Hints, tips and possible solutions

- All of the programming required for the activities is done in Python, which should be easy to run on all popular operating systems. It does however require Python to be installed on your system, as well as the `numpy`, `matplotlib`, `wxPython` and `PyOpenGL` and `PyOpenGL-accelerate` packages – these can be installed using the `pip` tool. For more information on installing/using Python and Python packages on your system, have a look at Python For Beginners. If you are using the Anaconda/Conda environment, the above packages can easily be installed using `conda`.
- The code provided is fully commented for you to look through and explore. Feel free to modify and change things – in fact it is encouraged that you play around with the code. Try things out, see how they work, and form for yourself a practical understanding of using Python for mathematical/scientific modelling.

Activity 1 – Plotting sinusoids

Task 1

- Recall that $1^\circ = \frac{\pi}{180}$ rads
- To plot graphs using `matplotlib`, the data needs to be stored in a list-type structure. An easy way to do this is to use `numpy`:

```
# Import the numpy module and refer to it as np
import numpy as np

# Create an array, x
x = np.linspace(0, 4 * np.pi, 1000)

# Calculate y = sin(x) for all values of x;
# Store the results in an array, y
y = np.sin(x)
```

- In the above program, `x` is an array holding 1000 linearly spaced values, between the limits of 0 and 4π; `y` is an array which holds the sine of each of these values
- It is very easy to then plot the data using `matplotlib`:

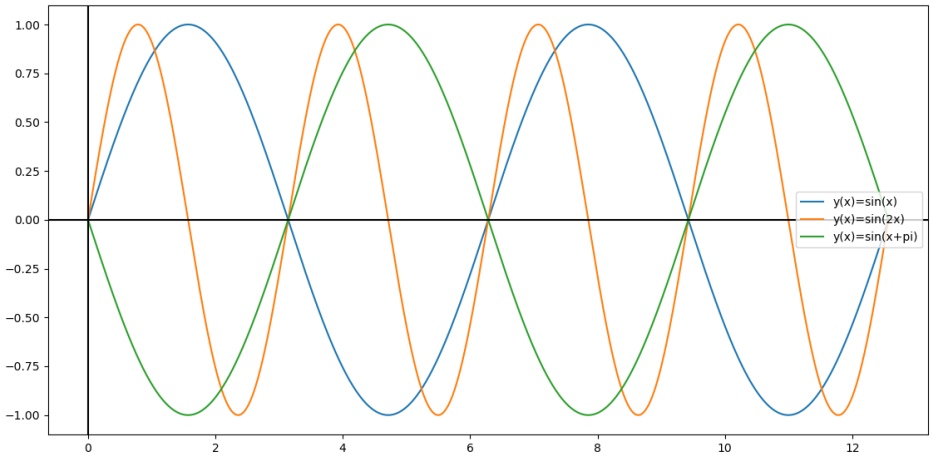
```
# Import the plotting module within matplotlib;
# Refer to it as plt
import matplotlib.pyplot as plt

# Plot the above x and y data;
# Display the graph
plt.plot(x, y)
plt.show()
```

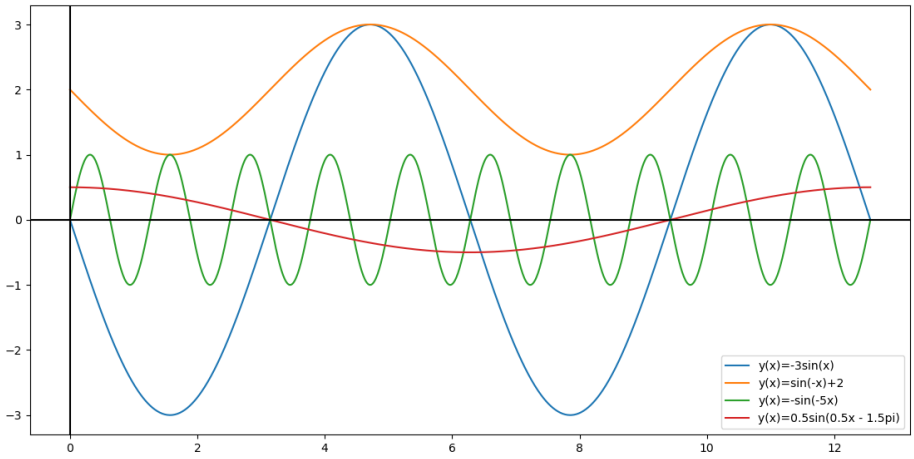
- More details for using `numpy` and `matplotlib` can be found in the Numpy tutorial and in the Matplotlib tutorials.

Task 2

1. y_1 should give you a sinusoid with twice as many waves as that in the previous task
2. y_2 should give you the same sinusoid as that in the previous task, this time shifted by half a wave



Task 3



Activity 2 – Parametric equations

Task 1

- For this task, first create an array, `t`, holding linearly spaced values between 0 and 5. Make sure to have a suitable number of values – 500 is more than enough
- Then calculate `x` and `y` points corresponding to these values

```
# Create array, t
t = np.linspace(0, 5, 500)

# Calculate x(t) and y(t)
x = 2 * t
y = 112.5 + 2 * t - 4.9 * t**2

# Plot x(t) against t, and y(t) against t;
# Display a legend to label each plot
plt.plot(t, x, label="x(t)")
plt.plot(t, y, label="y(t)")
plt.legend()
plt.show()

# Plot x(t) against y(t)
plt.plot(x, y)
plt.show()
```

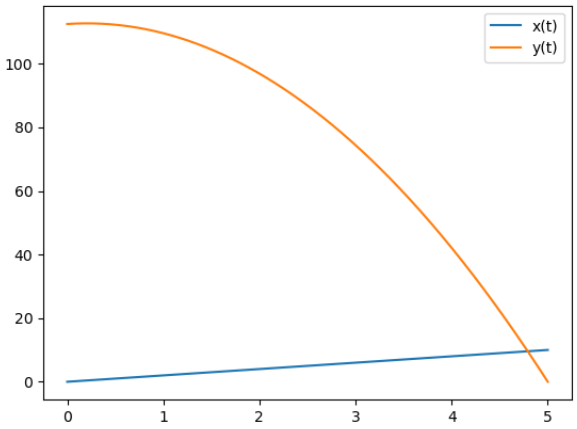


Figure 1: $x(t)$ and $y(t)$ plotted against t

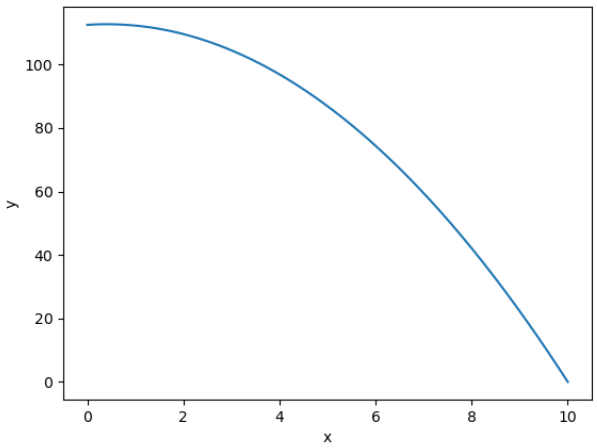


Figure 2: x plotted against y

Task 2

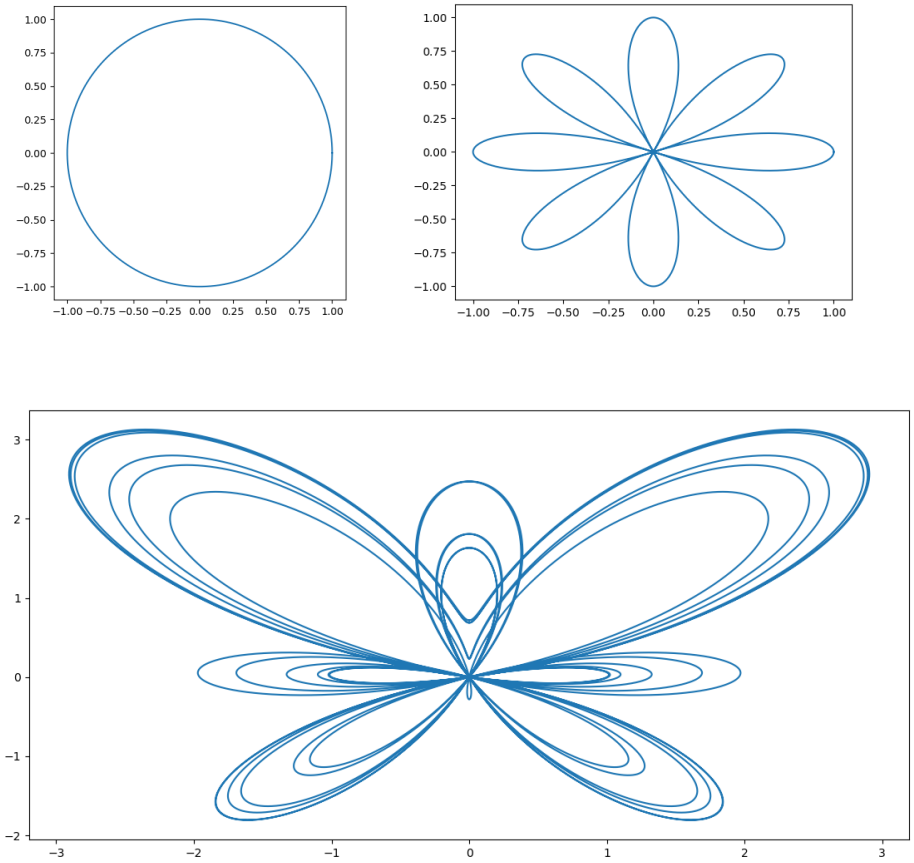


Figure 3: Top left: A simple circle; Top right: A flower pattern known as a rose or rhodonea; Bottom: A butterfly curve

Activity 3 – Investigating Lissajous figures

Task 1

Note that `lissajous.py` *requires both* `wxPython` *and* `PyOpenGL` *to run. The code has been fully commented so if you are interested in GUI programming, feel free to read through the code to see how it works.*

- For the investigation, it is easiest to do it with the animation off
- Try to be systematic with your investigation – e.g. vary only one thing at a time
- Questions to guide your investigation:
  - Keeping the ω_x angular frequency constant at 1 rad/s and varying ω_y (in integer increments), how many loops are there in the resulting Lissajous figures?
  - Now keeping the ω_y angular frequency constant at 1 rad/s and varying ω_x (in integer increments), how many loops are there in the resulting figures? How are these figures different to the ones when ω_x was constant and ω_y varied?
  - What do you notice about the figures when both angular frequencies, ω_x and ω_y, are odd integers? And when one is odd, and the other even?
  - What do you notice about the figures for a constant ratio ω_x/ω_y? e.g.

$$\omega_{x1} = 1,\ \omega_{y1} = 2,$$

$$\omega_{x2} = 10,\ \omega_{y2} = 20$$

  - Set the ratio of x/y to 2. What appears to happen to the figure when you vary the phase shift setting?
  - What do you notice about the figures when using non-integer values for ω_x and/or ω_y? (This is further explored in the next task)

Task 2

- When decimal values are used for ω_x and/or ω_y:
  - You can see from the plots of x and y against t that at a certain t, x and y both complete a wave at the same time
  - You can think of this as being a pseudo lowest common multiple. After this point, the figure will repeat and go over itself
  - From the value of t at which this happens, the number of 2π cycles required for a complete figure to be drawn can be determined
  - Multiplying the decimal values of ω_x and ω_y by the number of cycles required to draw the full figure should give you integers on the top and bottom of the ratio ω_x/ω_y, i.e. the Lissajous figure is only closed if the ratio is rational

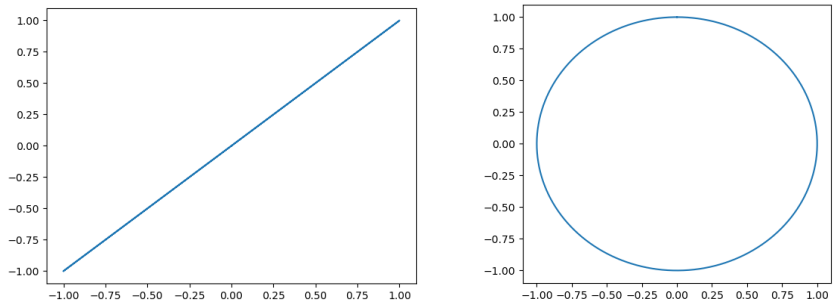


Figure 4: Questions 1 and 2; effects of adding a phase shift

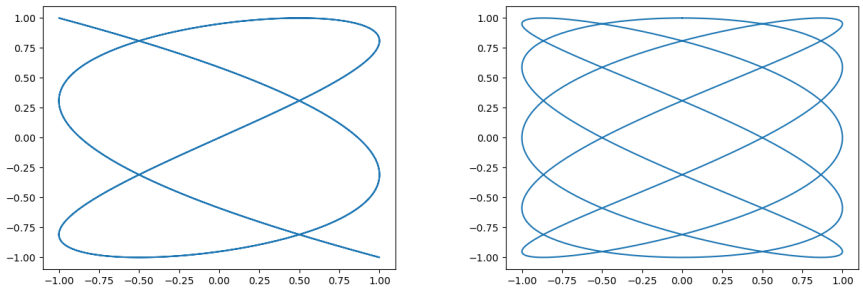


Figure 5: Questions 3 and 4; effects of adding a phase shift

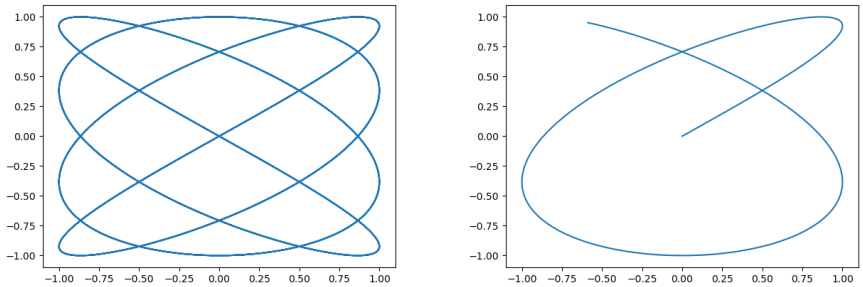


Figure 6: Questions 5 and 6; effects of an irrational ratio

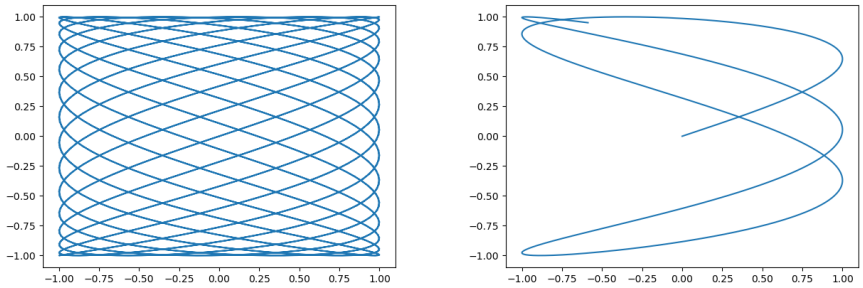


Figure 7: Questions 7 and 8; effects of an irrational ratio

Activity 4 – Laser surface heat treatment

Task 1

- For laser 1, at $x = 0$, the power is 1 W, so for 5 seconds, $1 \times 5 = 5$ joules of energy is transferred.

  At $x = 3$ the power is zero. Thus no energy is transferred.

- For laser 2, at $x = 0$, the power is 1 W, so for 5 seconds, $1 \times 5 = 5$ joules of energy is transferred.

  At $x = 3$ the power is zero. Thus no energy is transferred.

Task 2

- Laser 1 first hits the object at $x = 0$ at $t = 3$ seconds, and leaves $x = 0$ only at $t = 5$ seconds. Since this laser has a uniform power distribution, the total energy transferred is simply $1 \times 2 = 2$ joules.

- Laser 2 first hits the object at $x = 0$ at $t = 3$ seconds, and leaves $x = 0$ at $t = 5$ seconds. The power distribution is however not uniform. To calculate the energy transferred at this point, we can use the fact that the laser moves with constant velocity. This means that each part of the laser spot will be incident on $x = 0$ for the same amount of time. Thus simply taking the area under the distribution will give the total amount of energy transferred (see the next question as to why this works). The area is 1, i.e. 1 joule of energy is transferred.

Task 3

- Power, P is the rate of change of energy, E. If power is a function of time, i.e. $p(t)$, then this relationship can be written as $p(t) = \frac{\mathrm{d}E}{\mathrm{d}t}$. Thus to find E, we can integrate p:

$$E = \int p(t)\mathrm{d}t$$

- The power distribution given in the question however is a function of position, x, and the question asks for the energy transferred at the single point $x = 2$

- From the question, we know the position of the laser at any time, t, i.e. $x = t^2$

- Thus at a certain position along the object, x_0, the power is simply a transformation of the original distribution, i.e. $p(x_0 - t^2)$

- Hence integrating this function with respect to t – the x_0 is simply a constant – gives the total energy transferred by the laser to the object at the point x_0

- For this question the laser is incident on $x = 2$ when $-1 \leq x_0 - t^2 \leq 1$, which is when t is in the interval $[1, \sqrt{3}]$. In this time, 1 W is incident on the object, so the total energy transferred is $(\sqrt{3} - 1)$ joules

Task 4

- To plot a 2D Gaussian distribution, you will need to import an extra module from `Matplotlib` as shown in `gaussPlot.py` and below:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # 3D plotting tools

# Define the variances in the x and y directions
sigma2X = 1
sigma2Y = 1

# Create x and y arrays
x = np.linspace(-4, 4, 101)
y = np.linspace(-4, 4, 101)

# Create a meshgrid
# This enables us to calculate the Gaussian function over a grid of x,y
    values
X, Y = np.meshgrid(x, y)

# Calcuate the Gaussian function
Z = np.exp(-0.5 * ((X**2 / sigma2X) + (Y**2 / sigma2Y)))

# Set up the 3D axis to plot
fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")

# Plot the Gaussian and display
ax.plot_surface(X, Y, Z)
plt.show()
```

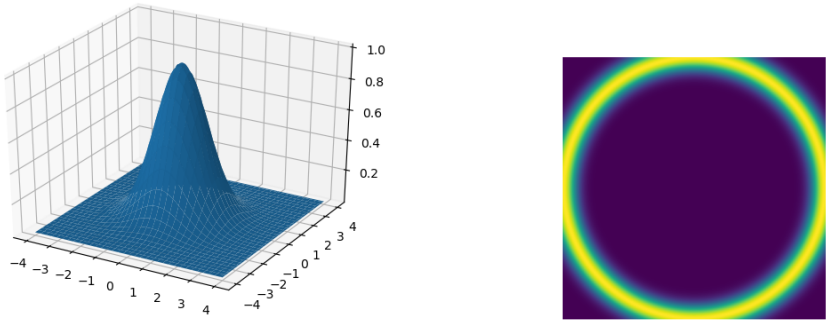


Figure 8: Left: A 2D Gaussian distribution; Right; A heatmap for a Lissajous laser heat scanner

- The output shows a heatmap with the hottest part being the Lissajous figure itself
- When σ_x and σ_y change, the width of the lines in the Lissajous figure change as well
- Many possible strategies (using the model presented) can be used. One strategy to get uniform hardening is to use a very large ratio $\frac{\omega_x}{\omega_y}$, but only place the object in the part of the Lissajous figure where the lines are very close to each other. Another strategy could be to increase σ_x and σ_y and use a smaller ratio