flow}.

Kuramoto-Sivashinsky Equation

The Kuramoto-Sivashinsky (KS) equation is a 1-dimensional, nonlinear PDE known for its chaotic behaviour. The state variable u is defined on a periodic domain $x \in [0, X]$ and satisfies

$$\frac{\partial u}{\partial t} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^4 u}{\partial x^4} + \frac{1}{2}\frac{\partial (u^2)}{\partial x} = 0. \tag{1}$$

The KS equation is often used as a low-dimensional and computationally cheap, yet highly chaotic testbed for numerical solvers. The periodic domain allows for an efficient calculation in Fourier space. Denoting the Fourier transform as $\mathcal{F}$, the following relation of partial derivative hold in Fourier space

$$\hat{u}(k) = \mathcal{F}(u(x)), \qquad ik\hat{u}(k) = \mathcal{F}(\frac{\partial u(x)}{\partial x}), \tag{2}$$

where k denotes the wavenumbers. You can use this relation for efficient and exact derivative calculation. Since the KS equation is a stiff system, we will use an exponential time-stepping method for temporal integration. A thorough discussion of exponential time-stepping methods for stiff problems can be found in [1]. Following their notation, the equation is split into linear and non-linear parts, such that

$$\frac{\partial \hat{u}}{\partial t} = L\hat{u} + N(\hat{u}), \tag{3}$$

where L represents the treatment of linear parts (i.e. second- and fourth-order derivatives) and $N(\hat{u})$ the non-linear ones. In Fourier space, the linear parts are efficiently computed as element-wise multiplication with a constant vector dependent on the wavenumbers k, as indicated by Eq. (2). The non-linear term is treated by transforming into real space, where the non-linear function is computed, and back into Fourier space at every step. The exponential time-stepping Runge-Kutta of second order (`etrk2`) takes an input state u_n and computes u_{n+1} as follows.

$$a_n = \hat{u}_n e^{L\Delta t} + N(\hat{u}_n) * (e^{L\Delta t} - 1)/L \tag{4}$$

$$\hat{u}_{n+1} = a_n + (N(a_n) - N(\hat{u}_n)) * (e^{L\Delta t} - 1 - L\Delta t)/(L^2\Delta t) \tag{5}$$

Note that the factors $(e^{L\Delta t} - 1)/L$ and $(e^{L\Delta t} - 1 - L\Delta t)/(L^2\Delta t)$ are not defined for $k = 0$, as $L(0) = 0$. The same is true for $k = \pm 1$, where $k^2 = k^4$ holds and thus $L(\pm 1) = 0$. To obtain the factors for $k = 0$, calculate the limiting form of these factors of $\lim_{L\to 0}$, i.e. $\lim_{L\to 0} = \frac{e^{L\Delta t}-1}{L}$ while using Taylor expansion for exponential terms.
A solver step will be referred to as $u_{n+1} = \mathcal{P}(u_n)$.

(1) Solver implementation

Implement a differentiable solver for the KS equation in Φ_{Flow} and generate a dataset of sample solutions. Note again that the temporal integration is computed in Fourier space where the advancement of the linear terms is done by multiplication with a *constant* term $e^{L\Delta t}$. Along with similar terms, this can be pre-computed to give a faster solver. Your solver should be capable of batch-processing, as this severely accelerates dataset calculation and training of the differentiable models in the next exercise.

(2) Dataset generation

After implementing the above procedure, we can create a dataset based on variations in the initial conditions. In the coming learning tasks, we will set the physical length scale $X = 10$, which is discretised by 50 grid-cells. For the stability of `etrk2`, we set $\Delta t = 0.5$. Use Φ_{Flow}'s double-precision capabilities by creating and executing your solver instance and in a `with math.precision(64):` block. With these parameters, generate a training dataset based on a set of initial conditions of the form $u_0 = \cos(3\pi x/X) \pm 0.1 \cos(2\pi x/X)\big[1 - \alpha \sin(2\pi x/X)\big]$. Choose some variations of the perturbation (i.e. $\pm$) and the parameters α from $[-8, 8]$. Your simulations should look similar to Figure 1. Generate 6 simulations with 8000 samples each (ca. 10MB of data in total).

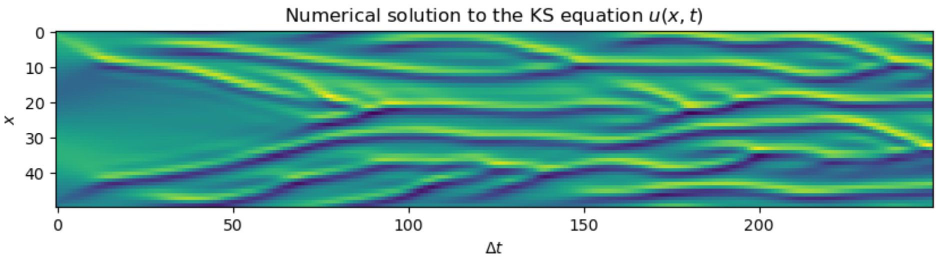

Figure 1: Dataset sample

Hint: The `phi.math` library contains all necessary functions:

- `math.fft()` and `math.ifft()` transform tensors to and from Fourier space. It only affects spatial dimensions.
- `math.fftfreq()` creates a tensor that lists the Fourier frequencies matching `fft()`.
- `math.divide_no_nan()` divides two tensors, returning 0 where the divisor is 0.
- Real/imaginary parts of complex numbers can be accessed via `tensor.real` and `tensor.imag`.

References

[1] S. Cox and P. Matthews, "Exponential time differencing for stiff systems," *Journal of Computational Physics*, vol. 176, no. 2, pp. 430–455, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999102969950

Submission instruction

Please upload a single PDF file containing your results along with your code for implementation tasks or your derivation for non-implementation tasks (LaTex typesetting). The uploaded PDF should only include the final code, so please trim empty spaces and your intermediate work before submitting.

The easiest way to generate such a PDF is by using Jupyter notebooks and LaTex (we recommend MiKTeX for Windows users). With Jupyter and LaTeX installed, you can create a PDF from your notebook by running *jupyter nbconvert –to pdf your-notebook.ipynb*

Additional information

This is an individual assignment. Plagiarism will result in the loss of eligibility for the bonus this semester.

If you have any questions about the exercises, please contact us via the forum on Moodle. If you need further face-to-face discussion, please join our weekly online Q&A session (every Monday at 15:00 and 16:00 via BBB).