1 Simple example

A book is described by its title, author and year of publication (to keep it simple). Write a class `Book` that store those information. Write the `__repr__` and `__str__` methods.

A bibliography is mainly a list of book. Write a class `Bibliography` that store the list a book (the class has an attribute which is a list).

At the end we want to be able to use the code as follow

```
book1 = Book("A very nice book", "F. Dupont", 2014)
book2 = Book("A very smart book", "A. Einstein", 1923)
book3 = Book("A very stupid comics", "D. Duck", 1937)

bibliography = Bibliography([book1, book2, book3])
```

Now that the data are store in object, there are many methods you can think of :

- Write a method `to_latex` that returns a string that can be inserted in LaTeX
- Write a method `filter_by_year` to create a new bibliography, keeping books only from a specific year.

2 Complex impedance of a bipolar circuit

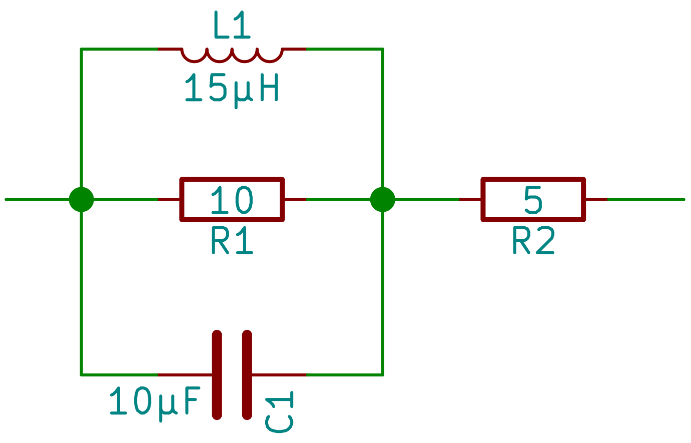

Fig. 1: Bipolar circuit

The purpose of this problem is to calculate the complex impedance of a bipolar circuit made with resistor, capacitor and inductor.

A bipolar circuit is either one of those devices or a parallel or serial combination of bipolar circuits.

The class structure will be the following

```
class BipolarCircuit(object):
    pass

class Combination(BipolarCircuit):
    pass

class Serial(Combination):
```

(continues on next page)

(continued from previous page)

```
    pass

# idem for parallel

class Device(BipolarCircuit):
    pass

class Resistor(Device):
    pass

# idem for capacitor and inductor
```

Write those classes so that we can use it as follow

```
my_circuit = Serial(Parallel(Resistor(10),Capacitor(1E-5),Inductor(15E-6)),
                    Resistor(5))
#or
my_circuit = (Resistor(10)|Capacitor(1E-5)|Inductor(15E-6))+Resistor(5)

Tfreq = logspace(3, 7)
plt.semilogx(Tfreq, np.abs(my_circuit.impedance(Tfreq)))
```

where the `|` (`__or__`) represents the parallel combination and the + (`__add__`) the serial combination.

You can also implement the method

```
my_circuit.plot_impedance(f_min=1000, f_max=10000000, use_log_scale=True)
```

The wikipedia page https://en.wikipedia.org/wiki/Electrical_impedance contains all the formulas you need!

To go further :

- make a package from this project. Add modules to perform unit test using the `unittest` package.
- make the classes compatible with the symbolic calculation using sympy. Use unit test to check that it is still compatible with numbers

3 Ray Tracing

All code pieces below are in a jupyter notebook.

3.1 Goal

A ray tracing programm is used to calculate the propagation of a light beam through an optical system. Such calculation can be done by commercial programm (such as Zemax or OSLO). The goal of this lecture is to write our own.

3.2 Optical Interface

We will assume that the interface between two media of index n_1 and n_2 is spherical.

Such an interface is caracterized by the position z_0 of its intersection with the z axis and by its curvature radius R. The convention is that the center of the sphere is at $z_0 + R$. An optional diameter describe also the diameter.

We will create a `SphericalInterface` class using :

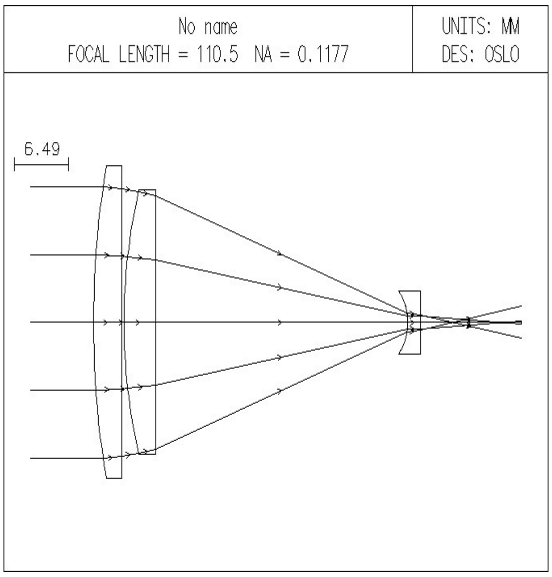

Fig. 2: Ray tracing obtained using OSLO. One ca see the spherical aberations.

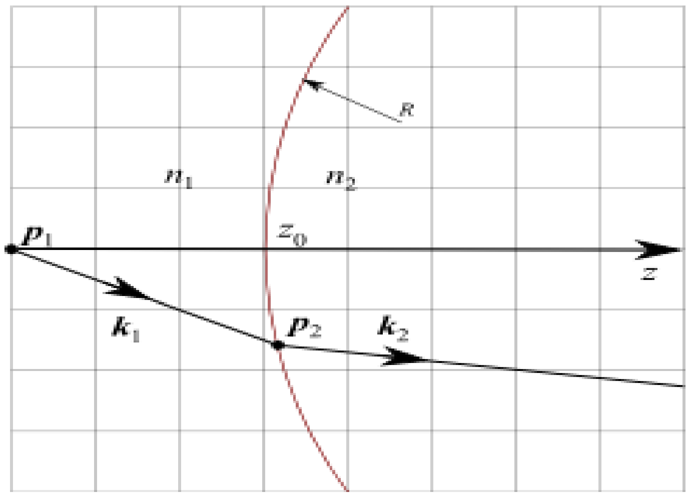

Fig. 3: Refraction by a spherical interface

```python
class SphericalInterface(object):
    diameter = 25.4
    def __init__(self, z0, R, n_1, n_2, diameter=None):
        self.z0 = z0
        self.R = R
        if diameter is not None:
            self.diameter = diameter
        self.n_1 = n_1
        self.n_2 = n_2
```

- Explain what is done with diameter in the `__init__` methode. Why not simply have `diametre=25.4` as an optional argument of `__init__` ?
- We need the position `z_center` of the center of the sphere. Calculate and add this attribute in the `__init__` method.
- Write the `__repr__` method that display the main paramters of the interface.
- Write the `plot` method that plot the optical interface in a graphic. The horizontal axis of the graph will be the direction of propagation (z axis) and the vertical is the y-axis. The equation is given by :

  $$z = z_0 + R \pm \sqrt{R^2 - y^2}$$

  where there is '-' is R>0 and a '+' if R<0.

3.3 Snell - Descartes law

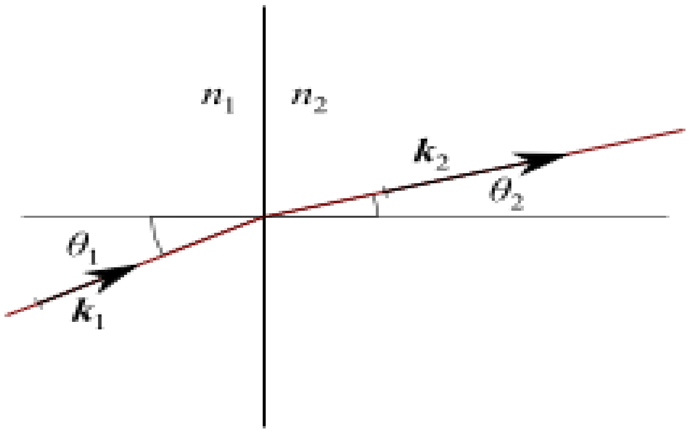

Fig. 4: Snell-Descartes law

You probably know the Snell - Descartes law as $n_1 \sin(\theta_1) = n_2 \sin(\theta_2)$. It will be more useful to use the wave vector $\mathbf{k}_1$ and $\mathbf{k}_2$. They are parallel to the propagation of the beam and their length is proportional to the refractive n of the medium. The Snell-Descartes law states then that the component of $\mathbf{k}$ parallel to the interface is conserved.

3.4 Light ray

A light ray is caracterized by a point $\mathbf{p}$ and a vector $\mathbf{k}$ such that the norm of $\mathbf{k}$ is the refactive index. It will be represented using the class

```python
class Ray():
    def __init__(self, p0, k, n=None):
        self.p0 = p0
```

(continues on next page)

(continued from previous page)

```
        self.k = k
        if n is not None:
            self.normalize(n)
    def normalize(self, n):
        """Normalize k such that ||k||=n"""
        pass
    def __repr__(self):
        return "Ray(p0={p0}, k={k}".format(p0=self.p0, k=self.k)
```

- Write the method `normalize`.

3.5 Refraction by a spherical interface

A ray $(\mathbf{p}_1, \mathbf{k}_1)$ cross the interface at the position $\mathbf{p}_2$. The wave vector of the new ray is $\mathbf{k}_2$.

We need two methods : the first one calculates the intersection and the second one the refracted ray.

- Vectors will be represented using numpy array
- The intersection with the sphere is obtained by solving the parametric equation $||\mathbf{p}(t) - \mathbf{c}||^2 = R^2$ where $\mathbf{c}$ is the center of the sphere and $\mathbf{p}(t) = \mathbf{p_1} + t\mathbf{k}$. The choice between the two solutions of this second order equation depends upon the sign of R.
- Calculate the vector normal to the interface at $\mathbf{p}_2$. Calculate the parallel component : $\mathbf{k}_{||} = \mathbf{k} - (\mathbf{k}\cdot\mathbf{n})\mathbf{n}$ and finally $\mathbf{k_2}$ with $\mathbf{k_2} = \mathbf{k}_{||} + \alpha\mathbf{n}$ where α is choosen such that $||\mathbf{k}_{||}||^2 + \alpha^2 = n_2^2$. Take care of the sign of α.

Calculation can be done with the following parameters :

```
p1 = np.array([0,0,-3])
z0 = 0
R = 6
n1 = 1
n2 = 1.5
k1_x = np.array([0, .5, sqrt(.75)])
```

The solution to this rather technical question is in the notebook

3.6 Beam

A beam is a list of rays.

```
class Beam(list):
    #There is no __init__ method.
    def plot(self):
        pass
```

- Write the `plot` method by joining the starting point of the rays.

3.7 Optical system

An optical system is a list of interface.

```
class OpticalSystem(list):
    def calculate_beam(self,r0):
```

(continues on next page)

(continued from previous page)

```
        beam = Beam()
        beam.append(r0)
        for interface in self:
            beam.append(interface.refract(beam[-1]))
        return beam
    def plot(self):
        pass
```

- Write the `plot` method (two lines).

3.8 Exemple

Below is a first example of our program (lens from Thorlabs):

```
wave_length = 780E-6 # mm

n_LAH64 = 1.77694
n_SF11 = 1.76583
n_air = 1.0002992


S1 = SphericalInterface(0,-4.7, n_air, n_SF11, diameter=3)

S2 = SphericalInterface(1.5,1E10, n_SF11, n_air, diameter=3)

LC2969 = OpticalSystem()
LC2969.append(S1)
LC2969.append(S2)

screen = SphericalInterface(100, 1e10, 30, n_air, n_air)

system = OpticalSystem()
system.extend(LC2969)
system.append(screen)

r0 = Ray(p0=np.array([0,1,-5]), k=np.array([0,0,1]), n=n_air)
beam = system.calculate_beam(r0)

LC2969.plot()
beam.plot()
```

3.9 To go further

- The `append` and `extend` method of `OpticalSystem` heritates from `list`. Rewrite those methods so that you raise an error if the argument of `append` is not an `SphericalInterface` and the argument of `extend` is not an `OpticalSystem`. Of course, you need to call the parent method.
- Write the __add__ of `SphericalInterface` and `OpticalSystem`, such that the following works correctly :

```
S1 = SphericalInterface(0,-4.7, n_air, n_SF11, diameter=3)
S2 = SphericalInterface(1.5,1E10, n_SF11, n_air, diameter=3)
LC2969 = S1 + S2
```

(continues on next page)

(continued from previous page)

```
screen = SphericalInterface(100, 1e10, 30, n_air, n_air)

system = LC2969 + screen
```

- Write the `reflected` method (reflection along the z=0 plane). Take the lens above and check that spherical aberation depends stongly upon the sens of the lens.
- Write the method `translated(self, d)` for SphericalInterface and OpticalSystem. It returns a *new* object translated by the distance d along the z axis.
- Actually, on can consider the `SphericalInterface` as a particular `Interface`. Write a parent class `Interface`. Create now a `SphericalInterface` class that heritates from `Interface`. Create then a `PlannarInterface` class and a `Screen` class (that heritates from `PlannarInterface`).
- Create a `PlanoConvexLens` class (heritates from `OpticalSystem`). Parameters are thickness, curature radius, diameter and refractive index.
- Look on the Throlabs web page to understand how aspherics are described. Implement an `AsphericInterface` class.
- There is a confusion in the programm between the glass we use and the value of its refractive index. Create a class that describe a glass, with an attribute that give the refractive index. Actually, the refractive index should be a function that depends upon the wavelength. Modify the `Ray` class to add a wavelength attributes that propagates correctly in the `Beam` class, and implement chromatic dispersion in the programm. Check that the LC1969 is an achromatic doublet.