Prof. Dr. Nils Thuerey
Dr. Liwei Chen, Dr. Shuvayan Brahmachary
Bjoern List, Philipp Holl, Patrick Schnell

Technical University of Munich
Faculty of Informatics
Summer Term 2022

Advanced Deep Learning for Physics (IN2298)
Exercise 1
Sphere Packing

The arrangement of atoms and molecules is widely studied in condensed matter physics and chemistry. As a simplified model, each atom can be represented by a solid sphere. When densely packing equally-sized spheres, they form a hexagonal pattern but when their sizes vary the resulting positions can become chaotic.

In this exercise we want to measure how densely spheres of different sizes can be packed. We consider a simplified 2D model where the layout can be visualized more easily. We will start by distributing spheres (or rather circles) in a periodic 2D domain of fixed size. Next, we optimize the sphere locations until we get a valid physical state where no spheres overlap. By repeating these steps for smaller and smaller domains, we can find the smallest area that can fit all spheres.

(a) Randomly distributed

(b) Non-overlapping

Figure 1: 1000 spherical particles in a square domain, in a non-physical and physical state.

(a) Setup

Create a sufficiently large square periodic 2D domain and randomly distribute 1000 spheres within it. Let half of the spheres have radius $R = 1$ and the other half radius $R = 0.7$. Visualize the resulting configuration for multiple seeds. You can use Φ_{Flow}'s built-in visualization.

```
vis.plot(Sphere(locations, radius=radii), lib='matplotlib')
```

(b) Energy function

Before optimizing the sphere locations, we need to define a loss or energy function that penalizes overlapping spheres. The value of the energy function should be 0 for spheres that do not overlap and should increase quadratically with overlapping distance.

Hint: To evaluate pairwise distances between the locations listed along the dimension `'spheres'` in `x`, use

```
dx = x - math.rename_dims(x, 'spheres', 'others')
```

Hint: You can apply a transformation to `dx` in order to respect the periodic boundaries.

(c) Gradient Descent Optimization

Write a gradient descent optimizer to minimize the energy function defined above. The optimizer should adjust the sphere locations until none of them overlap (energy = 0). You can use `math.functional_gradient()` to compute the gradients w.r.t. the locations. Plot the optimized configuration, ensuring all spheres lie within the domain!

Hint: The derivative of the `sqrt` function (required to compute the length of a vector) is infinite at 0. To avoid `NaN` values in the optimization, this has to be avoided, e.g. with a lower cutoff or custom gradient or by avoiding the `sqrt` completely.

(d) Higher-order Optimization

The SciPy library provides a number of higher-order optimizers, such as `'L-BFGS-B'`. These are available in Φ_{Flow} via the `minimize()` function, e.g.

```
locations = math.minimize(energy_function, Solve('L-BFGS-B', 0, 1e-5,
max_iterations=2000, x0=initial_locations))
```

Minimize the energy using a higher-order optimizer. How many iterations are needed compared to your gradient descent optimizer?

(e) Smallest domain

Incrementally decrease the domain size until the loss cannot be reduced to below 0.01 anymore. How large is the smallest domain that can contain all spheres without any overlaps? Repeat the experiment for multiple values of $R \in (0, 1]$ and multiple initial configurations each. Plot the domain size against R, showing the standard deviation across random configurations as error bars.