Modelling and Optimisation of Optical Systems

Júlia Tompa

December 2023

1 Introduction

This semester, I continued my work regarding the modelling and optimisation of optical systems, perfecting and expanding the C++ code I have created over the last two semesters. I added a lot of new features – many of which are centred around the new *Material* class – and used the thus improved code to design an optimised achromatic lens. Furthermore, I optimised the parameters of an old Tessar objective, using materials that are available today - as opposed to the ones that were originally used for this model. This optimisation is not yet finished, I plan to complete the process in my thesis.

2 The structure of the C++ project

First, I would like to give a brief outline of the structure of the expanded C++ project I created. It includes a *.cc* file with the *main()* function, while classes, structs and their member functions are contained in the following five header files: *euclidean.h, errors.h, material.h, surface.h* and *objective.h.* The project is made for modelling optical systems that contain flat and spherical surfaces, simulating light rays passing through them, analysing optical aberrations, and visualisation, for which purpose I used the Cairo graphic library. The program's ability to handle light rays of different wavelengths and calculate the refractive index of a lens based on wavelength is a significant improvement. The program can be used to optimise the parameters of optical systems with the help of repeated simulations.

2.1 *euclidean.h*

This section of the code is designed to simulate the three-dimensional Euclidean space, using two structs: *Vec3* and *Ray.* The former is a straightforward model of a vector from the Euclidean space with corresponding operations, while the latter – a representation of a ray of light – has been given a third parameter in addition to the two already existing ones: it can now be defined with a base (its starting point in space), a direction and a wavelength.

2.2 *material.h*

The addition of the struct *Material* is a significant improvement to my project. It not only enables us to specify the material of a lens by name, but also allows us to simulate light rays of different wavelengths. This feature is crucial in considering chromatic aberrations during the optimisation process. Chromatic aberrations occur when different wavelengths are not focused to the same point, which can happen because the refractive index of an optical medium – an indication of its light-bending ability – may vary with the wavelength of the light. This means that it is insufficient to define a lens using only one number as its refractive index since this number depends on the parameters of the light ray passing through it.

The refractive index of a material can be calculated, given the wavelength of the ray, using – for example – Sellmeier's dispersion formula [3]:

$$n^2 - 1 = C_1 + \sum_{i=1}^{8} \frac{C_{2i}\lambda^2}{\lambda^2 - C_{2i+1}},$$

where the coefficients C_i $(i = 1, ..., 17)$ are specific to the material. For the program to calculate the refractive index, it requires access to these parameters for each material used. Fortunately, an online database exists [3] that contains this information for most types of glass that are currently in use, however, it also contains pages of information that are not relevant to our case, making it complicated to navigate. As a solution, I wrote a Python script using the *BeautifulSoup* class from the *bs4* module, which made it possible to parse the whole database and extract only the data needed, namely the type of the used dispersion formula, the 17 coefficients and the Abbe number (an approximate measure of the material's dispersion) for each glass type. This data is collected in a JSON file, which is read by the program once and then stored as a static data member of the *Material* class.

An instance of the class *Material* can be created with one parameter: the name of the glass (e.g. "N-SF14"). The two member functions of this class are the following:

- The constant member function *ri* determines the refractive index of the material given the wavelength of the light ray (587.6 nm by default).
- The function *guessMaterial* takes two arguments: a refractive index value at 587.6 nm and an Abbe-number. Given these two parameters, the function identifies a similar known material (within a difference of 0.01 for the refractive index and 0.5 for the Abbe number). This function is useful because, when analyzing existing objectives, we often need to identify a glass type knowing only these two parameters. The situation can be complicated even further when the glass used in those objectives is no longer in production, which creates the need to identify similar materials that can be used as a replacement.

2.3 *surface.h*

The *surface.h* header file contains the *Surface* class used for modelling optical surfaces, alongside its two derived classes: *FlatSurface* and *SphericalSurface*. An instance of the *FlatSurface* class can be defined by a normal vector, the centre of the surface, its size (in millimetres), the material on the entry side of the medium and the material on its exit side, whereas the constructor of the

SphericalSurface class requires the centre of the sphere, its radius, the direction of the surface from the centre of the sphere, the angle of aperture, and finally, similarly to the flat surfaces, the material on its entry and exit sides.

For both surface types, the program is able to calculate the intersection point with a light ray, and the path of a ray after it passes through the surface.

2.4 *objective.h*

All the features mentioned above come together in the class called *Obj* (short for objective). An instance of this class is in fact nothing more than instances of the two surface classes stacked after one another, positioned perpendicular to the x-axis. Two important new additions to this class are the following member functions:

- the function *focus* calculates the focal point of the objective, taking the wavelength of the passing light ray as an argument (since, as a consequence of chromatic aberration, the focus will be different based on the wavelength),
- the function *focalLength* returns the focal length (that is, the distance from the focal point to the front principal plane) of the objective for a given wavelength.

The function *drawPaths* has been updated so that it is now suitable for visualising different coloured light rays passing through an optical system. It does this with the help of the Cairo graphics library.

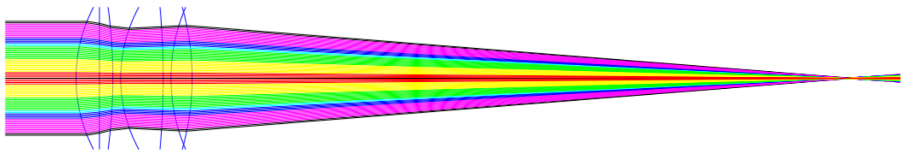

Figure 1: Light rays of different wavelengths passing through a Tessar objective

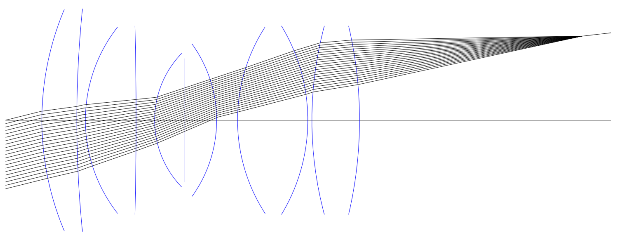

Figure 2: Light rays passing through a Biotar objective

3 Designing an achromat

3.1 Optical aberrations

When designing an optical system, we must consider the inaccuracies that arise due to the fact that the paraxial theory – where light rays from any given point of an object would pass through the lens and come together at a single point in the image plane – is not a completely accurate model, and real lenses do not focus light into a single point. These inaccuracies in imaging are known as optical aberrations, and they are categorised into two types:

- monochromatic aberrations and
- chromatic aberrations.

Monochromatic aberrations occur due to the geometric properties of the surfaces in the optical system, and they also appear when using monochromatic light. The most common monochromatic aberrations are *spherical aberration*, *coma*, *astigmatism*, *field curvature* and *image distortion*.

Last year, I used my C++ program to visualise and analyse some of these aberrations in a Tessar-, a Biotar- and a Biogon-type objective.

Chromatic aberrations, on the other hand, do not appear when monochromatic light is used. They are caused by dispersion, the variation of a glass's refractive index with wavelength. Due to dispersion, light rays of different wavelengths are not focused to the same point, which causes the image of a point to spread out.

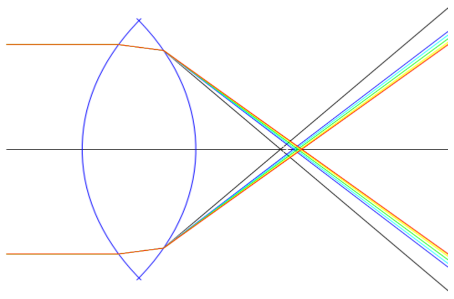

Figure 3: Chromatic aberration in a single equiconvex lens

This phenomenon can be observed in Figure 3, where I simulated some light rays of different wavelengths passing through an equiconvex lens made of the material *K7* (a type of crown glass). The difference in the focal points of different coloured rays is clearly visible.

3.2 What is an achromatic lens?

An achromatic lens or achromat is designed to reduce the effect of chromatic aberration. The goal is for the difference between the focal points of two selected wavelengths (usually a shade of red and blue) to be as small as possible [1]. Typically, an achromatic lens consists of two elements: an equiconvex part made out of a type of crown glass, and a concave part made out of flint glass.

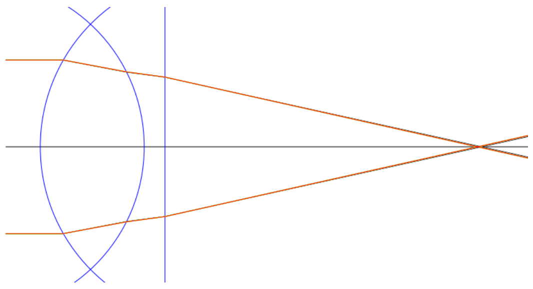

Figure 4: Chromatic aberration in an achromatic lens

3.3 Optimising the parameters of an achromat

When designing an achromat, my goal was to reduce the difference between the focal point of the wavelengths 486.1 nm and 656.3 nm, with the following constraints:

- The achromat should consist of two elements: an equiconvex lens made of *K7*, and a lens made of *F2* (a type of flint glass) with a planar surface.
- The first lens should be 5 mm thick, the second 1 mm.
- The focal length of the achromat should be 100 mm.

Since the focal lengths and focal points of an optical system can be calculated using paraxial approximation, this optimisation problem could be solved analytically, without the need for simulation. My reason for choosing the latter option was, however, that I wanted my method to work for more complicated optical systems and more complex optimisation problems as well.

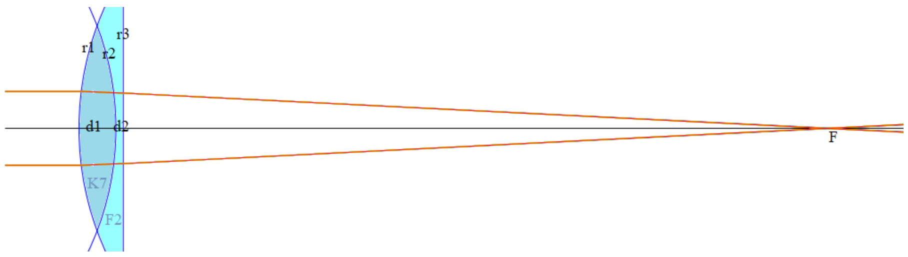

Figure 5: Achromatic doublet

My task was to determine the radii $r1$ and $r2$ (as shown in Figure 5) so that the focal length is 100 mm and the aforementioned focal points are as close to each other as possible, with the other three parameters set to $r3 = \infty$, $d1 = 5$ and $d2 = 1$. (A radius of ∞ means that the surface is flat, while a positive radius indicates a spherical surface whose centre lies on its exit side. On the other hand, if the radius is negative, it means that the centre of the sphere is located on the entry side of the surface.) In order to do this, I first wrote a function (*find_r1*) that calculated the value of $r1$ from any given $r2$ so that the focal length of the lens at 656.3 nm remained 100 mm. This function

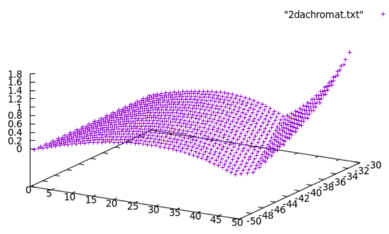

Figure 6: The difference between the focuses as a function of $r1$ and $r2$

implements binary search to find the value of $r1$ (between 1 and 100 mm) that results in a focal length within 0.0001 mm of the desired length. Binary search finds the optimal $r1$, because, as seen in Figure 7, if all other parameters are set, then the difference between 100 mm and the focal length of the lens is a quasiconvex function of $r1$. In Figure 7, the x-axis displays the values of $r1$, while the y-axis shows the amount by which the focal length deviates from 100 mm. $r2$ has been set to -22. We can observe that the optimal value of $r1$ for $r2 = -22$ is around 35 mm.

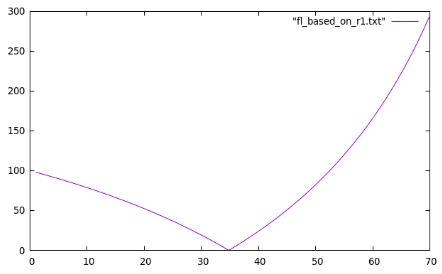

Figure 7: The amount by which the focal length deviates from 100, as a function of $r1$, where $r2 = -22$.

To proceed further, we needed to find a value for $r2$ that would minimise the difference between the focal points of the two chosen wavelengths. Note that the value of $r1$ is dependent on the value of $r2$, as the focal length at 656.3 nm should always be equal to 100. We can determine the suitable value for $r1$ with the help of the function *find_r1*.

Once again, we were dealing with a quasiconvex function as shown in Figure 8. To get this figure, I plugged in values for $r2$ from -100 to -10, used the function *find_r1* to determine $r1$ for which the focal length is 100, and then calculated the distance between the focal points of the selected wavelengths. We can note that the optimum is around $r2 = -38$. To get a precise result, I implemented a type of line search called the Golden Section Method [4].

3.4 The Golden Section Method

In nonlinear optimisation, we often rely on calculating derivatives. However, in this case, our target function is not defined analytically, which makes us unable

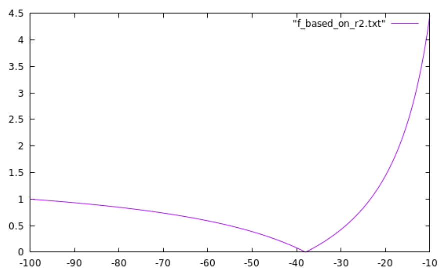

Figure 8: The target function as a function of $r2$

to use any method based on derivatives. Luckily, the assumption of strict quasiconvexity allows us to find the optimum using the Golden Section Method, which is an effective way of finding the minimum of a strictly quasiconvex univariate function.

To understand the fundamentals of this algorithm, we need to define the concept of an interval of uncertainty. An interval $[a, b]$ is called the interval of uncertainty if a minimum point of our function lies in $[a, b]$, though its exact value is not known. The search procedure relies on the principle that by evaluating a strictly quasiconvex function at two points inside the uncertainty interval, we can exclude portions of the interval that do not contain the minimum and so the interval of uncertainty can be reduced.

Consider, for instance, the line search problem to minimize $f(x)$ subject to $a \leq x \leq b$, so the initial period of uncertainty in $[a, b]$. Let $\lambda, \mu \in [a, b]$ such that $\lambda < \mu$. If $f(\lambda) > f(\mu)$, then the minimum lies in $[\lambda, b]$. If $f(\lambda) < f(\mu)$, then the minimum lies in $[a, \mu]$. This means that we can reduce the interval of uncertainty by evaluating the function at two points. Different methods exist for selecting these two points, the Golden Section Method being one of the most efficient. For this method, the value of the reduction ratio:

$$\frac{\text{Length of interval of uncertainty after } v \text{ observations are taken}}{\text{Length of interval of uncertainty before taking the observations}}$$

is equal to $(0.618)^{v-1}$.

3.5 The result

The implementation of the Golden Section Method yielded the following values for the two radii:

- $r1 = 40.0526\,\text{mm}$,
- $r2 = -37.968\,\text{mm}$.

The optimised achromat resulted in a significant improvement, with a difference of $1.55 \cdot 10^{-7}$ between the focal lengths for the two selected wavelengths, compared to the initial value of approximately 0.667.

4 Analysis and redesign of a Tessar objective

After completing the optimisation of the achromat, I began working with the same Tessar objective whose properties I analysed and visualised last year. The information available to me regarding this objective includes the radii of the 7 surfaces in the objective, the distance between them, the refractive index of each material used (at 587.6 nm) as well as the Abbe-numbers of the materials.

The design of this objective was created in the last century, and despite having its parameters, we would be unable to build it today. The reason for this is that the glasses used in its lenses are no longer in production. Therefore, my first goal was to find materials that resemble the ones used in the design. With the help of the previously mentioned function *guessMaterial*, I managed to locate materials with refractive indexes within 0.01 and Abbe-numbers within 0.5 of the original glasses.

My next goal is to change some parameters of the objective in order to correct some optical aberrations in its imaging while setting its focal length to 100 mm. This optimization is partially a task for the future – I will complete the process and configure the parameters to correct the aberrations in my thesis. I have 13 parameters to optimise: the radii of the 7 surfaces and the distances between them. To achieve a focal length of 100 mm, I wrote a function similar to the one called *find_r1* that I used for designing the achromat. This new function (*find_r7_tessar*) also utilises binary search to determine the radius of the last surface in the Tessar objective for any given combination of the other 12 parameters, so that the focal length is 100 mm.

I analysed the objective according to the following aspects:

- axial chromatic aberration,
- spherical aberration and
- field curvature.

In terms of reducing the effect of axial chromatic aberration, I aim to minimise the difference between the focal points of the two wavelengths 486.1 nm and 656.3 nm, similarly to the case of the achromat.

Spherical aberration occurs when light rays parallel to the x-axis are not focused to one point after passing through an optical system. To measure this, I simulated such rays passing through the objective and measured the biggest difference between their intersections with the x-axis. My target is to reduce this difference.

Field curvature or Petzval field curvature (named after Hungarian mathematician, inventor and physicist Joseph Petzval) is the optical aberration where the image of a flat object perpendicular to the x-axis is not focused on a flat image plane, instead, the image is curved. It can both be examined analytically and through simulation. I chose the former option: with the help of the so-called *Petzval sum*, we can calculate the curvature of the curved image field based on the radii and the refractive indices of the lenses in the optical system [2].

The *Petzval sum* of an objective is

$$\sum_i \frac{n_{i+1} - n_i}{r_i n_{i+1} n_i}, \tag{1}$$

where r_i is the radius of the i-th surface and the n-s are the indices of refraction on both sides of the surface.

After modelling the Tessar objective with the help of my program, using the materials I found to be similar to the originals, I calculated the amount of the above optical aberrations, with the following results:

- the difference between the focus of the two wavelengths was 0.01 mm,
- the amount of spherical aberration was 0.397 mm
- and for the Petzval sum, I obtained a value of 0.00317.

These values are quite good, so to test my optimising algorithm, I changed the materials in the objective to ones that are less similar to the original glasses. For this changed – and not yet optimised – optical system, the values turned out to be as follows:

- the difference between the focus of the two wavelengths was 1.89 mm,
- the amount of spherical aberration was 0.59 mm
- and the Petzval sum was 0.0017.

My current goal is to find values for the 12 parameters for which these values are as small as possible. To do this, I defined a target function, which is equal to a linear combination of these three values. I chose the coefficients so that the order of magnitude is approximately the same for all three. To sum up, I have to solve a nonlinear optimisation problem in the 12 dimensional Euclidean space, where my target function is not given analytically - it is a so-called "black-box function".

As I mentioned, this optimisation is partly a task for the future. However, I would like to give a brief outline of how I proceeded so far. Since the objective function cannot be calculated analytically, I opted for a multi-dimensional search method that does not use derivatives. First, I defined a starting point (p_0) and calculated the objective function at that point. Then, I implemented a line search along all 12 coordinate axes. This means that the program calculates the target value at 100 points along each axis, starting from p_0 and stopping at a distance of δ (for which I have tried different values) from p_0, thus staying inside a 12 dimensional brick. (It is a brick and not a cube since I used different δ values for the parameters representing the radii of the surfaces, and those representing the distances between them.) After that, it calculates the objective value at each vertex of this brick and chooses the next point (p_1) so that its objective value is the smallest of all the examined points. It then continues to search along the coordinate axes inside the same brick (now starting from p_1), until the improvement of the objective function stops or slows down to the extent that it only decreases by a small ε between two rounds. Then, as a next step, a new brick can be defined around the current optimal solution, and the process can be repeated.

So far, I managed to find a solution for which

- the difference between the focus of the two wavelengths is 0.65 mm,
- the amount of spherical aberration is 0.3 mm

- and the Petzval sum is 0.0014.

Even though the values improved, I will still need to work on the algorithm to get better results.

5 Summary

My program can be used to model, visualise and analyse optical systems, and with its help, we can also optimise the parameters of simpler optical systems (e.g. an achromatic doublet), but in order to use it for more complicated optimisation problems, a faster and more efficient algorithm needs to be implemented. My plan for the next semester is to implement such an algorithm as well as to familiarise myself with the full consequences of the wave nature of light so I can better understand the geometry used in the modelling procedure.

References

[1] *Achromatic lens.* URL: `https://en.wikipedia.org/wiki/Achromatic_lens`.

[2] *Petzval field curvature.* URL: `https://en.wikipedia.org/wiki/Petzval_field_curvature`.

[3] Mikhail Polyanskiy. *refractiveindex.info - online database.* URL: `https://github.com/polyanskiy/refractiveindex.info-database`.

[4] M. S. Bazaraa - H. D. Sherali - C. M. Shetty. *Nonlinear Programming - Theory and Algorithms.* John Wiley & Sons, Inc., 1993. ISBN: 0-471-55793-5.