EE4 2005
University of Iceland

Hannes Jónsson
Kristján Alexandersson

Classical trajectories

In this exercise, you will use the velocity Verlet algorithm to calculate classical trajectories. First of all (in part A), you will gain familiarity with the algorithm and study its properties by applying it to the dynamics of a diatomic molecule. You will then (in part B) study the collision of an atom with a diatomic molecule to see, in particular, whether an exchange reaction takes place. Two different potential surfaces will be used to describe the interaction between the atoms, one could be used to describe interaction between rare gase (and possibly metal) atoms while the other describes a situation where only one chemical bond can be formed in the system, for example halogen and/or hydrogen atoms. You will see how features of the potential energy surface affect the dynamics of an exchange reaction. In some cases the probability of an exchange reaction is increased more when energy is added to the vibrational degree of freedom rather than the translational degree of freedom. In other cases, the translational degree of freedom enhances the reactivity more than the vibration.

A: Diatomic molecules

A1. The Lennard-Jones pair potential

The simplest and probably most frequently used potential function for describing the interaction of two atoms is the Lennard-Jones (L-J) potential

$$v(r) = 4\varepsilon((\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6})$$

where r is the distance between the two atoms. The zero of energy is chosen here to equal the limiting value of the potential energy as the atoms are brought infinitely far apart, v(r) -> 0 as r -> ∞. The attractive part of the potential (the second term) is consistent with a London interaction (induced dipole - induced dipole), but the form of the repulsive part (the first term) is chosen to be the 12-th power of the inverse distance merely for mathematical convenience. It would be more realistic for most molecules to use an exponential form for the repulsive part but it leads to a problem at very short distances since the diverging $-1/r^6$ attractive part will eventually dominate, leading to a collapse of the two atoms. There are ways around this, but they make the potential form much more complicated. The L-J potential function describes quite well the interaction between rare gas atoms, such as Ar-Ar. The Morse potential which has exponential attractive part and exponential repulsive part is more appropriate for describing the interaction between atoms forming a chemical bond. We will use the L-J potential here and, first of all, we will use Matlab to gain more familiarity with the L-J potential function. (The Morse potential is used in next weeks exercise). First, define a Matlab function for the potential. It is convenient to choose units of energy and length in such a way that both parameters in the L-J potential, ϵ and σ, get the numerical value of unity.

```
---------------------------------------------------------
function y=potent(r)
epsilon=1;
sigma=1;
y=4*epsilon*((sigma./r).^12-(sigma./r).^6);
---------------------------------------------------------
```

The potential can then be plotted with the plot function

```
>> r=[0:0.02:3];
>> plot(r,potent(r))
>> axis([0 3 -2 6])
```

Note that epsilon is the well depth (here -1) and sigma is the distance at which the potential is zero. What is the optimal distance between the two atoms? The optimal distance corresponds to minimum potential

energy. We can use Matlab to find it.

```
>> xmin=fminbnd(@potent,0.6,1.5)
```

Here @ is a function handle. It is used to retrieve all the information about a function that Matlab needs.

Q1: Verify that this is indeed the value that gives the local minimum, by differentiating the potential function by hand and then making it equal to zero.

The force on the atoms reflects the tendency to move towards lower potential energy. For any distance, r, the force is simply the negative derivative of the potential function. It can be convenient to use Matlab to take the derivative and define a force function. While the L-J potential is simple enough that this can as easily be done by hand, more realistic potential functions can be quite complicated and the process of taking the derivative tedious. You will see an example of that in part B. We first take the derivative here with respect to a symbolic variable rr and then replace rr with the value of the distance, r, for which the force should be evaluated

```
--------------------------------
function y=force(r)

syms rr

difansw=-diff(potent(rr));

rr=r;

y=subs(difansw);
--------------------------------
```

It is a good idea to test and see if the force function has been defined correctly by asking Matlab what the form of the function is by typing (compare this with the expression for the derivative in Q1)

```
>> syms r
>> force(r)
```

Q2: Plot the force and the potential on the same graph. Note that the force is zero at the potential minimum. Identify which curve is which.

Q3: Imagine initially holding the atoms at rest with a separation of 1.0. How will the atoms move when they are released? What is the sign of the force when the distance between them is 1.0? Is it consistent with your answer to the first question (a positive force will act to increase the distance, a negative force will act to decrease the distance).

As the symbolic differentiation takes a lot of processing time, it is prudent to define a new force function.

```
--------------------------------------------------------------------------
function y=force(r)

epsilon=1;

sigma=1;

y=-4*epsilon*(-12*sigma^12*r.^-13+6*sigma^6*r.^-7);
--------------------------------------------------------------------------
```

The task is now to calculate the classical trajectory of the two atoms. No forces act on the center of mass and we take it to be stationary, so the goal is to get the distance between the two atoms, the relative coordinate, as a function of time, r(t). The method is based on discretizing time and using an iterative finite difference algorithm, the so called velocity Verlet algorithm (see lecture notes). Time is advanced by one time step for each iteration of the algorithm. It is a good idea to clear all the variables so that we start off with a clean slate.

```
>> clear all
```

First of all, the time step size is defined, h, and total time specified, from which the total number of time steps can be calculated.

Secondly, the initial values of the distance, the force and the velocity can be specified. Here a distance of r = 2 and a velocity of - 0.1 are chosen

Then, the velocity Verlet algoritm can be applied 'nsteps' times to generate the distance, force and velocity at time points n*h:

```
---------------------------------------------------
h=0.02;
totaltime=10.0;
nsteps=round(totaltime/h)+1

r(1)=2;
f(1)=force(r(1));
v(1)=-0.1;

for n=2:nsteps
   r(n)=(r(n-1))+h*v(n-1)+h^2*f(n-1)/2;
   f(n)=force(r(n));
   v(n)=v(n-1)+h*(f(n)+f(n-1))/2;
end
---------------------------------------------------
```

It can be instructive to list the values of the various quantities for each time step. First, evaluate the kinetic energy from the velocity (the unit of mass is chosen here to give unit reduced mass for the two atom system) and then the total energy

```
>> kineticen=0.5*v.^2;
>> totalen=kineticen+potent(r);
```

It is informative to plot the data. First of all, a plot of the distance between the two atoms as a function of time

```
--------------------
t=[1:nsteps]*h;

plot(t,r)
xlabel('t')
ylabel('r')
--------------------
```

Q4: Interpret the shape of the r vs. t curve in terms of the motion of the atoms. Why is the curve pointed for small r and smooth for large r?

The motion of the two atoms can be animated (animations will become even more useful later when you simulate collisions involving three atoms). Choosing the origin to be right in between the two atoms, the position of atom A is -r/2 and the position of atom B is r/2. The following commands will produce the animation.

```
-------------------------------------------------------------------
nres=5;

% This loop picks out every fith set of coordinates
for n=1:nres:nsteps
   ptA((n-1)/5+1)=-r(n)/2;
   ptB((n-1)/5+1)=r(n)/2;
end
```

```
nanimate=round(nsteps/nres);

for t=1:nanimate
   plot(ptA(t),0,'ob',ptB(t),0,'ok','MarkerSize',69)
   axis([-3 3 -3 3])
   axis off
   M(t) = getframe;
end

movie(M,5)
```

The trajectory gets repeated over and over again. Since the distance and velocity at the end of the trajectory happened to be the same as at the beginning of the trajectory in this case, the repeated animations give the impression of seemles continuation.

The most important check on a numerical calculation of a classical trajectory is the conservation of the total energy. If Newton's equation is satisfied, then the total energy is constant. This will only be true approximately in the finite difference calculation (recall the higher order terms in the Taylor expansion that were neglected in the derivation of the Verlet algorithm) but the energy conservation will become better and better as the time step size is made smaller (until round-off errors start showing up for very small step size).

Q5: Make a plot of the total energy as a function of time. Note that any deviation from the initial total energy is an error in the energy. For what configuration of the atoms is the error largest? Recall the derivation of the Verlet algorithm with Taylor expansions. When is the neglected higher order term largest?

Notice the remarkable stability of the Verlet algorithm. An error made at one step tends to decrease in later steps. In contrast, an unstable algorithm will tend to magnify errors at later steps.

Q6: Repeat the calculation of the trajectory using a smaller time step size, for example half as large. Then plot the total energy again. How does the error in the energy change as the step size is changed?

It turns out that the global error in the energy, i.e. the error over an extended interval in time, goes as the second power of the step size. That is, if a trajectory spanning some total time T is calculated for different time step sizes, then a plot of the error made (for example the root mean square error in the energy over the whole trajectory) vs. time step size squared would be linear. This is a good test to see if the Verlet algorithm has been programmed correctly. (Optional: carry out this test).

Q7: Repeat the calculation of the trajectory for a higher energy. One way to increase the energy is to increase the initial velocity. Try for example an initial velocity of -0.2 and -0.5. Check the energy conservation (plot energy vs. time). Note that the energy is less well conserved the higher the energy is. This means that you need to choose a smaller time step size at larger energy in order to have similar accuracy in the velocity Verlet algorithm. Why is that?