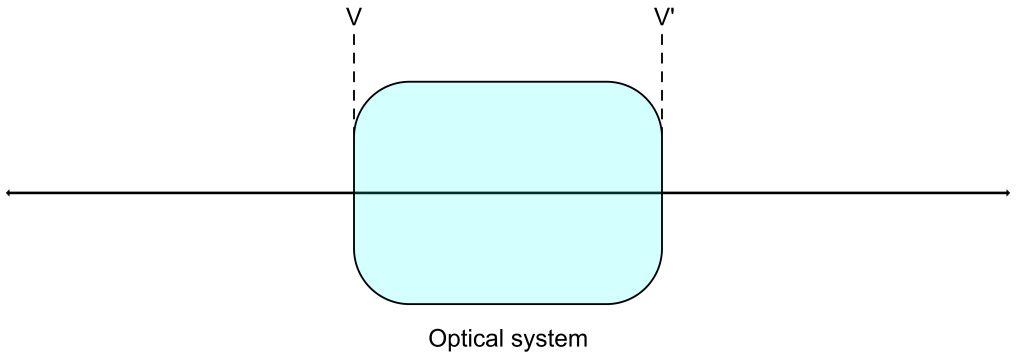
Gaussian Reduction Project

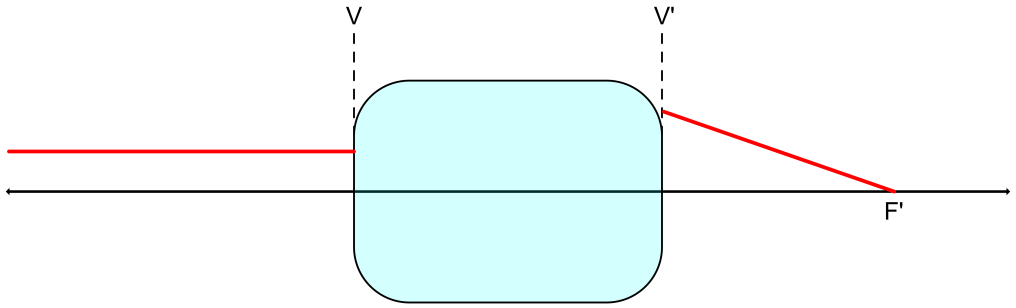
OSC Python Workshops, Fall 2023

Background

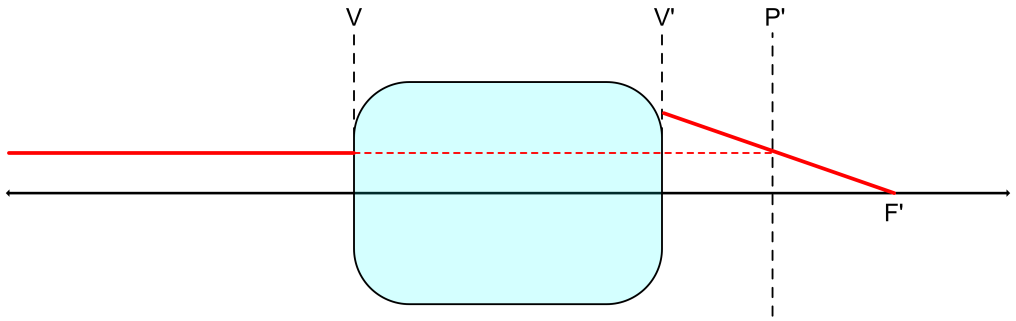
Gaussian reduction is a process by which a complex lens system can be reduced into a more basic form by identifying the location of surfaces called **principle planes**. Consider a black box optical system:



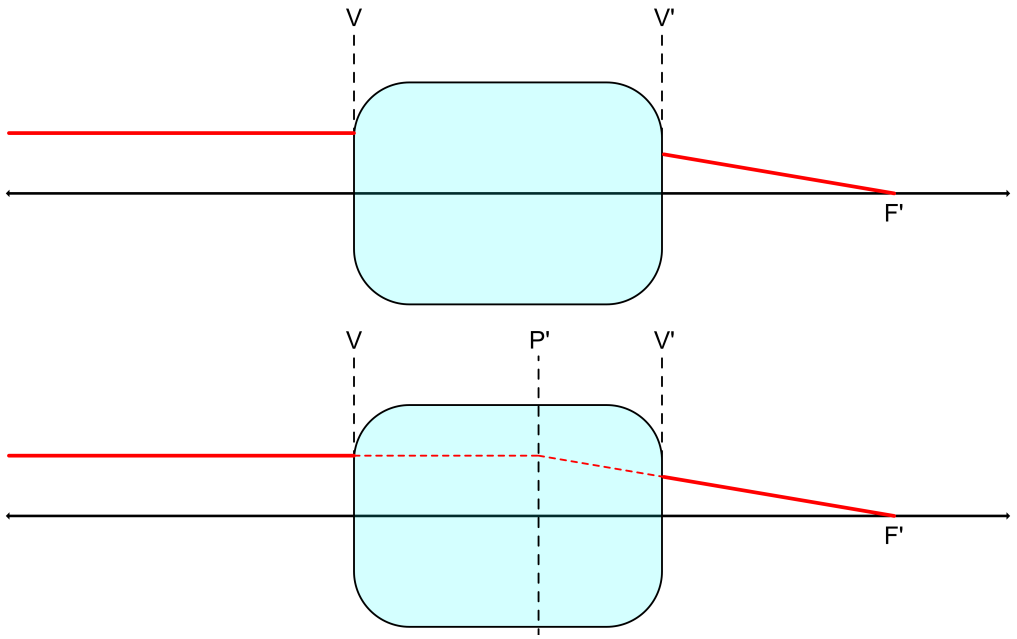
V and V' are called the front and rear vertex, which denote the location of surfaces farthest to the left and to the right along the optical axis. If we send in a ray from infinity on the left, it will refract through the optical system, then exit on the other side. The location where this ray crosses the optical axis is called the rear focal point (F'), as a well designed optical system will have all rays from infinity converging to the same point on the optical axis.



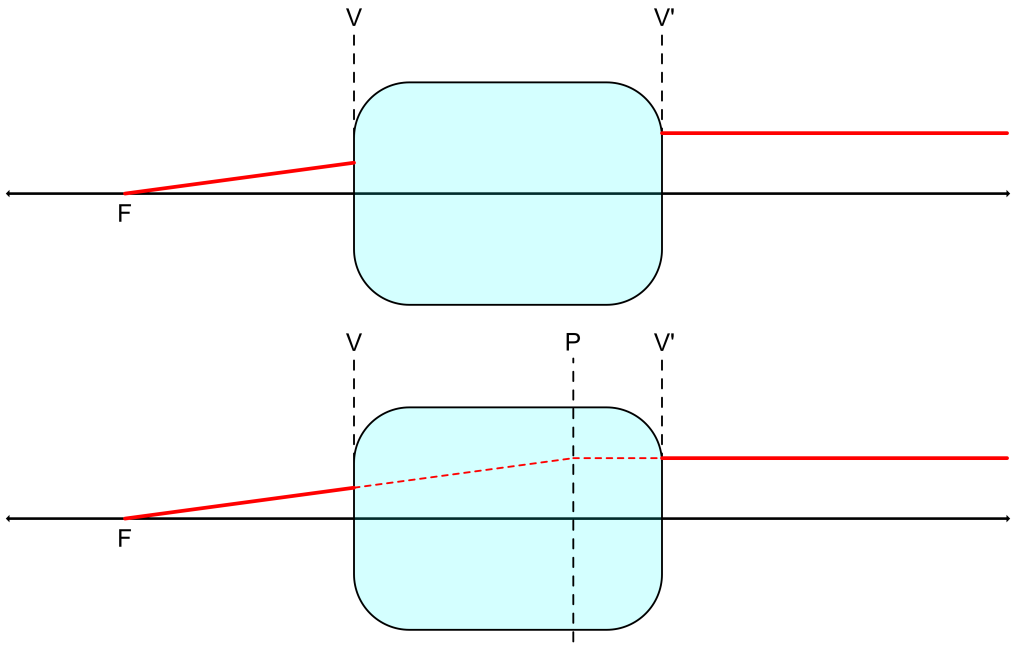
We consider the ray on the left of the system to be in "object space" as it has not yet passed through the optical system, and the ray on the right to be in "image space" as it has already passed through the optical system. The location where these two rays intersect is the **rear principle plane** (P').



Below is another example of an optical system and the location of P':



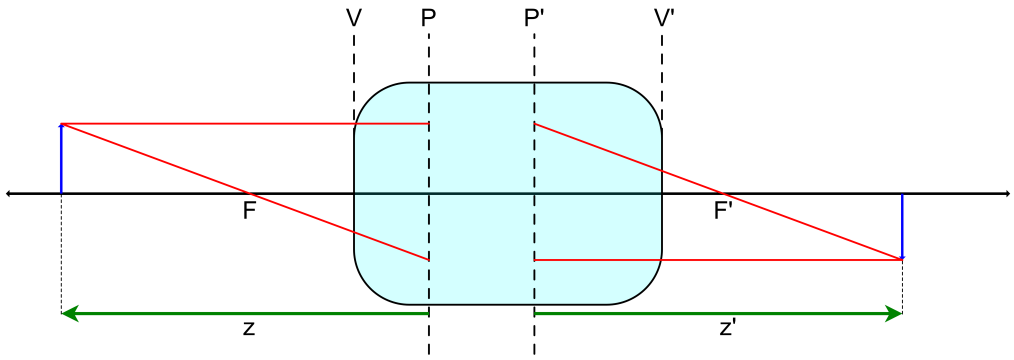
The **front principle plane** (P) and front focal point F are defined similarly, where we instead consider a parallel ray in image space.



The principle planes can be thought of as "the location where rays refract," thereby simplifying a complex optical system into two surfaces. The effective focal length of the system can be calculated using Gaussian reduction, after which the thin lens imaging equation can be used to find image and object locations.

$$\frac{n'}{z'} = \frac{1}{f} + \frac{n}{z} \tag{1}$$

Where z is the distance from P to the object, z' is the distance from P' to the image, n is the refractive index in object space, and n' is the refractive index in image space.



Gaussian reduction is a process which takes two lens systems with their own principle planes and focal lengths, and combines them into a single lens system. Let P_1, P_1', ϕ_1 denote the principle planes and power of the first system and P_2, P_2', ϕ_2 denote the principle planes and power of the second system. We need to know the index of refraction n between the two systems and the distance t between P_1' and P_2, from which we set

$$\tau = \frac{t}{n}. \tag{2}$$

which is the reduced distance between the two systems. Then, the power of the combined system is given by

$$\phi = \phi_1 + \phi_2 - \phi_1\phi_2\tau. \tag{3}$$

Then, the distance d from P_1 to the new front principle plane P is

$$d = n\frac{\phi_2}{\phi}\tau \tag{4}$$

and the distance d' from P_2' to the new rear principle plane P' is

$$d' = -n'\frac{\phi_1}{\phi}\tau \tag{5}$$

For this project, we will write a script that performs Gaussian reduction and implement various methods for using this tool. We will break down this script into several smaller tasks, which have various levels of difficulty and focus on different skills. However, this project is quite linear, so you generally need to have completed the previous task in order to move onto the next one.

1: Creating a Gaussian Lens System Class

For this project, we will use object oriented programming (OOP). This first task involves creating "Gaussian system" objects which store all the relevant information about a Gaussian system.

Part (a)

Create a class called "GaussianSystem," which stores all the relevant parameters for a Guassian lens system:

1. n: The refractive index of the material before the lens system.
2. n': The refractive index of the material after the lens system.
3. ϕ: The power of the lens, where
$$\phi = \frac{1}{f} \tag{6}$$
4. $\overline{VP}$: The distance *from* the front vertex V *to* the front principle plane P. Note that if P is to the left of V, then $\overline{VP}$ is negative, while if P is to the right of V, then $\overline{VP}$ is positive.
5. $\overline{V'P'}$: The distance *from* the rear vertex V' *to* the rear principle plane P'. Note that if P' is to the left of V', then $\overline{V'P'}$ is negative, while if P' is to the right of V', then $\overline{V'P'}$ is positive.
6. t: The total thickness of the system, equal to the distance between V and V'. This is not required for performing Gaussian reduction, but you may want to keep track of it for various other purposes.

In the object initialization method, all six of these variables should be inputs.

Part (b)

Create a few basic bound methods for this class. If any of the following are undefined, return "None."

1. Add an *efl* bound method which computes and returns the *effective focal length* f.
2. Add a *ffl* bound method which computes and returns the *front focal length* f_F. This is the distance from P to F, and is given by
$$f_F = -nf. \tag{7}$$
3. Add a *rfl* bound method which computes and returns the *rear focal length* f'_R. This is the distance from P' to F', and is given by
$$f'_R = n'f. \tag{8}$$
4. Add a *ffd* bound method which computes and returns the *front focal distance*, which is the distance from V to F.
5. Add a *bfd* bound method which computes and returns the *back focal distance*, which is the distance from V' to F'.

Part (c)

Add a string method to this class, which has the form:

```
def __str__(self):
```

In this method, return a string which contains any information you want to be displayed when you print an instance of the GaussianSystem class.

2: Creating Instances of the Class

Now, we want to create instances of the GaussianSystem classes based on user input, and check to see if they are behaving as expected.

Part (a)

Write a method which creates and returns an instance of the GaussianSystem class for a thin lens with an arbitrary focal length f. Include inputs for the object space index n and image space input n'. You will need to use the fact that for a thin lens, V, V', P, P' are all located at the lens. Generate an instance of this object with $f = 100\ mm$, $n = 1$, $n' = 1.4$, and make sure the bound methods from the previous problem all output the values which you would expect:

1. $efl = 100$
2. $ffd = ffl = -100$
3. $bfd = rfl = 140$

Part (b)

Write a method which creates and returns an instance of the GaussianSystem class for a thin lens with an arbitrary radius R. Include inputs for the object space index n and image space input n'. The power of a lens with radius R is given by the formula

$$\phi = \frac{n' - n}{R}. \tag{9}$$

Note that V, V', P, P' are all located at the point where the surface intersects with the optical axis. Generate an instance of this object with $R = 100\ mm$, $n = 1$, $n' = 1.4$, and make sure the bound methods from the previous problem all output the values which you would expect:

1. $efl = 250$
2. $ffd = ffl = -250$
3. $bfd = rfl = 350$

Part (c)

Write a method which takes an input of a list of N focal lengths and a list $N + 1$ of refractive indices, then generates a list of GaussianSystem instances for each surface. For instance, if we input

$$f = [100, 200, 140] \qquad n = [1, 1.4, 1.2, 1]$$

The code would first create a

1. GaussianSystem with $f = 100$, $n = 1$, and $n' = 1.4$,
2. GaussianSystem with $f = 200$, $n = 1.4$, and $n' = 1.2$,
3. GaussianSystem with $f = 140$, $n = 1.2$, and $n' = 1$,

and combine these in a list.

Part (d)

Write a method which takes an input of a list of N radii of curvature and a list $N + 1$ of refractive indices, then generates a list of GaussianSystem instances for each surface. For instance, if we input

$$R = [100, 200, 140] \qquad n = [1, 1.4, 1.2, 1]$$

The code would first create a

1. GaussianSystem with $r = 100$, $n = 1$, and $n' = 1.4$,
2. GaussianSystem with $r = 200$, $n = 1.4$, and $n' = 1.2$,
3. GaussianSystem with $r = 140$, $n = 1.2$, and $n' = 1$,

and combine these in a list.

3: Performing Gaussian Reduction

Part (a)

Write a method which takes as inputs two GaussianSystem objects and a distance t between the rear vertex of the first one and the front vertex of the second, and then calculates the reduced distance τ between the principle planes. Then, compute the variables ϕ, d, d' as described in the background section. Make sure to add a step which checks that n' for the first lens is the same as n for the second lens.

Next, have this method return a new GaussianSystem object whose front vertex V is the front vertex of the first GaussianSystem, and whose back vertex V' is the back vertex of the second GaussianSystem. To test your method, input the following Gaussian systems and combine them with a separation of $t = 10$, then check if the output of your code matches the expected output:

Attributes	Lens 1	Lens 2	Reduced lens
n_1	1.3	1.5	1.3
n_2	1.5	1	1
ϕ	0.02	0.04	0.05413333
$\overline{VP}$	2	-3	9.0443
$\overline{V'P'}$	-4	-5	-7.7094
t	7	3	20

Part (b)

Expand on your code from part (c) in the previous section such that there is an additional input of separation distances (which should be a list of length $N-1$), and reduces the system into a single Gaussian lens. Using the inputs from part (c), along with the thicknesses $t = [6, 8]$, check that you get the following output:

$$n = n' = 1 \qquad \phi = 0.02091837 \qquad \overline{VP} = 4.7154 \qquad \overline{V'P'} = -6.7610 \qquad t = 14$$

Part (c)

Expand on your code from part (d) in the previous section such that there is an additional input of separation distances (which should be a list of length $N-1$), and reduces the system into a single Gaussian lens. Using the inputs from part (d), along with the thicknesses $t = [6, 8]$, check that you get the following output:

$$n = n' = 1 \qquad \phi = 0.00164180 \qquad \overline{VP} = -12.1652 \qquad \overline{V'P'} = -22.6929 \qquad t = 14$$