Computational Physics: Simple Harmonic Motion

Ujwal Kumar

Jan 27, 2021

1 Introduction

The restoring force exerted by springs is key to engineering pens and cars alike. This restoring force is given by:

$$F = -k|r|\hat{\mathbf{r}} \tag{1}$$

Where k is the spring constant and $\mathbf{r}$ is the vector pointed from the equilibrium point of a segment of a spring to the compressed or expanded point of that same segment. Equation 1 describes the force that restores the spring to equilibrium after a compression or expansion by distance, r.

Expanding this idea to Cartesian coordinates requires us to resolve this force into x and y components, $F_x = -kx$ and $F_y = -ky$. The motion that then occurs due to this force is given by Newton's laws of motion in differential form:

$$\frac{d^2x}{dt^2} = -\frac{k}{m}x$$
$$\frac{d^2y}{dt^2} = -\frac{k}{m}y$$

And the solutions are given by:

$$x(t) = A_x sin(\omega t + \phi_x) \tag{2}$$
$$y(t) = A_y sin(\omega t + \phi_y) \tag{3}$$

Where A_x and A_y are amplitudes in the x and y direction, $\omega = \sqrt{k/m}$ is the frequency of oscillation, and ϕ_x and ϕ_y are the shifts in phases of each wave. Equations 2 and 3 thus give us the coordinates of the trajectory of a particle subject to this simple harmonic motion.

To analyze these trajectories, we build a code in C that inputs values for A_x, A_y, ϕ_x, and ϕ_y and outputs coordinate pairs. We then export this data to Mathematica to plot these trajectories in Cartesian space and describe these results in this report.

2 Results

We can recognize the oscillatory motion of the mass through the sin functions described in 2 and 3. We can explore the plots for $A_x = 1, A_y = 1$ and $\Delta\phi = 0, \pi/4, \pi/2$ and π.

From Figures 1-8, we notice plots of two types: straight line trajectories and elliptical trajectories.

Straight line trajectories occur either when x(t) = y(t) or x(t) = -y(t). This is largely dependent on the periodicity of the sin function and the inputted phase differences. In the cases where $A_x = 1, A_y = 1$, x(t) = y(t) either when the phase difference is 0 and therefore, the functions are the same. When the phase difference is π, you see that x(t) = -y(t) as sin(x) = -sin(x+π). In the cases where $A_x = 2, A_y = 1$, we see straight line but with a slope of 2 (or -2) instead of 1 as phase differences of 0 and π result in x(t) = 2y(t) or x(t) = -2y(t). These scenarios are seen in Figure 1, 4, 5, and 8.

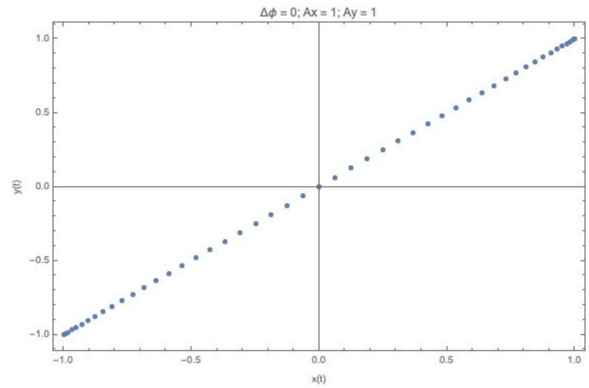

Figure 1: Ax=1, Ay=1, $\Delta\phi$=0

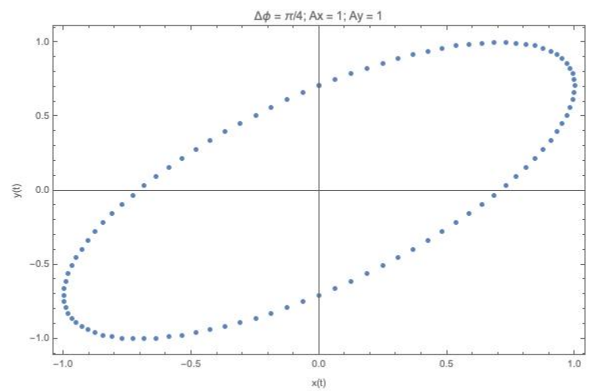

Figure 2: Ax=1, Ay=1, $\Delta\phi$=$\pi/4$

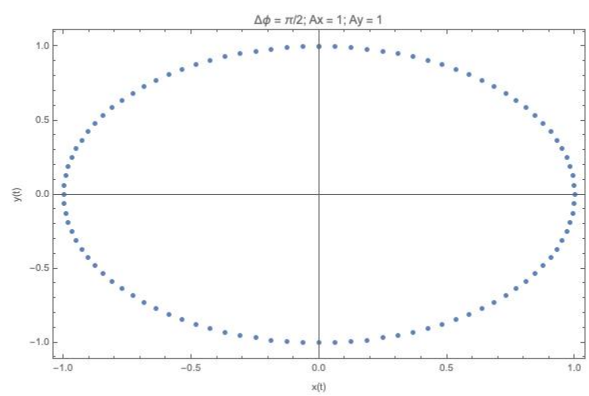

Figure 3: Ax=1, Ay=1, $\Delta\phi$=$\pi/2$

Now looking at the plots 2, 3, 6, and 7, we observe elliptical trajectories. In the cases where $A_x = 1, A_y = 1$ and $\Delta\phi = \pi/2 or \pi/4$, we see that the functions described by equation 2 and 3 are the same sine functions but shifted by a phase of a quarter-period or an eighth-period. With $\Delta\phi = \pi/2$, sin(x+π/2) = cos(x) and this one negative derivative offset results in elliptical trajectories seen in Figures 3 and 7. This ellipse is offset by an angle of $\pi/4$ when $\Delta\phi = \pi/4$ resulting in the trajectories seen in Figure 2 and 6.

Thus, we see that there are primarily two trajectories – straight line or elliptical trajectories – and they

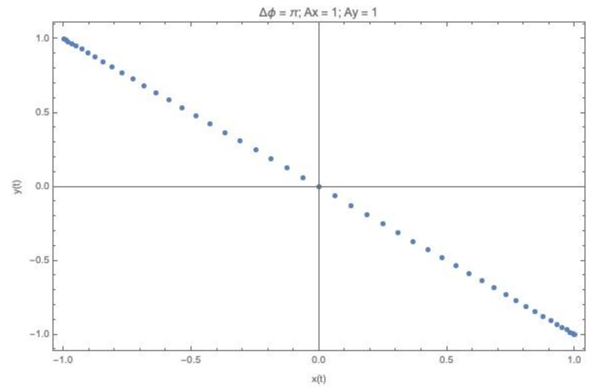

Figure 4: Ax=1, Ay=1, $\Delta\phi$=π

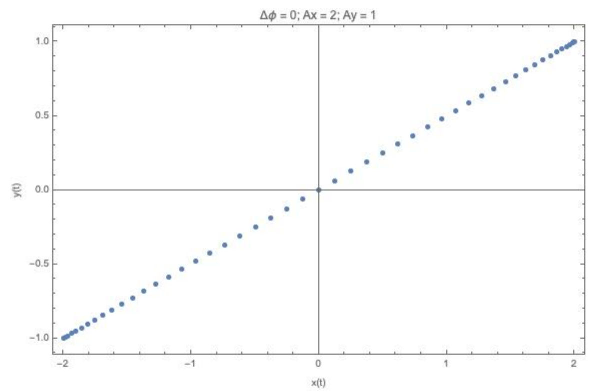

Figure 5: Ax=2, Ay=1, $\Delta\phi$=0

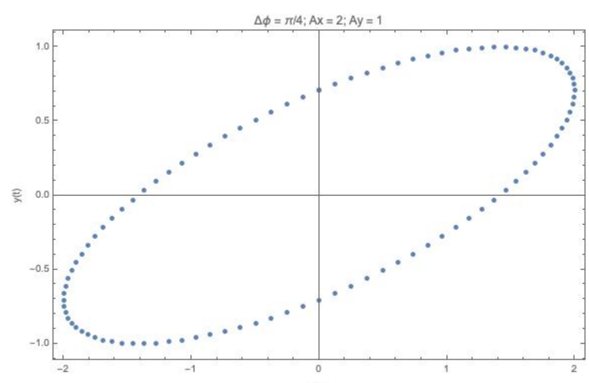

Figure 6: Ax=2, Ay=1, $\Delta\phi$=$\pi/4$

are dependent on the initial conditions inputted. When the phases are offset such that the sine function is the same (in magnitude), we see straight line trajectories and offsets other than that result in elliptical trajectories whose angle is determined by $\Delta\phi$. At the same, the width of these trajectories are determined by the amplitudes.

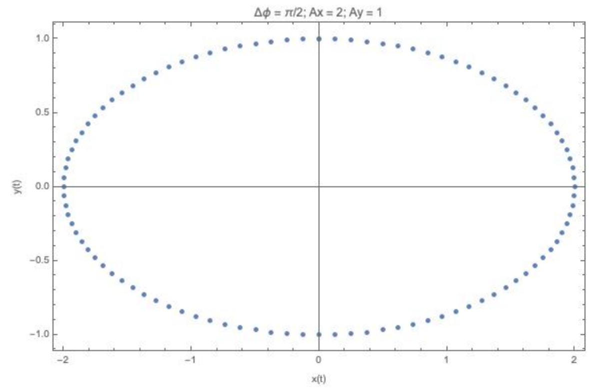

Figure 7: Ax=2, Ay=1, $\Delta\phi$=$\pi/2$

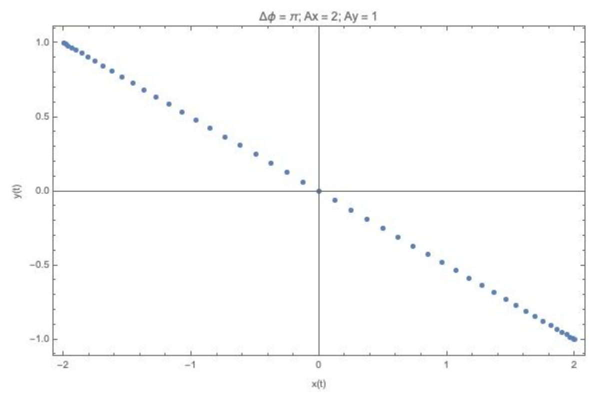

Figure 8: Ax=2, Ay=1, $\Delta\phi$=π

3 Conclusions

This project was my first attempt at coding in C and Mathematica and this proved to be a great way to begin not just learning to code in those languages, but to also produce professional quality work that is efficient.

Previously, I would have manually moved and exported data produced by code to make plots or would have used Python (which prevents users from understanding data types and their importance) to solve these problems. Making a README file with the initial code was a sign that I would not just be learning how to code, but also learning how to create code that contributes to the scientific community. As coding in C, Mathematica, and using GCC were unfamiliar to me, I found this to be a fairly new learning curve, but one that I surpassed well due to my experience with Python and Command Line language from research and other classes.

A large challenge was making my code clean and efficient. For example, I initially declared variables inside a for loop which slowed my code down. I chose to then declare these variables outside of the loop and then run the loop (that produced 100 coordinate points) and this produced results much quicker. I hope to better my ability to write clean and efficient code that is computationally efficient and easy to understand while most importantly, building my skills in C and Mathematica.

4 Lissajous Figures

I edited my code to allow users to input values of ω_t and ω_y and letting N (number of data points) equal to 10,000. With a phase difference of 0 and amplitudes and frequencies of 1, we produced our first beautiful Lissajous figure:

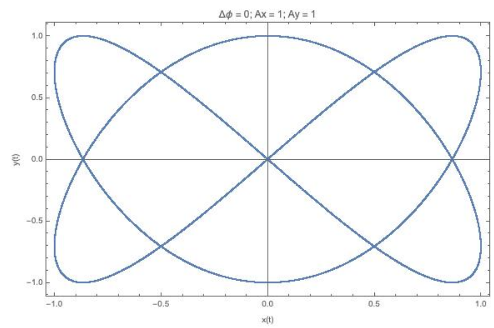

Figure 9: Lissajous figure with parameters Ax=1, Ay=1, $\Delta\phi$=0

A Lissajous figure with the ratio of frequencies, $\omega_x/\omega_y = \sqrt{3}$.

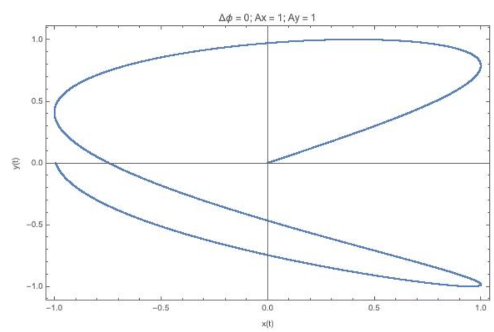

Figure 10: Lissajous figure with parameters Ax=1, Ay=1, $\Delta\phi$=0 and $\omega_{frac} = \sqrt{3}$

And lastly, a figure with ratio of frequencies, $\omega_x/\omega_y = \sqrt{3}$ with a phase difference of $\pi/2$.

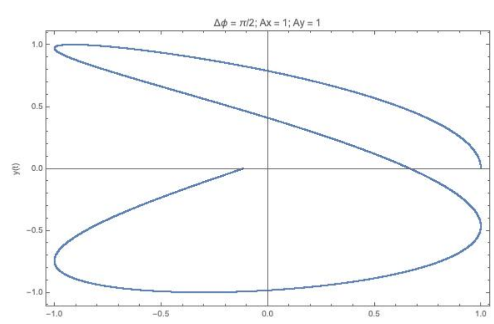

Figure 11: Lissajous figure with parameters Ax=1, Ay=1, $\Delta\phi$=$\pi/2$ and $\omega_{frac} = \sqrt{3}$