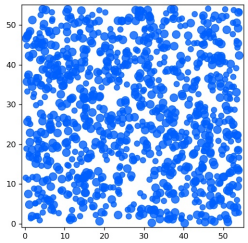
Prof. Dr. Nils Thürey
Dr. Mario Lino, Benjamin Holzschuh, Patrick Schnell, Qiang Liu and Chengyun Wang
Technical University of Munich
School of Computation, Information and Technology
Summer Semester 2025

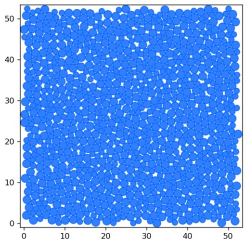
Advanced Deep Learning for Physics (IN2298)
Exercise 3
Sphere Packing

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.

(1) Setup

Create a **sufficiently large** square 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.

(2) 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. Note that for the energy function, the square domain should have periodic boundaries.

(3) 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). 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.

(4) Higher-order Optimization

The SciPy library provides a number of higher-order optimizers, such as `'L-BFGS-B'`. Minimize the energy using a higher-order optimizer. How many iterations are needed compared to your gradient descent optimizer?

(5) Smallest domain

Incrementally decrease the domain size until the loss (energy function) cannot be reduced below 0.01 anymore. How large is the smallest domain that can contain all spheres without any overlaps?

(6) Extensive Sphere Packing (optional)

Repeat the experiment for multiple values of $R \in (0, 1]$ and multiple initial configurations each. Half of the spheres should always have radius $R = 1$ and the other half radius $R \in \{0.1, 0.2, \ldots, 1.0\}$. Plot the domain size against R, showing the standard deviation across random configurations as error bars.

Hint if you are using PhiFlow

- You can use Φ_{Flow}'s built-in visualization:

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

- You can use `phi.geom.Sphere()` to represent spheres.
- To evaluate pairwise distances between the locations listed along the dimension `'spheres'` in `x`, use
  `dx = x - math.rename_dims(x, 'spheres', 'others')`
- You can apply a transformation to `dx` in order to respect the periodic boundaries:
  See `math.extrapolation.PERIODIC` .
- You can use `math.functional_gradient()` to compute the gradients w.r.t. the locations.
- High order optimizers 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))`

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).