PHOT 110: Introduction to programming

Lecture 08: supporting materials

Michaël Barbier, Spring semester (2023-2024)

Exercises on creating plots with Matplotlib

We will summarize the method to generate simple line plots using Matplotlib, and elaborate on the styling of the plot. I put two Matplotlib cheat-sheets/hand-outs on teams which can also be found on the Matplotlib website (I put the hand-outs for beginner and intermediate users on teams).

At the beginning of the script you should first import the required libraries (numpy and matplotlib):

```
# Import numpy and matplotlib
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use("WebAgg")
```

Exercise 1: Refraction at a glass interface

Plot the refracted angle θ_2 (using Snell's law) of a ray of light incident at a glass medium, as function of the incoming angle: use the incoming angle $\theta_1 \in [-90, 90]$ degrees medium 1 has $n_1 = 1$, medium 2 has $n_2 = 1.55$.

Snell's law is given by:

$$n_1 \sin(\theta_1) = n_2 \sin(\theta_2)$$

If refraction indices n_1 and n_2 are known, the refracted angle can be calculated as follows:

$$\theta_2 = \arcsin\left(\frac{n_2}{n_1} \sin(\theta_2)\right)$$

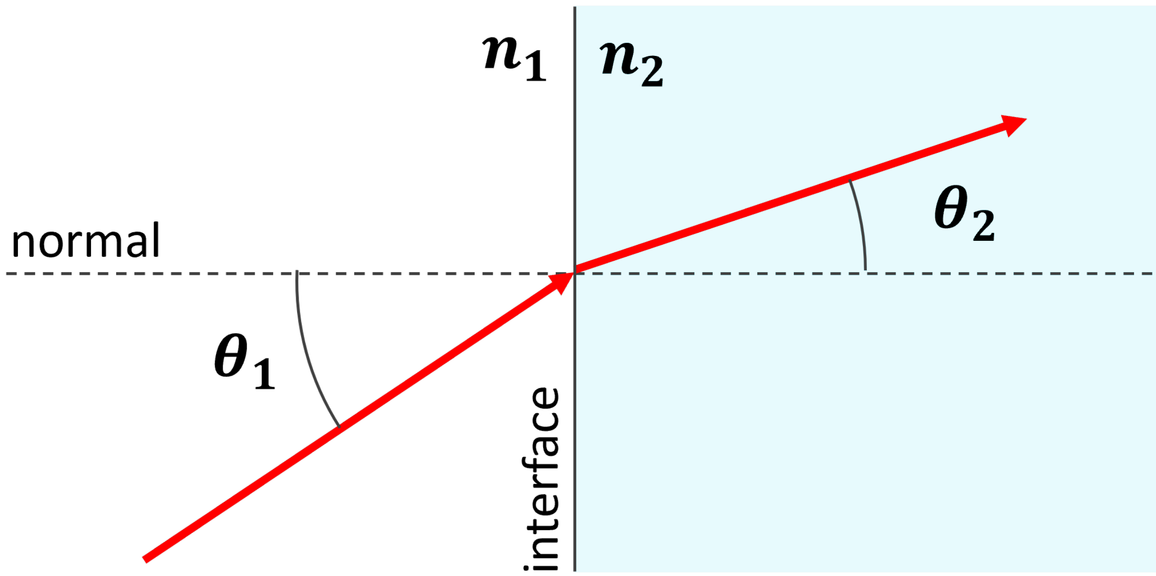
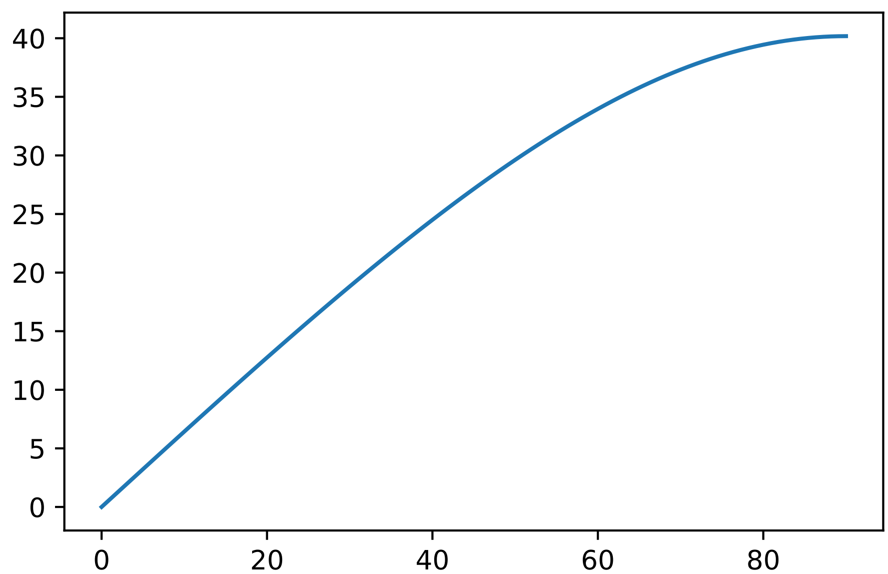


Figure 1: Snell's law

Hint: Use Numpy arrays for the angles (np.linspace()), and Numpy functions: np.sin(), np.arcsin() (convert to radians by multiplying by a factor pi/180) or use the functions np.rad2deg and np.deg2rad to convert between the two.

The output of your code should be similar to following plot:



Exercise 2: Adapt previous plot to include axis labels, etc.

- Add labels to your axes using the `set_xlabel` and `set_ylabel` methods. Put $-symbols around mathematical/Greek letters and double-escaped Greek letters such as theta (the escape symbol is "\"). For example:

```
ax.set_xlabel("$\\theta_1$ (in degrees)")
```

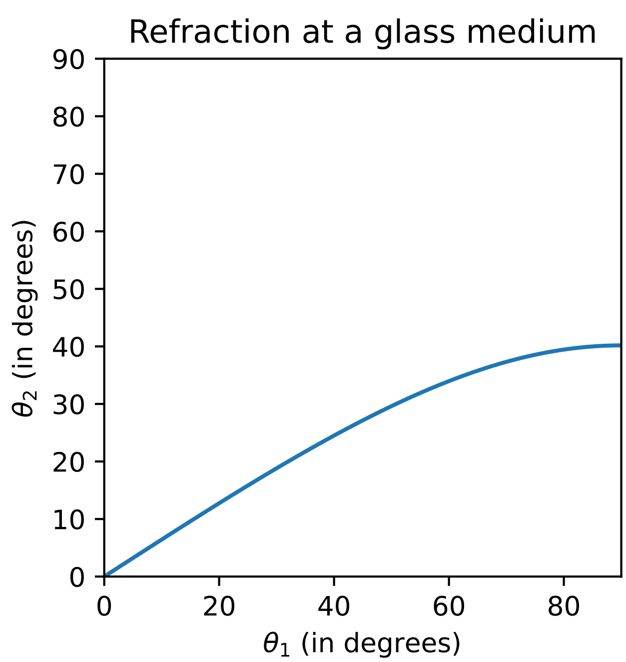
- Add a title to your plot (e.g. "Refraction angles") by using the method: `ax.set_title()`
- Set the intervals to be plotted between 0 and 90 degrees, and set the aspect ratio of the axes equal to one:

```
    ax.set_xlim([0, 90])
    ax.set_ylim([0, 90])
    ax.set_aspect('equal', 'box')
```

- Save the figure using fig.savefig() as a png figure, for example:

```
    fig.savefig("./ex2_figure.png")
```

The output of exercise two should give you a similar plot to the one below:



Exercise 3: Extend previous script to show multiple incident angles

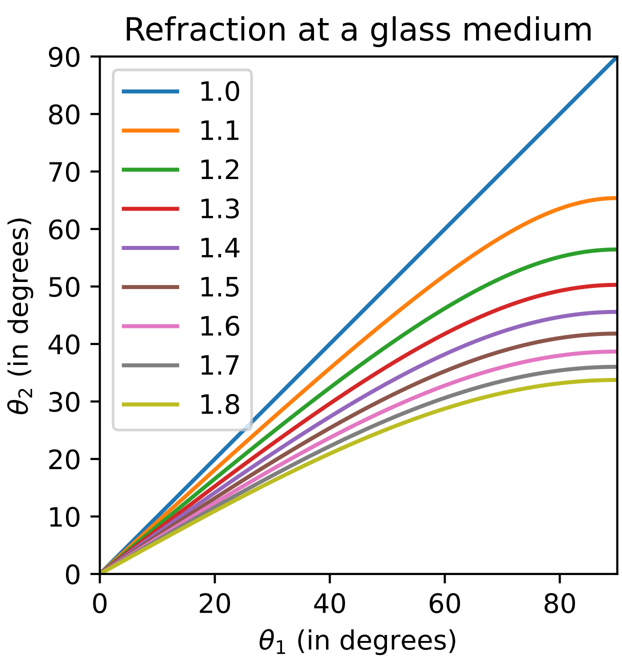
Plot the previous plot but now with multiple values for $n_2 \in [1, 1.8]$

- use the arange function of Numpy to create values for n_2 within the interval, for example each value separated by $\delta n = 0.1$

```
n2s = np.arange(1, 1.85, 0.1)
```

- Plot in the same axis when looping over the values (but initialize the `fig, ax` only once before the loop)
- add a legend using `plt.legend(n2s)`
- to remove the extra decimals due to rounding errors, use `plt.legend(np.round(n2s, decimals=2))` to round the numbers before showing the legend.

The output of exercise 3 should give the below plot:



Exercise 4: Plot ellipses with various parameters

Plot ellipses as parametric plots with parameter $t \in [0, 2 * \pi]$ and where the (x, y)-curve is given by:

$$x = a \cos(t)$$
$$y = b \sin(t)$$

Plot multiple ellipses with different values of a and b:

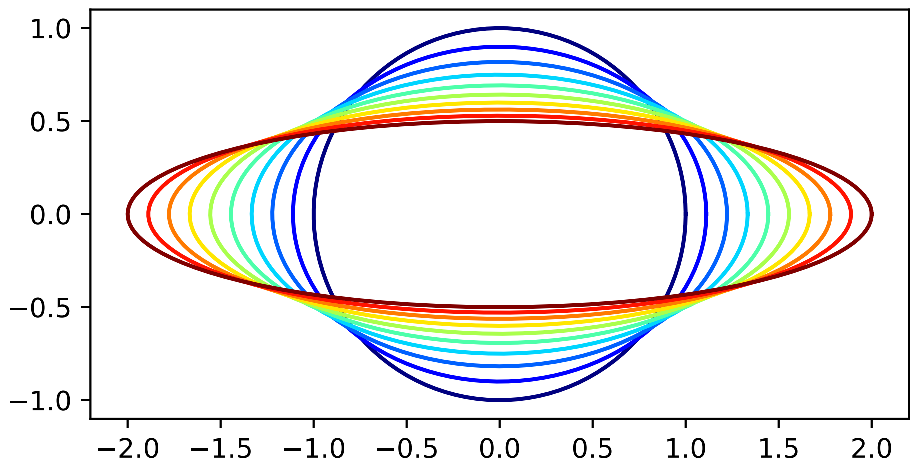
- take parameter $a \in [1, 2]$ and choose a small (e.g. 8) curves using `np.linspace()`, afterwards calculate corresponding value for parameter b from a by using $b = 1/a$
- give every ellipse a different color and choose the color palette Matplotlib uses by using e.g.:

```
colors = plt.cm.jet(np.linspace(0, 1, n_curves))

# Then afterwards define the line color:
ax.plot(xs, ys, color=colors[i])
```

- use: `ax.set_aspect('equal', 'box')` to set the aspect ratio of the x and y axis equal

The output of the script should give the following plot:



Exercise 5: Plot Poinsot's spiral (2nd form),

Poinsot's spiral is defined in polar coordinates as:

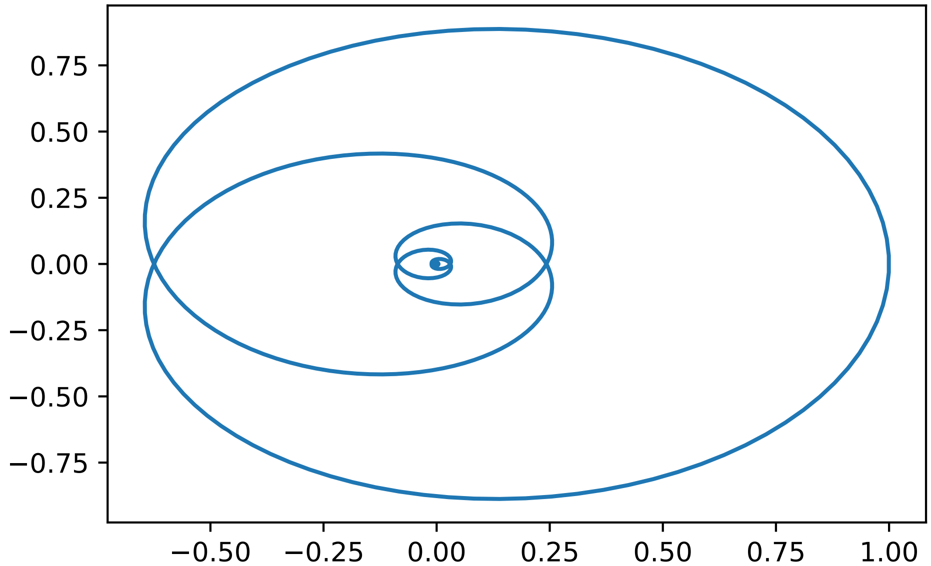
$$r = \frac{1}{\cosh(n\theta)}$$

plot with θ in interval $[-10 * pi, 10 * pi]$ and convert from polar coordinates to (x,y)-coordinates by

$$x = r\cos(t)$$
$$y = r\sin(t)$$

- Take the parameter n for example $n = 1/3$ to have a typical curve

The output plot should look as below:



Random numbers in Numpy

Exercise 1: Plot a Gaussian distribution

Generate 10000 random numbers with a normal distribution and plot the histogram - First initialize the random number generator

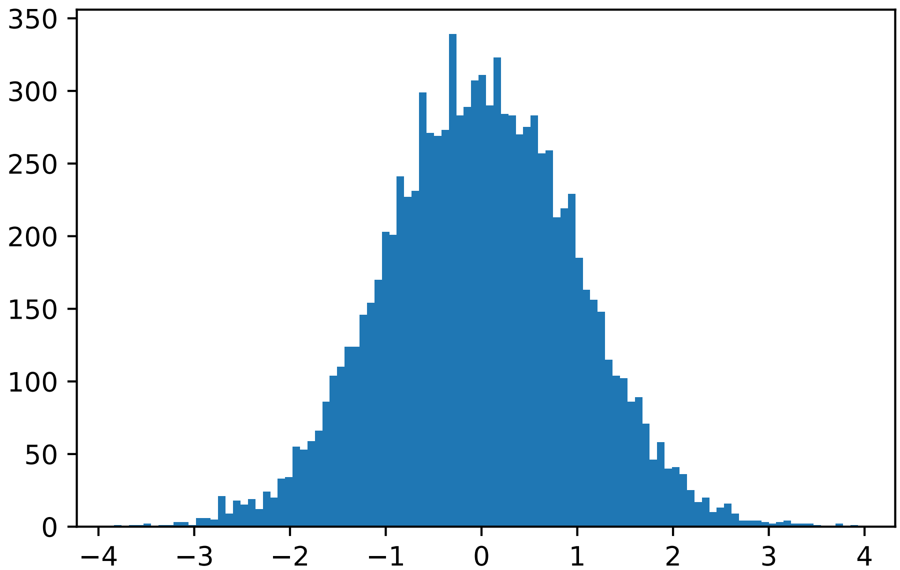
```
rng = np.random.default_rng(1)
```

- use the random number generator we created to generate 10000 random numbers from the standard normal distribution using:

```
xs = rng.standard_normal(size=10000)
```

- plot a histogram using `ax.hist(x_uniform, 100)`

The resulting plot should be:



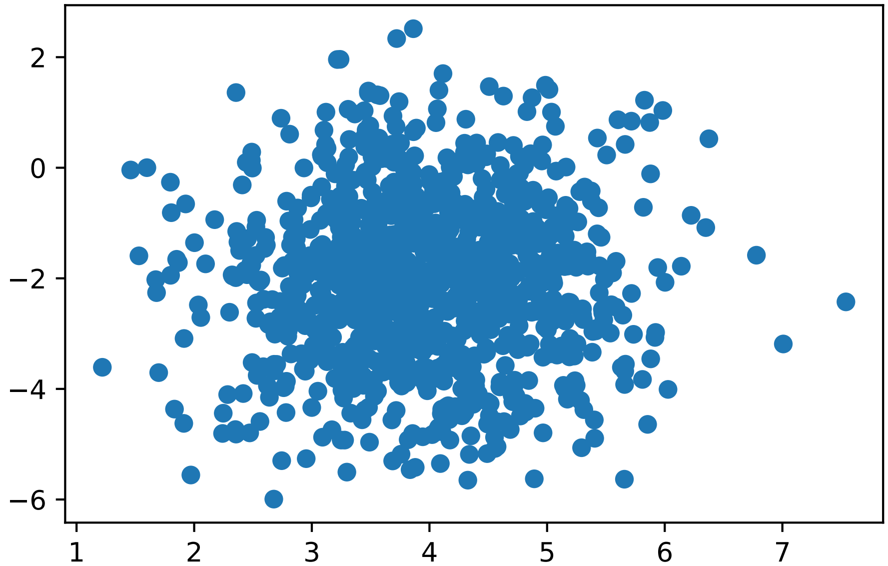
Exercise 2: Make a scatter plot of a Gaussian distribution in 2D

- use the random number generator we created to generate 1000 random numbers for both x and y coordinates,
- Use the normal distribution
- fill in `loc(=mean)`, and `scale(=sigma)` as extra parameters:

```
xs = rng.normal(loc=4, scale=0.9, size=1000)
ys = rng.normal(loc=-2, scale=1.5, size=1000)
```

- plot them in a scatter plot using: `ax.scatter(xs, ys)`

The resulting output should look as below:



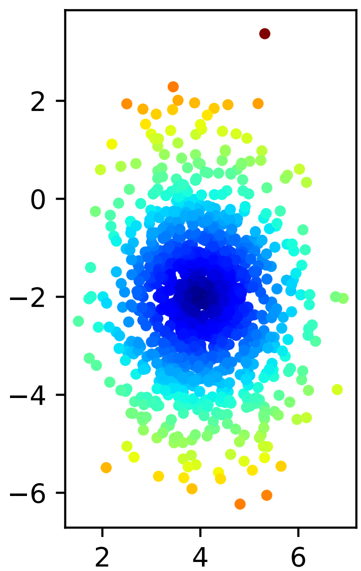
Exercise 3: Adapt the scatter plot of exercise 2 by specifying the size and color of the dots

```
ax.scatter(xs, ys, sz, colors, cmap=matplotlib.colormaps["jet"])
```

where both `sz` and `colors` can be arrays of the same size as the coordinates or scalars

- for the size use for example 10
- for the colors create an array containing the distance of the coordinates to the center of the distribution
- the cmap argument allows us to set the colormap used

The resulting scatter plot should look as below:


2
0
−2
−4
−6
2
4
6