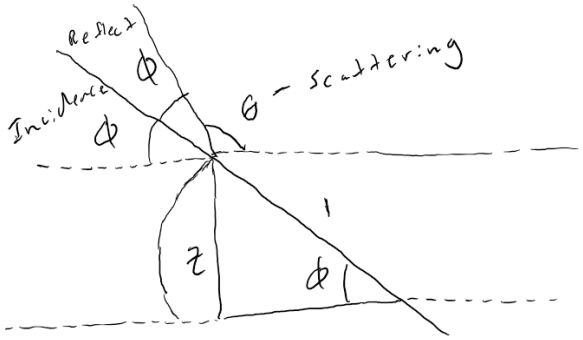
PHY407H1 Lab 1

Eric Yeung*
Department of Physical Sciences, University of Toronto, Toronto M1C 1A4, Canada
(Dated: September 21, 2015)

QUESTION 1

a) Using the law of reflection, the angle of reflection and incidence are equal $\phi_1 = \phi_2 = \phi$. I drew a figure to make my thought process a lot clearer.



Because there exists a transversal, we have a pair of angles (ϕ and ϕ) in each of the mini triangles.

$$\begin{aligned}
\pi &= \theta + 2\phi \\
\phi &= \frac{1}{2}(\pi - \theta) \\
\sin(\phi) &= \frac{\mathrm{z}}{1} = \mathrm{z} \\
\phi &= \arcsin(\mathrm{z}) \\
\frac{\pi - \theta}{2} &= \arcsin(\mathrm{z}) \\
\pi - \theta &= 2\arcsin(\mathrm{z}) \\
\theta &= \pi - 2\arcsin(\mathrm{z}) \text{ As expected.}
\end{aligned}$$

A function for the scattering angle as a function of height was written.

```
theta = lambda z: pi - 2*asin(z)
```

If z = 0.25, $\theta = 2.64$ radians $= 151°$ which makes sense as it is an obtuse angle.

b)

Pseudo-code submitted separately. See lab1_q1b.py.

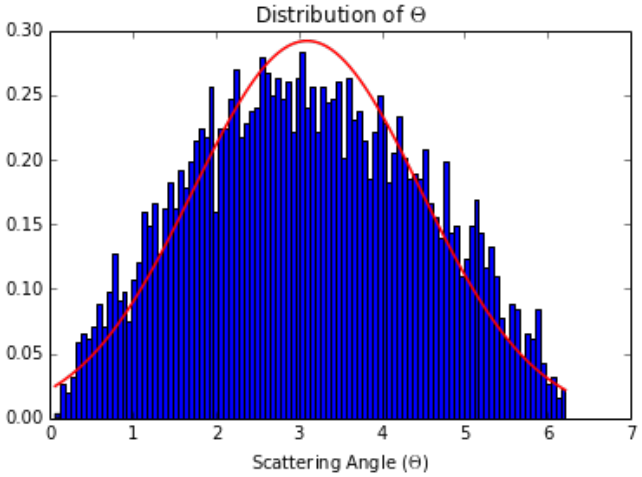
c)
Python code submitted separately. See lab1_q1c.py
It does resemble a uniform distribution to be expected since the heights were generated randomly (or pseudo-randomly). The relative probabilities at N=1000 are around 7.7% and 6.7% for the range(170,190) and range(90,110) respectively.

* eric.yeung@mail.utoronto.ca

```
The relative probability of finding the particle in the range(170,190) is 0.077 for N = 1000
The relative probability of finding the particle in the range(90,110) is 0.067 for N = 1000
```

After doubling the particles, the relative probabilities at N=2000 are around 7.65% and 6.05% for the range(170,190) and range(90,110) respectively. As we can see, the number doesn't change much after doubling the number of particles.

```
The relative probability of finding the particle in the range(170,190) is 0.0765 for N = 2000
The relative probability of finding the particle in the range(90,110) is 0.0605 for N = 2000
```

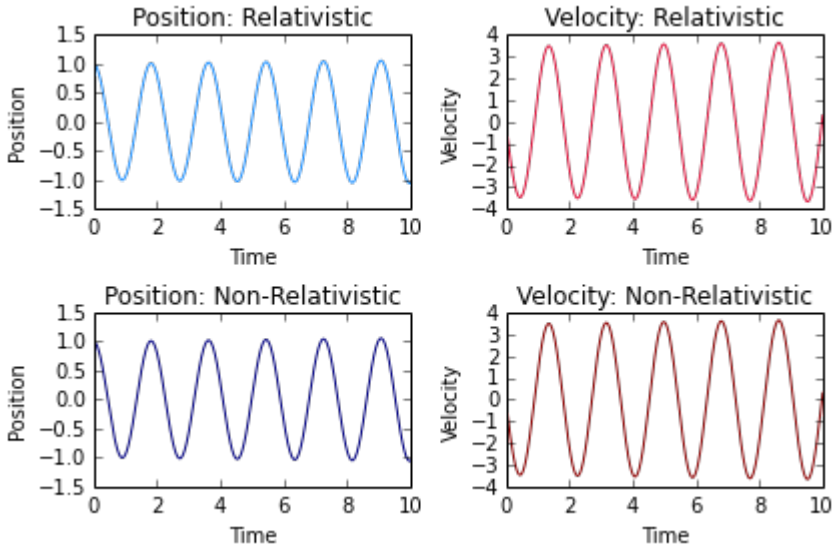


QUESTION 2

a) Using reduction of order, we end up with two coupled/simultaneous ODEs as shown in the assignment page. Pseudo-code submitted separately. See lab1_q2a.py

b)

Follows from pseudo-code, see lab1_q2b.py



It turns out that the relativistic effects are not important at all in this problem as one can see by the graph. To further illustrate the minuscule difference, I let the system evolve until the last time increment and then evaluate the entry for x[], xn[], v[], and vn[].

```
print x[9999] - xn[9999] # Let the system evolve until the limit (1e4)
print v[9999] - vn[9999]

1.33226762955e-15
-1.23789867246e-14
```

As you can see, the differences are of a magnitude of $\mathcal{O}(10^{-14})$.

c) We want to find x_c such that $\frac{dx}{dt}(x_c) = c$. We first solve the trivial differential equation for simple harmonic motion $m\ddot{x} = -kx$ with known solution:

$$x(t) = C_1 \cos(\omega t) + C_2 \sin(\omega t) \tag{1}$$

We have two initial conditions, the particle is still initially at rest $v(0) = 0$ and the new initial position $x(0) = x_c$. Using the first initial conditions trivially yields:

$$\begin{aligned} x'(0) = 0 &= -C_1\omega \sin(\omega \times 0) + C_2\omega \cos(\omega \times 0) \\ 0 &= C_2\omega \\ \rightarrow C_2 &= 0 \end{aligned}$$

For the second initial condition,

$$\begin{aligned} x(0) = x_c &= C_1 \cos(\omega \times 0) \\ \rightarrow C_1 &= x_c \end{aligned}$$

Our solution is now:

$$x(t) = x_c \cos(\omega t)$$

We know that the speed of the mass is maximum at the equilibrium point (or at different values of n), so we want to find the time at which $x(t) = 0$.

$$\begin{aligned} x(t) = 0 &= x_c \cos(\omega t_c) \\ \cos(\omega t_c) &= 0 \\ \omega t_c &= n\pi - \frac{pi}{2} \\ \rightarrow t_c &= \frac{1}{\omega}\left(n\pi - \frac{pi}{2}\right) \leq 10 \end{aligned}$$

Note that we have to impose an additional condition on the time such that $t_c \leq 10$, and subsequently $n \leq 11$, as 10 seconds is our final time.

We also calculate $\omega = \sqrt{k/m} = \sqrt{12}$ for convenience. We want only even values of n, since that would yield a position value of x_c as shown below.

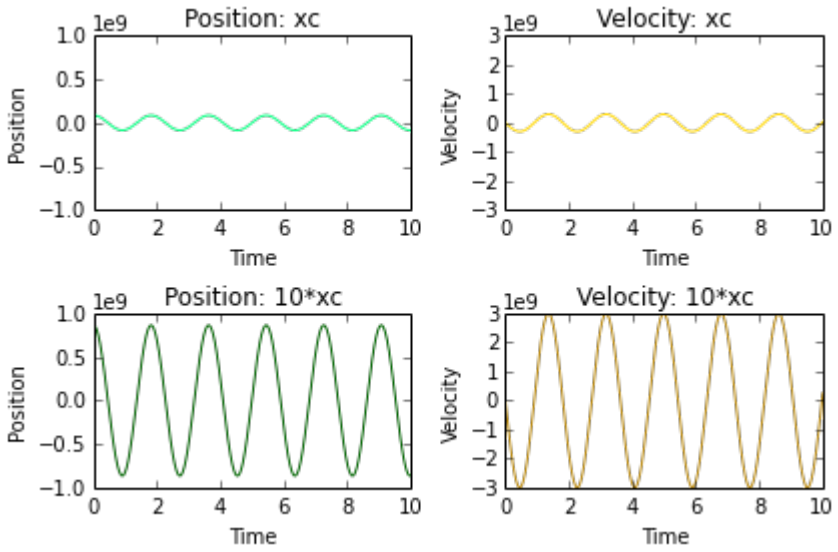
Let $n = 2 \rightarrow t_c = \frac{\sqrt{3}\pi}{4}$.

$$x'(t_c) = c = -x_c\sqrt{12}\cancel{\sin(\sqrt{12} \times \frac{\sqrt{3}\pi}{4})}^{-1} \tag{2}$$

$$x_c = \frac{c}{\sqrt{12}} \tag{3}$$

$$\rightarrow x_c = 8.654 \times 10^7 \text{metres} \tag{4}$$

For the code used to find the new solution, see lab1_q2c.py.



The period of the oscillation does not increase nor decrease. This makes sense since we haven't changed parameters m and k. From the graphs, the period looks like it's around 2. Calculating the period manually yields $T = 2\pi\sqrt{\frac{m}{k}} = 1.814$ which is close to our qualitative estimate. The time series does not change at all. The only thing that changes is the amplitude, which makes sense because we changed the initial conditions.

```
print x[999999] - xnew[999999] # Let the system evolve until the limit (1e6)
print v[999999] - vnew[999999]

776614650.424
-225059139.96
```

Unlike with the question above, there is a major difference when we fully evolve the system. That is because, in this question, the relativistic effects are very significant. The difference is of the order $\mathcal{O}(10^8)$

$40 * x_c/c$ yields approximately 11.54 which does not give a good estimate of the period. It is around 10 times too much. However, $4 * x_c/c = 1.15$ gives a better approximation of the period, although it is still about 2 times less than the observed/calculated one.

$4 * x_c/c$ is a good estimate of the period because it was obtained in the process of solving the differential equation, see [Eq. 3] done above. We just do some algebra manipulation.

$$
\begin{aligned}
x_c &= \frac{c}{\sqrt{12}} \\
\frac{x_c}{c} &= \frac{1}{\sqrt{12}}, \text{ Multiply each side by 4 to get our "estimate"} \\
\frac{4x_c}{c} &= \frac{4}{\sqrt{12}} \\
\frac{4x_c}{c} &\approx 1.15
\end{aligned}
$$

QUESTION 3

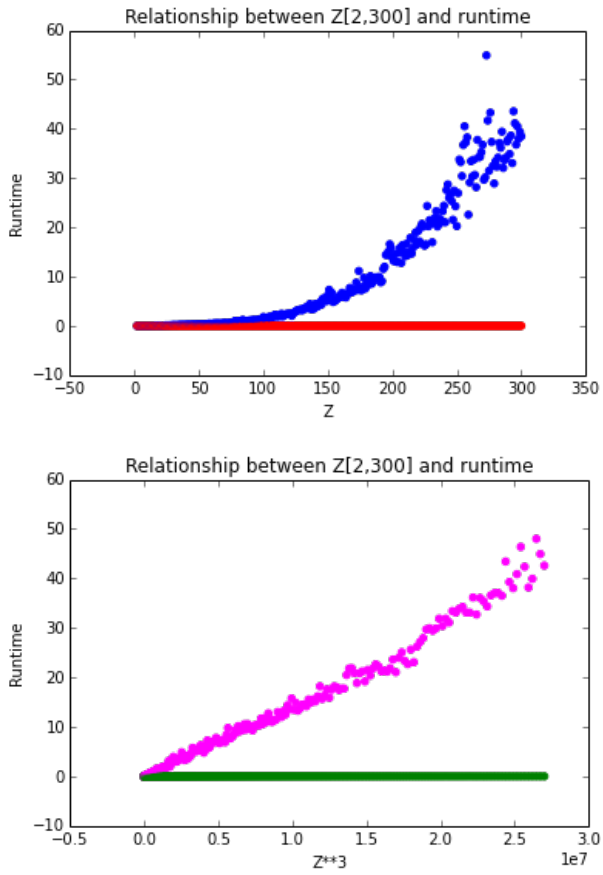


FIG. 1: Computation times plotted against Z and Z^3 respectively.

Code found in lab1_q3.py. I notice that, when plotting Z vs computation time, the time for manual matrix multiplication grows exponentially while the time for the np.dot function is pretty much constant. In terms of time complexity, it is a comparison between $\mathcal{O}(n^3)$ vs $\mathcal{O}(1)$ for the normal multiplication and the numpy operation respectively.