NB: The graded, first version of the report must be returned if you hand in a second time!

H1a: Classical scattering by a central potential

Victor Nilsson and Simon Nilsson

November 15, 2016

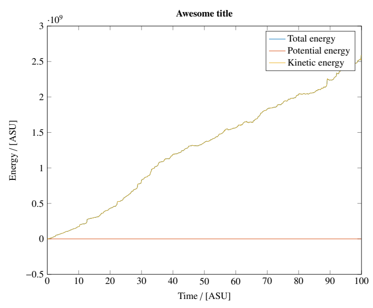
Task №	Points	Avail. points
Σ		

Introduction

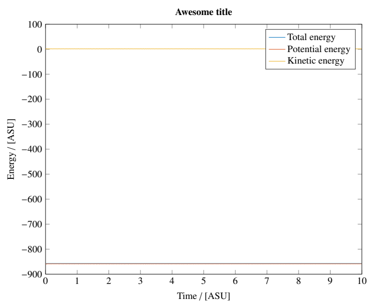
Already in antiquity people studied the effect of particles impinging on other particles. Since then the art has developed . . . (*If you like to do so, you may take the opportunity to put the methods in a wider perspective here.*) Here is a random reference.[1]

Problem 1

Bilder tagna i Linkping



(a) Energy, timestep 0.1



(b) Energy, timestep 0.01

As a starting point we first look at scattering from a hard-sphere potential. We also consider the Lennard–Jones potential, which is depicted in Figure 2. (*Always refer to Figures in the text.*)

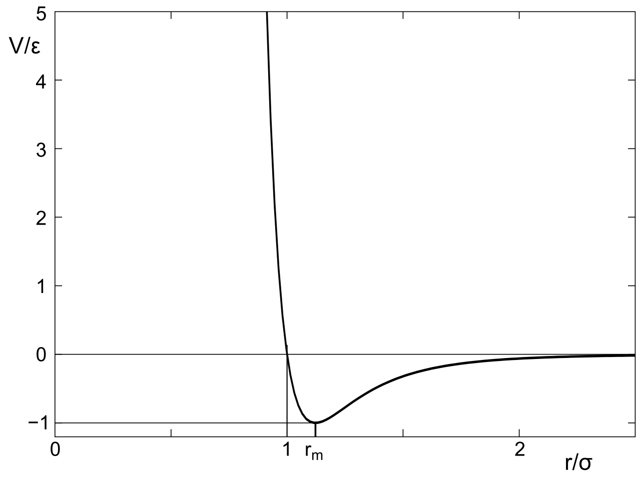


Figure 2: The Lennard–Jones potential. Make sure you label and have units on all axes! Also make sure that labels etc. are legible and that, if you print in black and white, that you use different line styles when required to differentiate between curves. In MATLAB you can export any figure to an .eps file from File → Export. . . in the Figure window.

Problem 2

In the following we give an example of how to produce a table. Use the code for Table 1 as a template.

Problem 3

If you find some part of the code particularly interesting you may include it in the text, otherwise it should be included in the appedix. If you do want to include code the following commands will print the text directly, with no LaTeX commands executed:

Table 1: A dummy table

Col. 1	Col. 2	Col. 3
the	quick	brown
fox	jumps	over
the	lazy	dog

```
% Hello world ten times in MATLAB
for i = 1 : 10
  fprintf('Hello world %d!\n',i);
end
```

```
# Hello world ten times in Python
for i in range(10):
  print 'Hello world %d!' % i
```

Problem 4

At some point it may be appropriate to include equations. It is done in the following way:

$$V(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{1}$$

Do number and reference all your equations.

Concluding discussion

Use your favourite flavor of LaTeX to compile the file:

```
xelatex template.tex
pdflatex template.tex
latex template.tex
```

should all work. If you use `pdflatex` or `xelatex`, included figures need to be in `pdf`, `jpg`, or `png` format. If you want to include eps figures, you can easily convert them to `pdf` using the command

```
ps2pdf -dEPSCrop figure.eps figure.pdf
```

References

[1] Leslie Lamport, *LaTeX: A Document Preparation System*. Addison Wesley, Massachusetts, 2nd Edition, 1994.

A Source Code

Include all source code here in the appendix. Keep the code formatting clean, use indentation, and comment your code to make it easy to understand. Also, break lines that are too long. (Keep them under 80 characters!)

A.1 Calculating pi using matlab: `pi.m`

```matlab
% Generate random points inside a centered square of area 4
trials = 1e6;
x = 2*rand(trials,1) - 1;
y = 2*rand(trials,1) - 1;

% Distance of points to (0,0)
r = sqrt(x.^2 + y.^2);

% Fraction of points inside unit circle
frac_inside = sum(r < 1)/trials;

fprintf('Pi is approximately %.6f.\n', 4*frac_inside);
```

A.2 Calculating pi using python: `pi.py`

```python
#!/usr/bin/env python

from pylab import *

# Generate random points inside a centered square of area 4
trials = 1e6
x = 2*rand(trials,1) - 1
y = 2*rand(trials,1) - 1

# Distance of points to (0,0)
r = sqrt(x**2 + y**2)

# Fraction of points inside unit circle
frac_inside = sum(r < 1)/trials

sys.stdout.write('Pi is approximately %.6f.\n' % (4*frac_inside))
```

A.3 Calculating pi using C: `pi.c`

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#define TRIALS 1e8

int main() {
  float* x = malloc(TRIALS*sizeof(float));
  float* y = malloc(TRIALS*sizeof(float));
  int inside = 0;

  /* Seed RNG with current time */
  srand((unsigned)time(NULL));

  /* Generate random points inside a centered square of area 4
     and count the proportion that falls within the unit circle. */
  int i;
  for (i = 0; i < TRIALS; i++) {
    x[i] = 2.0 * (rand()/(RAND_MAX+1.0)) - 1.0;
    y[i] = 2.0 * (rand()/(RAND_MAX+1.0)) - 1.0;
    if (sqrt(x[i]*x[i]+y[i]*y[i]) < 1.0) {
      inside++;
    }
  }

  free(x);
  free(y);

  printf("Pi is approximately %.6f.\n", 4.0*inside/TRIALS);

  return 0;
}
```