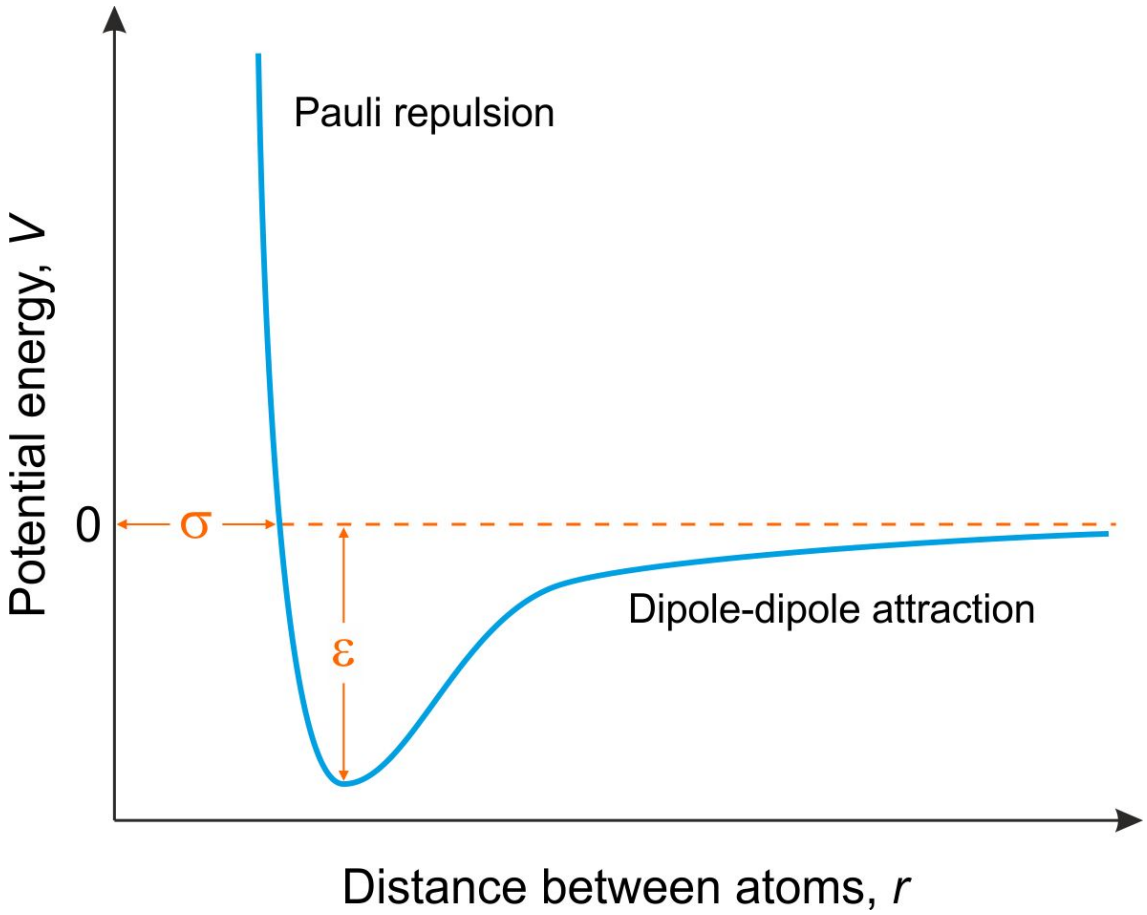
BMI 2005 Homework #1
Assigned January 16, 2019
Due January 30, 2019

Homework must be submitted to Blackboard using a Jupyter notebook!

1) Molecular dynamics simulations integrate Newton's equation of motion in order to study the time evolution of atomistic or molecular systems. This is a classical (rather than quantum) technique where the equations of motion are fully deterministic and time-reversible (more on this later!). Forces on the atoms and molecules are therefore fairly simple to approximate because we make classical (rather than quantum) assumptions about the form of the potentials which define the interactions between atoms and molecules. In this class, and through a few homework assignments, we are going to focus on the simplest possible MD calculation with relevance in biomedical applications, namely the "non-bonded" or Van der Waals molecular dynamics simulations. In these simulations, we consider only atoms that interact with each other over long ranges and do not form new bonds or participate in any chemistry. In many ways, these are similar to the real-world behavior of noble gasses in both liquid and gaseous form.

   We define the way that such atoms interact via a non-bonded potential function. Here, we will use perhaps the most common, the Lennard-Jones (LJ) potential, defined thusly:

$$V_{\mathrm{LJ}} = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]$$



Here, ε is the depth of the potential well, or how strong the non-bonded attraction is between two atoms, and σ defines the distance at which the inter-particle potential is zero. That is, σ defines where the interaction becomes repulsive at two particles approach one another.

For a given pair of particles, the distance r, and parameters ε and σ define the potential energy between two atoms. The potential energy of an entire atomistic system is therefore determined by the sum of all pairwise interactions of atoms in the system. This should immediately strike you as a very poor algorithm, and as we have learned this should scale as N^2. This is the naive, pairwise non-bonded molecular dynamics calculation.

a) Write a simple, naive function that computes the potential energy $V_{i,j}$ between two atoms i and j with distance r where ε and σ are set to 1. The only parameter to the function should be the distance r.

b) The following pseudocode implements an efficient function for calculating the potential $V_{i,j}$ with ε and σ supplied as parameters to the function. Prove that this algorithm is correct by recovering the original LJ function definition.

```
LJPOTENTIAL(r, σ, ε)
1 r² ← r * r
2 σ² ← σ * σ
3 z ← σ² / r²
4 u ← z³
5 return( -4εu(1 - u) )
```

c) Implement the above function (b) and report on its performance relative to the naive implementation you wrote in (a).

d) Execute a comparison of the performance of both algorithms you implemented with ε and σ set to 1. Which is more efficient? Argue why based on the number and type of operations each function implements.

2) The initial system configuration (or *phase point*) for a simple molecular dynamics calculation is provided as a set of positions (and, optionally, velocities/momenta) for each atom in the system. For simplicity, we will adopt the convention in this problem that all atoms are the same (that is, they all have the same definition of ε and σ).[1] We will assume ε and σ are always set to 1. Therefore we can basically forget about ε and σ for the purposes of the rest of this assignment.

An XYZ file is a simple file format that defines this initial system configuration. Here is a simple example of such a file:

```
3
Cambridge Cluster Database
LJ          0.5391356726        0.1106588251       -0.4635601962
LJ         -0.5185079933        0.4850176090        0.0537084789
LJ          0.0793723207       -0.4956764341        0.5098517173
```

Here, the first line is the number of atoms in the system, and the second line is a comment. The rest of the file contains the position of each atom in the system line-by-line, prefixed by a string that defines the type of atom (in the case where your system has multiple different atom types). Three other examples, each configured as a box, are available for download alongside this homework assignment.[2]

---

[1] BONUS QUESTION: We typically talk about ε and σ as being properties of the atoms, but what would be the result of two atoms having different ε and σ? How would one calculate the potential energy in this case? The answer is in the “combination” rule. What are the common combination rules? Look up and implement an additional potential energy function that takes as input multiple values of ε and σ and computes a combination rule before returning the potential.

[2] For fun, go ahead and download VMD from the internet (NIH Biophysics/University of Illinois) in order to visualize the box of 200, 500, and 1000 particles provided for this homework.

a) Write a small python program that takes as input an XYZ file and generates a 2D single-precision floating-point numpy array populated with the positions of the atoms from the XYZ file. This program should be capable of reading in an arbitrary positional XYZ file (that is, can read in a file with any number of total atoms in it).[3]

b) Show that it works for the 3-atom case above, and for each of the additional XYZ files provided along with this homework (200, 500, and 1000 atoms).

3) As we discussed above, the total potential energy of a system is determined by the sum of all pairwise interactions. This should immediately strike you as a very poor algorithm, and as we have learned this should scale as N^2.

a) For a given number of particles, what is the total number of pairwise potential energy calculations required to define the total potential energy of the system?

b) Write a program that extends and combines your answers to (1) and (2) above by computing the total potential energy of a system read in from an arbitrary XYZ file. Show your results for each system (3, 200, 500, and 1000 atoms). Note: this requires you to, for the first time, compute the distance between all atom pairs to generate r in order to call the potential energy function. We are interested in the Euclidian distance.

4) At an infinite distance between an atom pair, the LJ potential energy function asymptotically approaches 0, meaning that its contribution to the total potential energy of the system is basically zero. Consider that this observation tells us something about how we might improve the computational complexity of this problem, by simply not considering interactions beyond a certain distance. This is the so-called “cut-off” formulation of non-bonded molecular dynamics, and can substantially reduce the computational complexity of the problem to a problem that scales almost linearly (actually, $N\log N$, which we will prove later).

a) Implement a naive strategy for obviating the need to compute the potential energy for each pair beyond a certain distance. Here, the naive strategy is testing the distance between each pair before calling the potential energy function. In our case, let’s use $r_{cut} = 1.0$. Show the resulting total potential for all systems. Is $r_{cut} = 1.0$ a good value or not?

[3] In good software engineering practice, we would want to carefully consider what other limitations might have an impact on our ability to deal with an arbitrarily large system. There’s one obvious one -- RAM -- but perhaps there are others?

b) Plot the potential energy as a function of the cutoff distance from $0.5 < r_{cut} < 5.0$ in increments of 0.1 for the system of $N = 1000$ atoms. What is a "good" cutoff value? Argue for your choice (there is no definitive correct answer).

c) For your chosen ideal cutoff, show the scaling performance of the method by plotting the time to solution for the naive, and the naive cutoff, potential energy function for all system sizes. Have we solved the N^2 scaling problem? Why or why not?

5) Give the tilde approximations for the following quantities.

a) $N + 1$
b) $1 + 1/N$
c) $(1 + 1/N)(1 + 2/N)$
d) $2N^3 - 15N^2 + N$
e) $\log(2N)/\log(N)$
f) $\log(N^2+1)/\log(N)$
g) $N^{100}/2^N$

6) Give the order of growth (as a function of N) of the running times of each of the following code fragments.

```
a)
def problem_a(n)
    sum = 0
    while k > 0:
        for i in range(k):
            sum += 1
        k = k // 2
    return sum

b)
def problem_b(n):
    sum = 0
    while i < n:
        for j in range(i):
            sum += 1
        i = i * 2
    return sum

c)
def problem_c(n):
    sum = 0
```

```
while i < n:
    for j in range(n):
        sum += 1
    i = i * 2
return sum
```

7) Give a formula to predict the running time of a program for a problem of size N when doubling experiments have shown that the doubling factor is 2^b and the running time for problems of size N_0 is T.

8) Implement BetterChange (from Pevzner, pp. 21). This is the USChange problem but for any denomination. Note that M is a float, **c** is a vector of denomination values, and d is a positive integer corresponding to the number of denominations.

```
BETTERCHANGE(M, c, d)
1 r ← M
2 for k ← 1 to d
3   i_k ← r/c_k
4   r ← r - c_k * i_k
5 return (i_1, i_2, …, i_d)
```

   a) Implement the above method in python.
   b) Compute BetterChange() with M = 0.79, **c** = [1, 3, 9], and d = 3.
   c) What is the runtime of BetterChange as a function of M, **c**, and d?
   d) Provide a case (specify a value of M, **c**, and d) where BetterChange is correct.
   e) Provide a case (specify a value of M, **c**, and d) where BetterChange is incorrect.

9) *The Birthday Problem.* Write a program that takes an integer N and uses `np.random.randint(low = 0, high = N)` to generate a random sequence of integers between *0* and *N-1*. Run experiments to validate the hypothesis that the number of integers generated before the first repeated value is found is ~sqrt(pi * N / 2).

10) *The Coupon Collector Problem*. Generating random integers as in the previous exercise, run experiments to validate the hypothesis that the number of integers generated before all possible values are generated is ~$N\mathrm{H}_N$. Note that $\mathrm{H}_N = 1 + ½ + ⅓ + \ldots + 1/n \sim \ln(n)$.