**UNIVERSITY OF ECONOMICS HO CHI MINH CITY**

**NATIONAL UNITED UNIVERSITY**

**INSTITUTE OF INTELLIGENT & INTERACTIVE TECHNOLOGIES**

**A logo with a green triangle and white text

AI-generated content may be incorrect.**

A blue and black logo

AI-generated content may be incorrect.

**REPORT**

**APPLIED CALCULUS**

**Simulation and Analysis of Spherical Aberration in a Two-Surface Lens System with Aspherical Lens Optimization**

**PROF: LI-JEN HSIAO**

|  |  |  |
| --- | --- | --- |
| **STUDENT ID** | **NAME** | **CHINESE NAME** |
| U1399309 | TAN HONG PHONG | 新紅風 |

**1. Objectives**

This study focuses on investigating and simulating the refraction of light as it passes through a system composed of two spherical surfaces—especially aspherical lenses. Using numerical methods and geometric-optics formulas, we set the following specific goals:

**1.1 Simulate the refraction of a light ray through two spherical surfaces (lens) in Python**

Our first objective is to construct a mathematical model and then implement it in Python to reproduce the refraction of a light ray as it travels through two consecutive curved surfaces representing the two faces of a lens. We will begin with ideal spherical surfaces to establish a solid foundation for applying Snell’s law at each interface. The focus is on accurately computing the ray path after each refraction, ensuring continuity and adherence to fundamental optical principles.

**1.2 Compute and visualize the ray path based on Snell’s law for non-spherical (aspherical) lenses**

Once the spherical-surface model is complete, we will extend the simulation to handle aspherical lens surfaces. This requires more sophisticated algorithms to determine the surface normal at every point of incidence, which is no longer trivial as in the spherical case. Using these normals and Snell’s law, we will recalculate the refraction angle and determine the precise ray trajectory. The entire process will be visualized graphically, allowing viewers to observe how the ray direction changes when passing through aspherical surfaces.

**1.3 Study the influence of optical parameters and real-world applications**

The main goal here is to examine how optical parameters—incident angle, refractive index, surface curvature, and lens spacing—affect the ray path and focal point. We will analyze how aspherical lenses reduce aberrations (e.g., spherical aberration) to enhance image quality, thereby informing practical optical-design applications.

These objectives will be pursued sequentially to ensure rigor and logical coherence throughout the project, ultimately yielding a comprehensive simulation model and deep insights into light-refraction phenomena in optics.

**2. Fundamental Theory**

This section presents the fundamental principles and mathematical formulas that form the foundation for simulating the refraction of light through spherical and aspherical surfaces.

**2.1. Snell’s Law**

Snell’s Law, also known as the Law of Refraction, is a basic principle that describes how the direction of a light ray changes when it passes through the interface between two transparent media with different refractive indices. The law is expressed as:

Where:

● : Refractive index of the incident medium (e.g., air),

● : Refractive index of the refractive medium (e.g., glass),

● : Angle of incidence (between the incoming ray and the surface normal),

● : Angle of refraction (between the refracted ray and the surface normal).

The **surface normal** is a line perpendicular to the surface at the point of intersection with the ray. Accurately determining this normal is crucial for correctly applying Snell’s Law, especially for **aspherical surfaces** where the curvature varies across the surface.

**2.2. Sagitta Equation for Spherical and Aspherical Surfaces**

The **sagitta** (or sag) of a spherical or aspherical surface refers to the height of the surface at a given point relative to a plane that passes through the vertex and is perpendicular to the optical axis. It is the mathematical basis used to define the shape of a lens surface.

**2.2.1. Sagitta Equation for a Spherical Surface**

For a **spherical surface**, the sagitta z(r)z(r)z(r) at a radial distance rrr from the optical axis is given by:

Where:

* Sagitta (surface height) at radial distance rrr,
* : Radial distance from the optical axis,
* : Radius of curvature of the spherical surface.

This equation describes the geometry of a spherical surface and serves as the basis for calculating normals and ray paths when light interacts with such surfaces.

**2.2.2. Sagitta Equation for Aspherical Surfaces**

To describe **aspherical surfaces**, a more general sagitta equation is used, allowing precise control over the surface shape to reduce optical aberrations. This equation typically includes higher-order coefficients and is expressed as:

Where:

● : Sag (surface height) at radial position r,

● : Radius of curvature of the surface (positive for convex, negative for concave),

● : Conic constant defining the surface type (e.g., sphere, paraboloid, ellipsoid, etc.),

● 𝑟: Radial distance from the optical axis.

When , the surface is spherical. When , it is parabolic. Other values produce elliptical or hyperbolic surfaces.

**2.3. Derivative of the Sagitta for Aspherical Surface (without aspheric coefficients)**

For an aspherical surface defined by the sagitta equation *excluding* higher-order aspheric terms, the surface profile is given by:

To determine the slope of the surface at a radial distance , we take the derivative of with respect to . Let us denote:

Then, the derivative ​ becomes:

This expression represents the slope of the surface at a point rrr, which is needed to compute the surface normal.

**2.4 Method for Calculating the Intersection Point**

To determine the intersection between a straight line and an aspherical lens surface, we need to solve the nonlinear equation:

where

and the sag function is defined by the sagitta formula of the aspherical lens surface:

with:

as the radius of curvature of the lens surface,

as the conic constant (geometry constant),

defining the straight line equation .

This equation is nonlinear due to the square root and the complex expression involving through . Therefore, it cannot be solved analytically using standard algebraic methods. Numerical methods are required to find an approximate root xxx such that .

**2.4.1 Overview of Brent's Method**

Brent's method is a numerical algorithm for solving equations of one variable developed by Richard Brent in 1973. It combines the advantages of:

The bisection method: always convergent but slow,

The secant method: faster but not always convergent,

Inverse quadratic interpolation: fast and accurate.

Reference:<https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.root_scalar.html>

**2.4.2 Principle of Operation**

1. **Initial condition:** The algorithm starts with an interval where is continuous and satisfies , meaning the function changes sign and thus has at least one root by the Intermediate Value Theorem.
2. **Iteration process:**  
   The algorithm chooses a point near the root by applying interpolation (secant or inverse quadratic) to accelerate convergence.  
   If the interpolation step is unsafe (e.g., the new point is outside the interval or error does not improve), it switches to bisection to guarantee convergence.  
   This iterative process continues, updating the interval until the error is smaller than a given tolerance or the maximum iteration count is reached.

**2.4.3 Advantages of Brent's Method**

Guaranteed convergence: Always finds a root within the given interval if the function is continuous and changes sign.

Fast convergence: Usually faster than the pure bisection method.

No derivative required: Unlike Newton's method, Brent's method does not need the derivative of the function, convenient for complicated or non-differentiable functions.

Stable: It self-adjusts between interpolation and bisection steps to reduce numerical errors.

**2.4.4 Application in root\_scalar function**

In Python, the SciPy library provides the function root\_scalar with the parameter method='brentq' to use Brent's method for root finding.

Basic syntax in python:

: the function to find the root of,

interval containing the root,

: indicates if the algorithm converged,

: the found root.

**Reason for Choosing Brent's Method in the Lens-Line Intersection Problem:**The intersection equation is a complex nonlinear equation that cannot be solved by closed-form formulas. The function may be non-differentiable or difficult to differentiate, making derivative-based methods unsuitable. Brent's method guarantees finding a root within a bounded interval with high accuracy, combining stability and fast convergence. Therefore, it is well-suited for determining the intersection point between a straight line and an aspherical lens surface in engineering applications.

**2.5. Light Ray Path Calculation (Ray Tracing)**

**Ray tracing** is a core geometrical method used to track the path of light rays through an optical system. The process involves iteratively calculating the ray's interaction with each lens surface and updating its direction based on Snell’s Law.

**The main steps include:**

1. **Define the light ray**:  
   A ray is defined by a starting point and a direction vector
2. **Find the intersection with the surface**:

Use the parametric form of the ray:

Solve for the intersection point with the surface equation (spherical or aspherical).

For aspherical surfaces, this usually involves solving a nonlinear equation, often requiring **numerical methods** such as the Newton-Raphson method.

1. **Compute the normal vector**:  
   At the intersection point, calculate the surface's normal vector as described in section 2.3.
2. **Apply Snell’s Law**:  
   Use the law of refraction to determine the new direction of the ray after passing through the surface.
3. **Repeat**:  
   The new ray (starting from the intersection point and in the updated direction) is traced to the next surface, and the above steps are repeated.

The combination of **geometrical methods** (ray definition and theoretical intersection) and **numerical techniques** (solving complex surface equations and optimization) is essential, especially when dealing with aspherical surfaces, to ensure accuracy and flexibility in simulation.

**3. Introduction / Reason for Choosing the Topic**

**3.1. In-depth Understanding of Geometric Applications and Optical Simulation Algorithms**

This topic offers an excellent opportunity to deeply explore the strong relationship between geometry and optics. By simulating the path of light rays, we directly apply geometric concepts such as curves, surface normals, and intersection points. At the same time, developing a simulation algorithm requires understanding how to transform physical principles into iterative computational steps, thereby strengthening algorithmic thinking and problem-solving skills. This is particularly important when dealing with complex aspherical surfaces, where simple analytical solutions are no longer effective.

**3.2. High Practical Application in Optical System Design**

The knowledge gained from this project is highly applicable in many areas of optical engineering. The ability to simulate and analyze the path of light rays is fundamental for designing and optimizing complex optical systems. For example, in the manufacturing of microscopes or cameras, understanding how lenses refract light helps engineers minimize aberrations and enhance image sharpness and contrast. In particular, research on aspherical lenses opens up the potential to design more compact, lightweight, and high-performance optical systems compared to traditional spherical lens designs.

**3.3. Strengthening Interdisciplinary Knowledge: Mathematics, Physics, and Programming**

Conducting this project provides a comprehensive interdisciplinary learning experience by reinforcing knowledge across three core domains:

**Applied Mathematics**: Including analytical geometry, calculus (derivatives for computing normals), and numerical methods for solving complex equations.

**Physics**: Mastering fundamental principles of geometrical optics, especially Snell’s law and concepts like refraction, convergence, and aberrations.

**Python Programming**: Developing programming skills in Python using computational and visualization libraries (such as NumPy and Matplotlib) to build models, perform calculations, and effectively visualize results.

In summary, this topic not only provides a deep insight into an important aspect of optics but also equips learners with essential scientific, technical, and programming skills—forming a strong foundation for future research and practical applications.

**4. Implementation Steps**

**Step 1: Build Functions for Spherical Surface Geometry**

Develop the sagitta function: Calculate the lens surface height (z) based on radial distance (r) and parameters such as radius of curvature (R) and conic constant (k).

Build the sagitta derivative function: Compute the surface slope, necessary for determining the surface normal vector.

Support both spherical and aspherical surfaces: Ensure the functions can describe both types of surface geometry.

**Step 2: Create Function to Find the Intersection of Light Rays and the Surface**

Set up equations: Combine the ray’s linear equation with the lens surface equation.

Solve nonlinear equations: Use numerical methods (e.g., scipy.optimize.root\_scalar) to accurately find the intersection coordinates.

**Step 3: Calculate Refraction Angles Using Snell’s Law**

Determine the surface normal vector: Use the sagitta derivative at the intersection point to find the surface normal.

Apply Snell’s Law: Calculate the angle of refraction based on the incident angle, surface normal, and refractive indices of the two media.

Determine the new ray direction: From the refraction angle, compute the new direction vector of the light ray.

**Step 4: Simulate the Light Ray Path Through Two Spherical Surfaces**

First refraction: Compute the intersection and new direction as the light ray enters the lens.

Second refraction: Compute the next intersection and direction as the light exits the lens.

Trace the entire path: Record the complete trajectory of the light ray from start to finish.

**Step 5: Visualize Ray Paths and Lens Surfaces Using Matplotlib**

Plot the lens shape: Use the sagitta function to generate the shapes of the lens surfaces.

Plot the ray path: Clearly illustrate the curved ray trajectory, refraction points, and ray direction at each stage.

Visualization clarity: Ensure the graphs are clear and easy to understand.

**Step 6: Repeat the Simulation with Various Incident Angles and Observe the Focal Point**

Iterative Simulation: Re-run the entire simulation process for multiple light rays with different initial incident angles.

Focal Point Tracking: Monitor and record the position where each refracted ray intersects the optical axis after passing through both surfaces.

Aberration Analysis: Use this data to evaluate spherical aberration and demonstrate the improved performance of aspherical lenses.

**5. Results**

**5.1. Accurate Visualization of Light Ray Propagation Through Two Spherical Surfaces**  
We successfully developed a Python model capable of accurately visualizing the trajectory of a light ray passing through a system with two lens surfaces. The model not only illustrates the geometry of the lens but also clearly shows:

The intersection points between the light ray and each lens surface.

The change in direction of the ray due to refraction at each interface, strictly following Snell’s Law.

A screen shot of a graph

AI-generated content may be incorrect.

*Figure 1: Ray trajectory through the lens system*

The complete ray path from the initial medium, through the lens, and into the exit medium.

A graph with a line

AI-generated content may be incorrect.  
*Figure 2: Trajectory showing the refracted ray intersecting the optical axis at the image point*

These results confirm the correctness of the geometric formulas and algorithms used in modeling the refraction phenomenon.

**5.2. Rays Converge Near a Real Image – Validating the Physical Accuracy of the Model**  
**A graph of a graph

AI-generated content may be incorrect.**

*Figure 3: Trajectories of multiple rays refracted through the lens*  
**A graph of a graph showing a line graph

AI-generated content may be incorrect.**

*Figure 4: Zoomed view near the lens region from Figure 3*  
**A graph with a blue line

AI-generated content may be incorrect.**

*Figure 5: Zoomed view of the focal point region from Figure 3*  
**A screen shot of a computer

AI-generated content may be incorrect.**

*Figure 6: Terminal output of calculated image points*

When simulating multiple rays passing through the two spherical surfaces, we observed that the rays converge toward a specific region along the optical axis, forming a real image of the object. The convergence of the rays into a tight cluster (rather than random dispersion) validates the physical accuracy of the model we developed. This outcome demonstrates that the algorithm successfully simulates fundamental lens behavior consistent with geometric optics principles.

**5.3. The Model Allows Modification of Conic Constant (K), Incident Angle, and Visualization of Focal Variation**

One of the most important achievements of our model is its flexibility in allowing users to dynamically change input parameters and immediately observe their effects. Specifically, the model provides the ability to:

Adjust the conic constant (K) of the lens surfaces, enabling transitions between spherical and various aspherical lens shapes.

Modify the incident angle (or the initial position) of the incoming light rays.

When these parameters are varied, the model displays clear changes in the focal point of the rays. This enables us to conduct quantitative analyses on:

Spherical aberration effects: We observed that for spherical lenses ***(K = 0)***, rays with larger incident angles tend to focus at different points, causing a "blurred" focal region.

A graph with a blue line

AI-generated content may be incorrect.

*Figure 7: Illustration of focal errors when using spherical surfaces* ***(K = 0)***

Effectiveness of aspherical lenses: By tuning the value of K, we demonstrated that rays can be brought to focus more precisely at a single point, significantly reducing spherical aberration and improving image quality. The images shown in section 5.2 were generated after sweeping K from 0 to 10 to find the optimal value. The best result was obtained with **K = 6.46**, where rays from a point object located **d = 10 units** away from the lens (with **R₁ = R₂ = 10, n2 = 1.5**) converge to a focal point at **P₁ = (18.24, 0)**.

**A graph with a blue line

AI-generated content may be incorrect.**

*Figure 8: Illustration of improved focal convergence using an aspherical surface with optimal K*

These results not only validate the flexibility of our model but also lay a foundation for lens design optimization in real-world optical applications.

**6. Discussion**

During the development of this ray tracing simulation through two spherical surfaces, I gained several valuable insights.

**6.1. Challenges Encountered**  
One major challenge was solving the nonlinear equations to find the intersection points between light rays and the lens surfaces—particularly in the case of aspherical surfaces, which require careful domain control to avoid invalid solutions. Additionally, handling numerical errors such as NaN values or negative square roots was essential to ensure the stability and accuracy of the simulation.

**6.2. Academic Significance**  
This project has deepened my understanding of analytical geometry, differential calculus in physical optics, and how to apply Python programming to simulate complex physical phenomena. It has also strengthened my ability to independently plan and execute simulations involving advanced computational techniques.

**6.3. Future Work**  
In the future, I aim to extend this project by computing and optimizing the exact parameters of aspherical surfaces. The goal is to design lenses that minimize optical aberrations effectively, paving the way for practical applications in optics such as cameras, microscopes, and telescopes.\

Link of project: <https://github.com/TanHongPhong/applied-caculus-THP>