Upload Date:	`[Moodle Assignments]`	March 28th, 2023
Hand-In Deadline:	`[Moodle Assignments]`	April 18th, 2023, 08:45
Correction Session:	`[VU class]`	April 18th, 2023

7 **Force computation with Lennard-Jones. (5P)** The Lennard-Jones potential $U_{LJ}(\vec{r}_{ab})$ can be used to compute pairwise interactions between molecular beads a and b in a molecular-dynamics simulation.

$$U_{LJ}(\vec{r}_{ab}) = 4\epsilon \left(\left[\frac{\sigma}{r_{ab}}\right]^{12} - \left[\frac{\sigma}{r_{ab}}\right]^{6} \right) \tag{1}$$

Where $\vec{r}_{ab} = \vec{r}_b - \vec{r}_a$. For this exercise, assume $\epsilon = \sigma = 1.0$. Remember that $\vec{F} = -\nabla U$. The calculated forces should be repulsive for distances smaller than $\sqrt[6]{2}\sigma$, attractive for larger distances. **Note:** Molecular dynamics typically are too computation-heavy to be performed in Python, but plotting potentials and resulting forces is a common enough task.

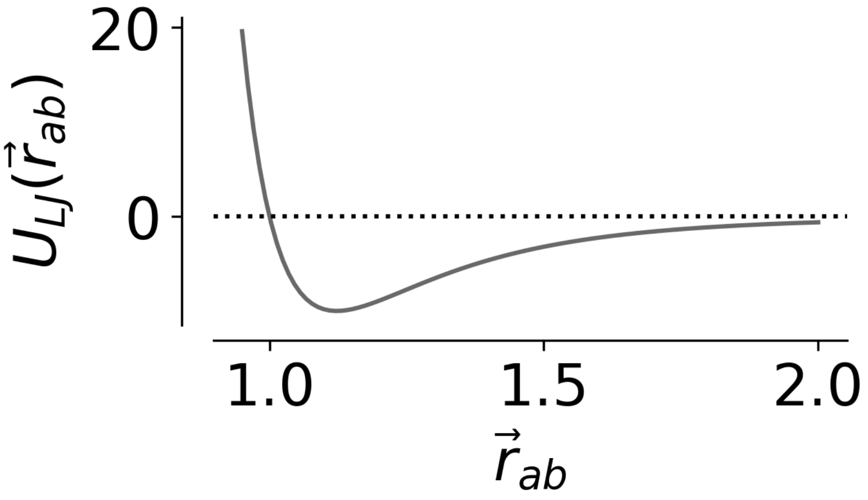

Figure 1: Exemplary Lennard-Jones potential Eq.1 for $\epsilon = \sigma = 1.0$.

Perform the following tasks:

1. Write a function that takes the distance of two particles (a and b) in $\mathbb{R}^3$ as input and returns U_{LJ}. The particles should be no closer than 0.01σ. Check that this is indeed the case (use an `assert` statement).
2. Write in a Markdown cell an explanation for why a check like this might be important. The value of 0.01σ is arbitrary, but can you think about a case that might cause problems?
3. Write a function that computes the force magnitude acting on particles at distance r, from a **numerical derivative** of $U(r)$. Do so by using `scipy.misc.derivative` to compute the derivative of the potential. **Hint:** Guidelines to import scipy modules can be found at: "https://docs.scipy.org/doc/scipy/reference/"
4. `scipy.misc.derivative` takes an argument `dx`. Write in a Markdown cell an explanation for why you should pay attention to this parameter.
5. Plot $10 * U_{LJ}(r)$ and $F_{LJ}(r)$ in a proper range ([1,2] should be fine) , draw an horizontal line at 0, and a vertical line at the potential energy minimum **Hint:** Use `scipy.optimize.minimize`

8 **Mass-Spring system with dampening.(6P)** Assume you are given a point-mass m hanging off the ceiling from a spring with spring constant k. Gravity will act on it, and, depending on the initial position x, it will bounce up and down. Assume here that x is the perturbation from its resting length, with positive x meaning that the spring is extended, and negative x that the spring is shortened. The corresponding differential equation looks as follows:

$$\ddot{x} = -\left(\frac{b}{m}\right)\dot{x} - \left(\frac{k}{m}\right)x + g \tag{2}$$

Take care of the following tasks:

1. Set all parameters of the system $(x_0, \dot{x}_0, m, k, b)$ according to:
   $x_0 = 0.0, \dot{x}_0 = 0.0, m = 1.0, k = 10.0$ and $b = [-0.2, 0.0, 0.2]$.
   $g \cong 9.81$ is the gravitational pull.
2. Using `scipy.integrate.odeint`, solve the differential equation for $x(t)$ and $\dot{x}(t)$ for $t \in [0; 15.0]$ for all three different values of b. **Hint:** You can pass a list $[x_0, \dot{x}_0]$ into `scipy.integrate.odeint` instead of a single starting value x_0. If you do so, the function you plug in must take in a list $[x, \dot{x}]$ and return a list $[\dot{x}, \ddot{x}]$ as well. In fact, you can return $[\dot{x}_{new}, \ddot{x}_{new}]$ in this form for the updated values. `odeint` will then return a list state such that $x = \text{state}[:, 0]$ and $\dot{x} = \text{state}[:, 1]$. For example:

```
def template(state, t, ...):
     x = state[0]
     dx = state[1]
     ...  # compute dx, ddx
     return [dx, ddx]
```

3. Identify the influence of b upon the system by plotting $x(t)$ and $\dot{x}(t)$ for $t \in [0; 15.0]$ into their own separate subplots. Give your recount of what b does.