Project 6: Matrix Methods, Ray tracing

Abstract

This project involves performing thick lens calculations using a matrix representation of optical elements. For this matrix operation, the movement of light involves two matrix operators, the refraction and translation transitions. We used these matrix operators to figure out the height of the ray from the principal axis after transitioning 10 cm and 30 cm from the middle of the lens and compared these results to our experimental results gotten from the lab performed in class. We found out that the distance from the principal axis for the ray 10 cm from the center of the lens is 0.04 cm and the one 40 cm from the center of the lens is 4.036 cm below the principal axis. We expected values 0 cm and 4.4 cm from our experimental measurements, this gives an error of 0.04 cm and 0.364 cm.

Description

Ray tracing mechanics is the use of matrices to monitor the propagation of a ray through a lens, be it thick or a thin lens. One mostly employs the use of two main operators when dealing with a lens, the refraction matrix and the translation matrix. The refraction deals with the bending of the ray as it passes from one medium to another, the translation deals with the movement of a ray from one position to another in the same medium. The ray tracing algorithm computes the position and angle of a ray with reference to the principal axis. The principal axis is one which a ray would pass through without refracting. We multiply the refraction and translation matrix together to come up with a system matrix that can be multiplied directly with the matrix defining the ray to come up with the final position of a ray. Multiplication of the successive matrices thus yields a concise ray transfer matrix describing the entire optical system. We can also go step by step by multiplying the ray with the appropriate matrix operator until we reach a desired position.

The ray tracing technique is based on two reference planes, called the input and output planes, each perpendicular to the optical axis of the system. At any point along the optical train an optical axis is defined corresponding to a central ray; that central ray is propagated to define the optical axis further in the optical train which need not be in the same physical direction (such as when bent by a prism or mirror). The transverse directions x and y (below we only consider the x direction) are then defined to be orthogonal to the optical axes applying. A light ray enters a component crossing its input plane at a distance x_1 from the optical axis, traveling in a direction that makes an angle θ_1 with the optical axis. After propagation to the output plane that ray is found at a distance x_2 from the optical axis and at an angle θ_2 with respect to it. n1 and n2 are the indices of refraction of the media in the input and output plane, respectively.

The ABCD matrix representing a component or system relates the output ray to the input according to

$$\begin{pmatrix} x_2 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} x_1 \\ \theta_1 \end{pmatrix}$$

For example, if there is free space between the two planes, the ray transfer matrix is given by:

$$\mathbf{S} = \begin{pmatrix} 1 & d \\ 0 & 1 \end{pmatrix}$$

where d is the separation distance (measured along the optical axis) between the two reference planes. The ray transfer equation thus becomes:

$$\begin{pmatrix} x_2 \\ \theta_2 \end{pmatrix} = \mathbf{S} \begin{pmatrix} x_1 \\ \theta_1 \end{pmatrix}$$

and this relates the parameters of the two rays as:

$$\begin{aligned} x_2 &= x_1 + d\theta_1 \\ \theta_2 &= \theta_1 \end{aligned}$$

Algorithm and Discussion

In this lab, we wanted to figure out the height of a ray propagated through a thick lens at 10 cm and 40 cm from the center of the lens. We passed the ray through the lens and measured its incident height at 40 cm and 10 cm before it hit the lens. We measured a height of 1cm at 40 cm from the center and a height of 1.2 cm at 10 cm from the center. We used trigonometry to figure out the angle of incidence.

$$\begin{pmatrix} y_1 \\ \theta_1 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{0.2}{30} \end{pmatrix}$$

We use small angle approximations, where $\sin\theta \approx \theta$ We are going to use the translation matrix operator to translate from the starting point to the curved surface, use the refraction matrix operator to refract through the curved surface, translate through the lens thichness, refract out of the other lens curved surface and translate to points that we need to measure the height at. The translation and refraction matrix operators are as follows:

$$\begin{aligned} \hat{R}\left[r_0\right] &= \begin{bmatrix} 1 & 0 \\ \frac{n_1 - n_s}{Rn_s} & \frac{n_1}{n_s} \end{bmatrix} \begin{bmatrix} y_0 \\ \alpha_0 \end{bmatrix} \\ \hat{T}\left[r_0\right] &= \begin{bmatrix} 1 & L \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ \alpha_0 \end{bmatrix} \end{aligned}$$

Where n_i and n_s are the indices of refraction on the left and right (respectively) of the refractive interface. R is the radius of curvature of the refractive interface (positive if the center of curvature is to the right, negative to the left). y_0 is the height of the ray before the transition, α_0 is the angle (in radians) of the ray before the transition. L is the length of the translation. We can get a system matrix by multiplying all refraction and translation matrix operators from the right to the left.

Implementation and Code

In [23]:
```
!matplotlib inline
import pandas as pd
import matplotlib.pyplot as pl
import numpy as np
import sympy as sp
import random
```

In [24]:
```
ng = 1.5   # index of glass
na = 1.0   # index of air
h = 1   # start with ray 1.0 cm above axis
```

In [25]:
```
hp = 10.625/10
hc1 = 17.872/10
hc2 = 15.986/10
h1 = hc1 - hp
h2 = hc2 - hp
b = 56.1/10
d = (0.5 * b)/np.cos(30 * (np.pi / 180.0))
d
```

Out[25]:
```
3.2389350101538006
```

Figuring out the radius of curvature using the intersecting chord theorem which states that when two chords intersect each other inside a circle, the products of their segments are equal. Having gotten the distance h by using the spherometer we computed the radius of curvature of the more and less curved surface below

In [26]:
```
r1 = (d**2 + h1**2) / (2 * h1)
r2 = (d**2 + h2**2) / (2 * h2)
print ("Radius of Curvature for most curved side:" , r1, "cm")
print ("Radius of Curvature for less curved side:" , r2, "cm")
```

```
Radius of Curvature for most curved side: 7.6003105535393958 cm
Radius of Curvature for less curved side: 10.053235321768327 cm
```

We figured out the angle of incidence of the incoming ray by using trigonometry

In [27]:
```
l = 27.9
l1 = 28.9
l2 = 29.1
alpha = (l2 - l1)/30
```

Computing the ray matrix operations by translating to the more curved surface, refracting through the more curved, translating through the lens, refracting out of the less curved surface and translating to the distances measured

In [28]:
```
ray1 = np.array([[h],[alpha]])

ray1
```

Out[28]:
```
array([[1.        ],
       [0.0066667]])
```

In [30]:
```
T1 = np.array([[1.0, 38.9],
               [0.0, 1.0]
               ])

T1
print (sp.latex(T1))
```

```
[[ 1.   38.9]
 [ 0.   1. ]]
```

In [8]:
```
ray2 = T1.dot(ray1)
ray2
```

Out[8]:
```
array([[1.25933333],
       [0.0066667]])
```

In [9]:
```
R1 = np.array([[1.0, 0.0],                  # entering the more curvy surface
               [(na-ng)/(ng*r1), na/ng]])
R1
```

Out[9]:
```
array([[ 1.        ,  0.        ],
       [-0.04385786,  0.66666667]])
```

In [10]:
```
ray3 = R1.dot(ray2)
ray3
```

Out[10]:
```
array([[ 1.25933333],
       [-0.05078722]])
```

In [11]:
```
T2 = np.array([[1.0, 2.2],
               [0.0, 1.0]                   # translating through the thick lens
               ])
T2
```

Out[11]:
```
array([[1. , 2.2],
       [0. , 1. ]])
```

In [12]:
```
ray4 = T2.dot(ray3)
ray4
```

Out[12]:
```
array([[ 1.14760146],
       [-0.05078722]])
```

In [13]:
```
R2 = np.array([[1.0, 0.0],                  # leaving the less curved surface
               [-(ng-na)/(na*r2), ng/na]])
R2
```

Out[13]:
```
array([[ 1.        ,  0.        ],
       [-0.04973974,  1.5       ]])
```

In [14]:
```
ray5 = R2.dot(ray4)
ray5
```

Out[14]:
```
array([[ 1.14760146],
       [-0.13326222]])
```

In [15]:
```
T3 = np.array([[1.0, 8.9],
               [0.0, 1.0]                   # translating 8.9 cm to the right
               ])
T3
```

Out[15]:
```
array([[1. , 8.9],
       [0. , 1. ]])
```

In [16]:
```
ray6 = T3.dot(ray5)
ray6
```

Out[16]:
```
array([[-0.03843228],
       [-0.13326222]])
```

The first entry of the matrix shows that at 10 cm from the center of the lens, the ray is about 0.038 cm below the principal axis.

In [17]:
```
T4 = np.array([[1.0, 30],
               [0.0, 1.0]                   # translating 30 cm to the right
               ])
T4
```

Out[17]:
```
array([[ 1., 30.],
       [ 0.,  1.]])
```

In [18]:
```
ray7 = T4.dot(ray6)
ray7
```

Out[18]:
```
array([[-4.03629882],
       [-0.13326222]])
```

The first entry on this matrix tells us that at 40 cm from the center of the lens, the ray is at a distance of 4.036 cm below the principal axis

In [19]:
```
M = T4.dot(T3.dot(R2.dot(T2.dot(R1.dot(T1)))))
M
```

Out[19]:
```
array([[ -3.40377804, -94.87811691],
       [ -0.11072727,  -3.38024237]])
```

In [20]:
```
M.dot(ray1)
```

Out[20]:
```
array([[-4.03629882],
       [-0.13326222]])
```

Multiplying all the refraction and translation ray matrix operators, we come up with a general matrix operator, M, that we can multiply the intial ray with and still end up with the same matrix when we carry out the matrix multiplication step by step

Our experimental figures showed that we should get at height of 0cm and 4.4 cm below the principal axis for positions 10 cm and 40 cm from the center of the lens respectively. This was pretty close to what we got from our matrix operations which we got to be 0.04 and 4.04 cm respectively.

Conclusion

In this project we propagated a ray through a lens of thickness, 22mm and tried figuring out the height this ray would be at from the principal axis when measured at positions 10 cm and 40 cm from the center of the lens. We compared these results to experimental results. We did this by using two matrix operators, the translation and refraction matrix operations. Results from the matrix operations was that the ray was 0.04 cm and 4.036 cm below the principal axis when it is 10 cm and 40 cm from the center of the lens respectively. Our experimental results showed we should get values 0cm and 4.4 cm below the principal axis at this positions. These values are pretty close with really small errors of 0.04 cm and 0.364 cm for positions 10 cm and 40 cm respectively.

Reference

Wikipedia (2010). Ray tracing. https://en.wikipedia.org/wiki/Ray_transfer_matrix_analysis

Steve Spicklemire. (2014). Matrix methods: Optics with Matrix. Scientific Computing 1.

Steve Spicklemire (2014). Matrix methods https://github.com/sspickle/sci-comp-notebooks/blob/master/P06-MatrixMethodsOptics.ipynb