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

Advanced Deep Learning for Physics (IN2298)
Exercise 1
Introduction to ΦFlow

This worksheet introduces you to Φ_{Flow}, the differentiable simulation framework used in the lecture and homework exercises. All important links can be found on the GitHub Homepage. From there, you can also find the documentation overview page.

Installation and usage

For the homework exercises, we recommend using Φ_{Flow} inside a Jupyter notebook. You can run these on your local machine or in the cloud using Google Colab. If you run it locally, you need to install Python 3.6 or newer as well as Jupyter.

Once you have a Python notebook running, you can install Φ_{Flow} by executing the following statement:

```
!pip install phiflow
```

The convenience import statement gives you access to the main classes and function:

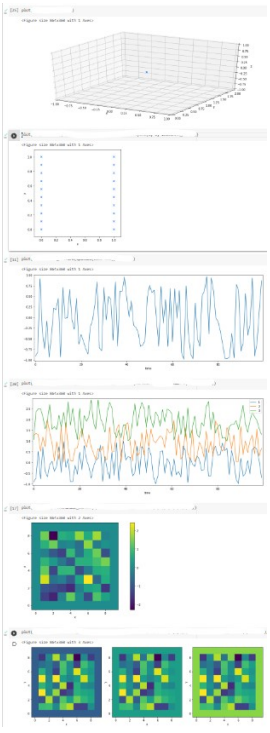
```
from phi.flow import *
```

(1) Tensors representing physical data

Tensors are the most fundamental way of dealing with data and there are numerous ways of creating tensors in Φ_{Flow} depending on the situation. Each tensor dimension in Φ_{Flow} is assigned one of four types: *channel*, *spatial*, *instance* or *batch*. Unlike with other frameworks, which determine these types from the dimension order, the types are explicitly specified by the user in Φ_{Flow}, e.g. `spatial(x=64, y=48)` creates two spatial dimensions named x and y. Φ_{Flow} also provides a simple way to visualize data via the `plot()` function. See the documentation for an explanation and examples.

Create and plot the following tensors:

1. A single vector representing the point $x = 0, y = 1, z = 0$.
2. Two columns of points, at $x = 0$ and $x = 1$, 10 points per column linearly spaced between 0 and 1.
3. A time-dependent waveform or signal consisting of 100 samples. Each sample is uniformly distributed between -1 and 1.
4. Three such curves in one plot, centered around 0, 1 and 2, respectively.
5. A two-dimensional scalar 10x10 grid. Each value is sampled from a normal distribution but all edge values are zero.
6. Three such plots next to each other, with edge values -1, 0, and 1, respectively.



Hint: Each task can be solved in one line of code. Functions you may used from `phi.math`: `vec()`, `tensor()`, `concat()`, `stack()`, `pad()`, `linspace()`, `random_uniform()`, `random_norm()`.

(2) Bouncing balls simulation

We want to simulate perfectly round balls with radius 0.1 bouncing up and down in two dimensions due to gravity and an elastic interaction with the floor. Initially, there are 10 balls, all located at $x = 0, y = 1$ with initial speed 3. Their angles of attack should be linearly spaced between horizontal (positive x) and vertical (positive y).

Write a simulation function that takes in the current state and a time increment Δt. It should compute the linear movement, air friction, gravitational force and elastic collision

with the floor at $y = 0$ and return the next state. For the air friction, you can simply multiply the speed by $0.7^{\Delta t}$ each step. Run your simulation for about 10 seconds with $\Delta t \approx 0.1$ and plot the final state. Also create a video of the simulation.

Hint: You may store the full state using a `PointCloud`, i.e. `balls = PointCloud(Sphere(x0, radius=.1), v0)` or you may store position and velocity separately.

Hint: You may use `phi.physics.advect.points()` to update the position, or you can implant simple Euler method by your self.

Hint: You can pass a `Sphere` instance to `plot` to draw circles. You may use the `animate` argument of `plot` to create an animated plot. Check out the animation gallery for examples.

Hint: You may use `iterate()` to run your simulation and get the full trajectory of states.

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).