Computer Modelling Exercise 3

Time Integration

Due: 16:00 Monday, Week 10, Semester 1

1 Aims

In this exercise, you will reuse your Particle3D class from Exercise 2 to write two simple time integrators, to simulate O_2 and N_2 molecules under a variety of initial conditions.

You will find that particles oscillate back and forth (vibrate), and you will measure the frequency of these vibrations and compare them to experimental data.

The goal is to write a short document collecting your results and assessing your code's performance.

2 Preparation

- Check the feedback you got for exercise 2, and fix any methods that did not work in your `Particle3D` class.
- Make sure you understand the symplectic Euler and velocity Verlet time integration algorithms described in the background material and the lecture. Ask questions if not.
- Read through section 3 of these instructions and make some notes about parts which you think may be most important.
- Download these files from Learn to your working directory (the first will only be available after submitting an exercise 2 attempt):
  `particle1D.py, simulate_particle.py, oxygen.dat`.
- in the Anaconda ipython console window, cd to your folder and type:
  `%run simulate_particle.py euler euler.dat`

Something wrong with first one

The output file, `euler.dat`, will have three columns of numerical data, each representing the time, position and energy of the simulation. These are easy to plot using Excel, LibreOffice, or your favourite software. On the School Linux machines, the file can be plotted using Grace: `xmgrace euler.dat`

Do I need to?

What is wrong with the python plots?

Make sure you understand the codes and confirm that both algorithms can simulate the motion of a particle in a double well potential.

3 Theory

This section goes through the theoretical context of the work we will do here. Read it now and make some notes, then refer back to it and them later.

3.1 The Morse potential

The particles in your simulation will interact with each other via the *Morse* potential. The Morse potential is a commonly used pair-potential to describe covalent bonds in molecules or clusters. Its expression for the energy of two particles at $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ is

$$U_M(\boldsymbol{r}_1, \boldsymbol{r}_2) = D_e \left\{ \left[1 - e^{-\alpha(r_{12}-r_e)}\right]^2 - 1 \right\}, \tag{1}$$

where $\boldsymbol{r}_{12} = \boldsymbol{r}_2 - \boldsymbol{r}_1$ and $r_{12} = |\boldsymbol{r}_{12}|$. The force on the particle at $\boldsymbol{r}_1$ is then:

$$\boldsymbol{F}_1(\boldsymbol{r}_1, \boldsymbol{r}_2) = 2\alpha D_e \left[1 - e^{-\alpha(r_{12}-r_e)}\right] e^{-\alpha(r_{12}-r_e)} \hat{\boldsymbol{r}}_{12} \tag{2}$$

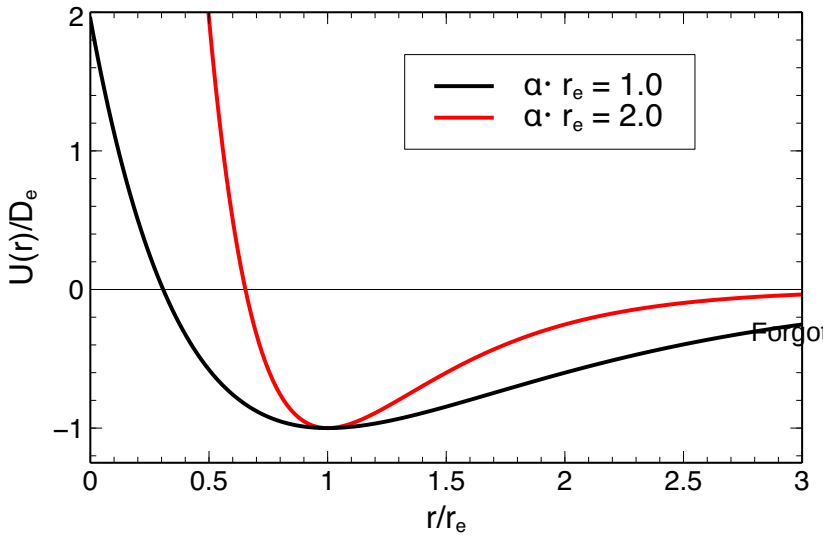
and $\boldsymbol{F}_2 = -\boldsymbol{F}_1$. Here, $\hat{\boldsymbol{r}}_{12} = \boldsymbol{r}_{12}/r_{12}$. The parameters r_e, D_e, and α control the position, depth and curvature of the potential minimum, respectively, and are depend on what type of atom is being simulated. Figure 1 shows two example cases.

3.2 Units

In computer simulations, it is often a good idea to adapt the units to the problem at hand. This minimises precision loss due to rounding off errors. In atomic or molecular physics, one suitable choice would be eV for the energy, Å for the length, and amu (atomic mass units) for the mass.

Given this choice, the unit of time is implicitly defined and completely fixed. Since energy has dimension $[E] = [M][L]^2[T]^{-2}$ we can re-arrange to find that our time unit $[T]$ is $\text{Å}\sqrt{\text{amu/eV}}$. This may be evaluated using CODATA, the current accepted best

Do I need to evaluate?



Forgot how to understand this figure

Figure 1: The Morse potential, in relative distance (r/r_e) and energy (U/D_e) coordinates, and for two different values of $\alpha \cdot r_e$.

values of physical constants. In this case, $[T]$ is $1.018050571 \times 10^{-14}$ seconds. When our time parameter is $t = 1$, this much physical time has lapsed in the simulation.

We will use these units in this program; this mainly matters when we compare results to expected values - the numbers we use below are already in these units, so you don't have to convert anything.

3.3 Measurements

Oscillation in atoms is measured by observing their photon emission and absorption, and is often described by the "wavenumber", which here means $\bar{\nu} = \nu/c$ where ν is the usual definition of frequency[1]. This is often measured in units of cm^{-1}.

3.4 Accuracy

A perfect simulation would perfectly conserve energy. Our simulations will not quite do so - the energy will oscillate up and down. The more inaccurate our simulation, the bigger those oscillations will be. Monitoring these energy fluctuations allows us to estimate how accurate our simulation is. We can measure them in terms of:

should we not look at energy for whole time?

$$\Delta E = \max(E) - \min(E) \tag{3}$$

[1]Note the different meaning to the usual wave-number k. They are different by a factor of 2π.

To be able to compare different simulations, we can use the value $\Delta E/E_0$ where E_0 is the initial energy. It is also important to keep track of the energy as a function of time, as opposed to merely knowing the maximum fluctuation, as this would show if the simulation energy drifts over time, or if any particular event breaks the simulation.

Similarly, we can see how our estimates of the wavenumber change in different quality simulations. Taking our best possible estimate as ν_0, and the value for any one simulation as ν, we measure inaccuracy using:

$$\frac{\Delta\nu}{\nu_0} \equiv \frac{\nu - \nu_0}{\nu_0}. \tag{4}$$

It is always a good idea to know how our computational set up influences the results we want to obtain. This is even more important when we want to compare our results to other theories or experiments.

4 Coding Tasks

You are expected to write *two* simulations, implementing the symplectic Euler and velocity Verlet time integrators respectively, to model the evolution of two interacting particles.

Document all the code you write carefully as you go along. You should include doc-strings (triple-quoted strings) at the top of all your functions and methods describing their purpose, inputs, and outputs, and in-line comments (starting #) through the code describing sensibly what you are doing.

You will also be marked on the clarity and elegance of your code and comments, and your use of sensible numpy.

4.1 Writing your time integrator

The two supplied example files, `Particle1D.py` and `simulate_particle.py` are reasonable starting points.

Task 1a Modify the codes to do 3D particle simulation with the Morse force instead of the double well. You will need to:

- Use your `Particle3D` from exercise 2 instead of `Particle1D`.
- Change the force and potential functions to take two particles as arguments, use the parameters α, D_e, r_e instead of a, b, and return the Morse force and potential instead of the double well.

- Change the integration loop to operate on both particles, not just one. **Critically**, you have to update both particle positions before updating their velocities. You should do this for both the Euler and Verlet versions.
- Correctly compute the total energy of the system.
- Use Newton's 3rd law instead of calculating the force twice (this makes the code a lot faster).

Task 1b Change the output so that it saves and plots the distance between the two particles instead of the position of just one of them. Save the plots generated by the code.

Saving the data to file should allow you to compare several simulations, and produce graphs using your program of choice.

one file for all different values or different files for each run?
do I save the plots i this case too?

4.2 Basic tests

Task 2 Test your integration works by running it with these two particles, which simulate an Oxygen molecule (the units are already in the ones we are using). One particle should use the + sign and one the − sign:

$$\begin{aligned} m &= 16.0 \\ \boldsymbol{x} &= [\pm 0.65661, 0, 0] \\ \boldsymbol{v} &= [\pm 0.05, 0, 0] \end{aligned}$$

Use these force parameters:

$$\begin{aligned} D_e &= 5.21322 \text{ eV} \\ r_e &= 1.20752 \text{ Å} \\ \alpha &= 2.65374 \end{aligned}$$

and these simulation parameters:

$$\begin{aligned} \delta t &= 0.01 \ [T] \\ N_{\text{step}} &= 10000 \end{aligned}$$

Your plot of the separation versus time should show oscillations with a period of a few time units. Your plot of the energy should show complicated periodic oscillations which for the Verlet are small compared to their average value.

4.3 Measuring the wavenumber and energy variation

Task 3a Work out (mathematically, not in your code) a relation between the period of an oscillation and the value $\bar{\nu}$ described above using your knowledge of wave quantities.

Task 3b Write code to measure the period of the oscillations of the particles. This could be done using the function `scipy.signal.find_peaks`. Read the online documentation for this function to learn how to use it, and write a new function in your code to:

- compute the period,
- convert the period from the units described in section 3.2 to seconds,
- convert it from there to $\bar{\nu}$, and return it in cm^{-1} units.

Call this function in your code to measure the frequency of oscillations in an oxygen atom, and print it out clearly so a new user can understand what it is showing.

Task 3c Modify your code to also report the print the energy inaccuracy $\Delta E/E_0$ for a simulation run.

5 Using your code for a computational experiment report

but we don't have N2

Now your code is written, you can use it to perform computational "experiments". The final task of this exercise is collecting a set of results of such an experiment: obtaining the vibrational frequency of N_2 and O_2 molecules interacting via the Morse potential.

You should submit a collection of plots and related information from the subsection below in your report. All graphs should be as high quality as possible, with labels, units, legends, and be clearly readable.

Your report should be around 2 pages long; a maximum of 3 pages will be marked.

5.1 Convergence

You should first determine the ideal time-step parameter δt. If this parameter is too high, the simulation will be inaccurate. The smaller it is, the longer the simulation takes to run. You run your code changing the δt manually each time, or write a version of your code that loops through different values automatically.

Simulating the same absolute time, and no less than 5 oscillations, run both integrators simulations with a wide range of δt's, e.g. 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0.1 and 0.5. In each case, record the energy inaccuracy, and estimate the frequency inaccuracy, taking ν_0 from your simulation with the smallest δt.

Plot your results[2] showing these as a function of δt, and record your observations ina few paragraphs. You should also discuss these questions: how does the frequency change? What about the energy? Are there qualitative differences?

State your estimate, for both time integrators, the maximum time step δt_{max} that can be used to simulate the system with a relative frequency error under 0.5%. From now on, use this time step in your simulations.

Do I have to include this side quest as part of my report?

5.2 Nitrogen & Rotating Molecules

Make and use a modified input file to run the simulation for Nitrogen instead of Oxygen. Nitrogen has these parameters:

$D_e = 9.90523,$	$r_e = 1.09768,$	$\alpha = 2.68867$
$m = 14.01,$	$\boldsymbol{x} = [\pm(r_e/2 + 0.05), 0, 0],$	$\boldsymbol{v} = [\pm 0.05, 0, 0]$

Record your measurements for Nitrogen in your document, including a plot of the oscillation.

Make another modified input file to put a spin on both the Oxygen and Nitrogen molecules by setting the initial velocities to $\boldsymbol{v}_1 = (0.05, 0.3, 0.0)$ and $\boldsymbol{v}_2 = (-0.05, -0.3, 0.0)$. Record these in the document too.

Why tho? and this is not gonna fit in 2 pages

5.3 Comparisons

Based on the simulations above, write a paragraph in your document stating which integrator is more accurate.

Write a final paragraph comparing your results to experimental values, $\bar{\nu}(\mathrm{N}_2) = 2359\ \mathrm{cm}^{-1}$ and $\bar{\nu}(\mathrm{O}_2) = 1580\ \mathrm{cm}^{-1}$, and explaining the numerical difference between purely vibrational and rotational frequencies, and its physical origin.

[2] Think about what kind of plot would be easiest to read here

isnt there just one option?

6 Submission

Package the subdirectory that contains an up-to-date `particle3D.py`, your simulation programs, input files, and your report in a zip file. See the instructions below specific to your operating system.

You should submit the compressed package through the course LEARN page, by **16:00 on Thursday, week 10 of semester 1**.

6.1 Compressing your submission directory

Windows Right-click on the folder, and under the "Send-to" menu, select "Compressed (zipped) folder".

Mac Ctrl-click on the folder and click "Compress name_of_folder".

Linux Run this in a terminal: `zip -r ex3.zip project_directory_name`

and verify the command ran without errors. Check that all the files you want are included by running: `unzip -l ex3.zip`

7 Marking Scheme

This assignment counts for 10% of your total course mark.

1. Particle Simulations [10]
   - Force and energy laws implemented correctly [3]
   - Symplectic Euler and velocity Verlet implemented correctly [2]
   - Frequency function works correctly [2]
   - Code Quality: layout, comments, naming [3]
2. Report [10]
   - Time-step plots and dt estimate [2]
   - Frequency measurements [2]
   - Plots of particle separations [3]

- Comparisons [3]

Total: 20 marks.

The baseline for Code Quality for a reasonably written, working code, with up-to-date comments and mostly consistent naming and whitespacing is **2 points**.