Phys 3510 Final Exam
Dec 13, 8am – 10am
100 pts total
If you arrived *x* minutes late, you will get to stay for an extra *x* minutes to finish the exam.

In this project, we will perform a molecular dynamics simulation that replaces an existing Lennard-Jones (LJ) potential with a similar potential: the Morse potential. We will do this in two steps. **In 1-morse.py**, we will review the LJ potential and compare it with the Morse potential. In **2-md.py**, we will perform the actual molecular dynamics simulation, where the exact same code we finished in class is provided as a starting point.

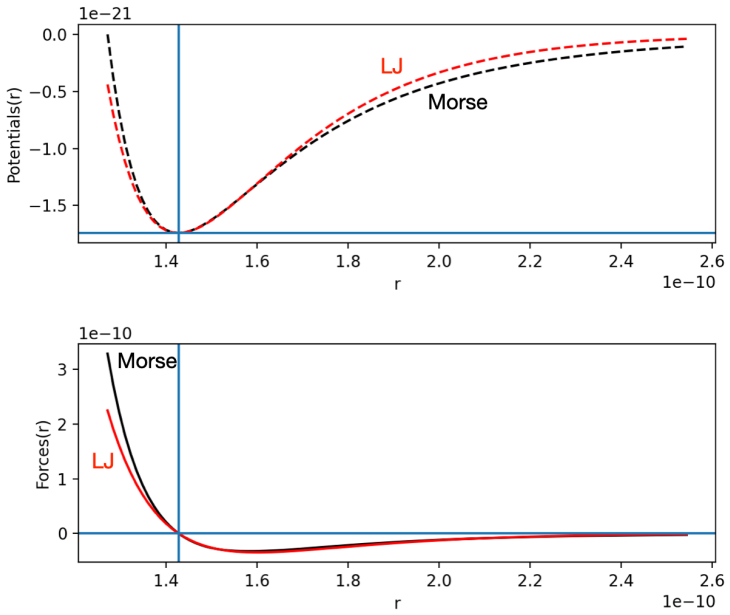
Just as a refresher, the 2-md.py program describes the dynamics of particles moving according to classical Newtonian mechanics, where we evolve the system in time using small timesteps dt, using the 2nd order Runge-Kutta ODE solver. The module needed to run it, myode.py, is also provided.

1. **(15 pts) Comment in morse.py.** As we have did many times in class, before any numerical calculation, we first establish some analytical groundwork to set up expectations. The existing code uses the Lennard-Jones potential given by $U_{LJ}(r) = 4\varepsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right)$, where ε and σ were our parameters.
   (a) What is the minimum value of $U_{LJ}(r)$? Expressed using ε, not numbers.
   (b) This minimum value is achieved at $r=r_0$. Find r_0. Expressed using σ, not numbers.
   (c) Particles with small energies compared with ε will mostly undergo simple harmonic oscillations near this energy minimum. What is the effective spring constant $k_{\text{eff,LJ}}$ of such oscillations? Expressed using ε and σ, not numbers.

2. **(15 pts) Comment in morse.py.** Following Task 1, consider a potential very similar to the LJ potential: the Morse potential, $U_{\text{Morse}}(r) = D_e(e^{-2a(r-r_e)} - 2\,e^{-a(r-r_e)})$, where D_e, a, and r_e are parameters. Same notations can be found on Wikipedia.
   (a) What is the minimum value of $U_{Morse}(r)$?
   (b) This minimum value is achieved at $r=r_0$. Find r_0.
   (c) For effective spring constant $k_{\text{eff,Morse}}$ of small oscillations in the Morse potential, show that $k_{\text{eff,Morse}} = 2\varepsilon a^2$. **Hint**: Taylor expansion of the exponential function to second order is $e^x \approx 1 + x + \frac{1}{2}x^2$ for small x.

3. **(15 pts) Still working in morse.py.**
   (a) In morse.py, construct two functions that calculates the Morse potential and the

forces derived from it. You can follow the given examples of the LJ potential, where LJ potential parameters are provided.
(b) Use what you've found in Task 1 and 2 to define the three parameters of the Morse potential, so that its minimum, its r_0, and its effective spring constant matches with the LJ potential.
(c) Final check in morse.py! Plot the Morse potential and the force derived from it in the two subplots. Comparing with the LJ potential should look like the following.



4. **(15 pts) Working in md.py.** First make sure that md.py works and generates the correct animation. You're now ready to replace forces derived from the LJ potential with forces derived from the Morse potential in the MD simulation. Copy what you need from morse.py to md.py, and modify md.py to see if everything works. The simulation animation should be very similar.

5. **(10 pts)** The existing code generates the same initial random velocities for each particle every time you run it. Modify the code to generate *different* random velocities every time. Very simple – don't overthink it.

6. **(15 pts)** Change the reflective-wall boundary conditions to **periodic boundary conditions**: a particle that pops out of the left boundary will pop in on the right side with its velocity unaffected and vice versa; same with upper and lower boundaries.

7. **(15 pts)** Evolve the system for 1000 timesteps. At the last timestep, using numpy functions, calculate the average value $<K>$ and the standard deviation σ_K of the kinetic energies of all the particles, both in eV.