AUGUST 2019 M.Sc. in Optical Engineering HABİL ZORLU

REPUBLIC OF TURKEY

GAZİANTEP UNIVERSITY

GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES

A MERIDIONAL RAY TRACING SOFTWARE DEVELOPMENT FOR LENS DESIGN

M.Sc. THESIS

IN

OPTICAL ENGINEERING

BY

HABİL ZORLU

AUGUST 2019

A MERIDIONAL RAY TRACING SOFTWARE DEVELOPMENT FOR LENS DESIGN

M.Sc. Thesis
in
Optical Engineering
Gaziantep University

Supervisor
Prof. Dr. Metin BEDİR

Co-Supervisor
Assoc. Prof. Dr. Ahmet BİNGÜL

by
Habil ZORLU

August 2019

REPUBLIC OF TURKEY
GAZİANTEP UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OPTICAL ENGINEERING

Name of the Thesis	:	A Meridional Ray Tracing Software Development for Lens Design
Name of the Student	:	Habil ZORLU
Exam Date	:	06.08.2019

Approval of the Graduate School of Natural and Applied Sciences.

Prof.Dr. A. Necmeddin YAZICI
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Ahmet BİNGÜL
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Ahmet BİNGÜL
Co-Supervisor

Prof. Dr. Metin BEDİR
Supervisor

Examining Committee Members:	Signature
Prof. Dr. Metin BEDİR	....
Prof.Dr. A. Necmeddin YAZICI	....
Asst. Prof. Dr. Rasim ÖZDEMİR	....

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Habil ZORLU

ABSTRACT

A MERIDIONAL RAY TRACING SOFTWARE DEVELOPMENT FOR LENS DESIGN

ZORLU, Habil
M.Sc. in Optical Engineering
Supervisor: Prof. Dr. Metin BEDİR
Co-Supervisor: Assoc. Prof. Dr. Ahmet BİNGÜL
August 2019
70 pages

In this thesis a simple meridional ray tracing software which can be used in the optical lens design is developed. The first version of this software is called Heysem 1.0 which has both paraxial and exact ray tracing capability. Heysem has a simple Graphical User Interface (GUI) to input the lens radii, thicknesses, number of rays and etc required in the design. It gives some outputs the system layout, a summary table and some basic lens aberration calculations and plots to the user. The outcomes and plots obtained in Heysem are compared with Zemax OpticStudio program. It is found that the ray tracing computations and plots in both softwares are the same.

Key Words: Lens Design, Ray Tracing, Heysem, Paraxial, Exact

ÖZET

MERCEK TASARIMI İÇİN MERİDYONEL IŞIN İZLEME YAZILIMI GELİŞTİRME

ZORLU, Habil
Yüksek Lisans Tezi, Optik Mühendisliği Bölümü
Danışman: Prof. Dr. Metin BEDİR
İkinci Danışman: Assoc. Prof. Dr. Ahmet BİNGÜL
Ağustos 2019
70 sayfa

Bu tezde mercek tasarımı için basit bir yazılım geliştirildi. Yazılımın ilk versiyon olarak adı Heysem 1.0 belirlendi. Bu proğram hem paraksiyal hemde gerçek ışın çizimi yapabilme yeteneğine sahiptir. Heysem basit kullanıcı arayüzüne sahiptir. Mercek tasarımı için giriş değerleri mercek yarı çapı, kalınlık, ışın sayısı vb gereklidir. Proğram çıktı olarak mercek çizimi, özet tablo bazı temel mercek kusurları hesaplamalarını ve çizimlerini kullanıcıya verir. Heysem proğramında elde edilen grafik ve sonuçlar Zemax proğramı ile karşılştırıldı. Işın çizim hesaplamaları ve grafikler her iki proğramdada aynı bulundu.

Anahtar Kelimeler: Mercek Tasarımı, Işın İzleme, Heysem, Paraksiyal, Gerçek

I dedicate this work to my family

ACKNOWLEDGEMENTS

I would like to express deeply grateful to my supervisor Prof. Dr. Metin BEDİR and co-supervisor Assoc. Prof. Dr. Ahmet BİNGÜL for their guidance, patience and support. I want to thank Uygar ŞAŞMAZ to support. I am deeply grateful to my family. I have felt their support behind me all the time of my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

y	Height of the ray
u	Ray slope
f	Focal length
t	Thickness
d	Edge thickness of the lens
R	Radius of curvature
n	Refractive index
V	Vertex of the lens
N	Nodal point of the lens
H	Principal point of lens
h	Distance vertex to principal point
D	Clear aperture

LIST OF ABBREVIATIONS

LSA	Longitudinal spherical aberration
TSA	Transverse aberration
EnP	Entrance pupil
ExP	Exit pupil
EFFL	Effective focal length
BFL	Back focal length
FFL	Front focal length
OBJ	Object
IMG	Image
A.S	Aperture stop
NA	Numerical Aperture
F/#	F number
OPD	Optical path difference
GUI	Graphical user interface

CHAPTER 1

INTRODUCTION

Before a lens can be produced it must be designed by a designer. Radius of curvature of surfaces, the thickness, index of lens element, the diameter of the various lens elements and optical lens material properties must be determined.

Ray tracing is a primary method used by optical engineers to determine optical system analysis and performance. It provides the calculating the angle of refraction and the height of light ray at the each surface in the optical system. Location, size, and orientation of the image formed by the lens can be defined with a ray tracing from the object plane to image plane sequentially. To implement ray tracing and physical optics in computer, there are many lens design programs such as Zemax OpticStudio [1] or Code V [2].

The main purpose of this thesis is to develop a ray tracing program, named Heysem[1] , for using in both academic and education purposes. Heysem uses two main ray tracing algorithms, which are paraxial and exact ray tracing. In Paraxial ray tracing, two different methods are used. They are known as y-u and y-nu trace for thin lenses and thick lenses, respectively. Heysem has a user friendly Graphical User Interface built in MATLAB [3]. The program is also capable of ploting spherical aberrations and optical path differences. It also provides a summary report called prescription data of the lens system to the user. For a comparison, Zemax is used in order to verify the outputs of Heysem.

The chapter organization of the thesis is as follows. In Chapter 2, general geometrical optics are given. The rays tracing algorithms are presented in Chapter 3. A user guide of Heysem is introduced in Chapter 4. Finally, a summary and a conclusion of the thesis is given in Chapter 5. The MATLAB codes of developed program can be found in Appendix.

[1] Ibn al-Haytham (born in Basra) mathematician and astronomer who made significant contributions to the principles of optics and the use of scientific experiments. The most important work is Kitab al-manazir (Optics) [4].

CHAPTER 2

BASIC GEOMETRIC OPTICS

2.1 Introduction

Design and analysis of lens systems uses numerical calculations based upon geometrical optics. Geometrical optics is based upon the fundamental assumption that light propagates along rays. Rays in a homogeneous medium follow straight lines. It does not account for certain optical effects such as diffraction and interference. The geometrical optical model is appropriate to define the properties of image formation by a lens. Ray tracing is the basic tool used in optical design. The geometrical ray-based model provides determination of image location and aberrations, and enables calculation of the pupil location and size. The last section of this chapter provides basic information about optical materials [14].

2.1.1 Light Propagation and Index of Refraction

Light is defined as electromagnetic radiation within a certain region of electromagnetic spectrum with wavelengths between 400 nm and 700 nm which is visible to human eye. The laws of optics and methods of optical design usually deal with visible region. While light waves emanate from a point source in all directions take a spherical wavefront form in the isotropic medium, radius of spherical shape is equal to distance from the source. If we trace a path of a hypothetical point on the wavefront surface as it moves through the space, we see that the point progresses as a straight line. The path of a point on the wavefront is called *light ray* [5].

As light rays traveling in a vacuum that has a velocity approximately $c = 3 \times 10^8$ m/s. But in the material medium, velocity and is less than the value in a vacuum [7]. The ratio of the vacuum velocity divided by the velocity of medium is called index of refraction material medium that is denoted by the letter n [5]:

$$n = \frac{c}{v} = \frac{\text{velocity in vacuum}}{\text{velocity in medium}} = \frac{\text{wavelength in vacuum}}{\text{wavelength in medium}} \tag{2.1}$$

2.1.2 The Law of Refraction

When the light ray traveling through the homogeneous medium, it encounters an another medium boundary, as some of light rays is reflected, remaining rays is transmitted into second medium. When the transmitted rays cross into the second medium, it changes both its direction and it refracts. The law of refraction, also known as Snell's law that relates the relationship between the sines of the incidence angle I and refraction angle I' measured from with respect to the normal to the surface and the indices of refraction of the two mediums [5]. As it is seen from Figure 2.1. This general relationship has the following mathematical form:

$$n_1 \sin I_1 = n_2 \sin I_2 \tag{2.2}$$

where I_1 and I_2 are, respectively, the incident angle and the refracted angle of the ray with respect to the normal to the surface, while n_1 and n_2 are the refractive indices of the two different material medium.

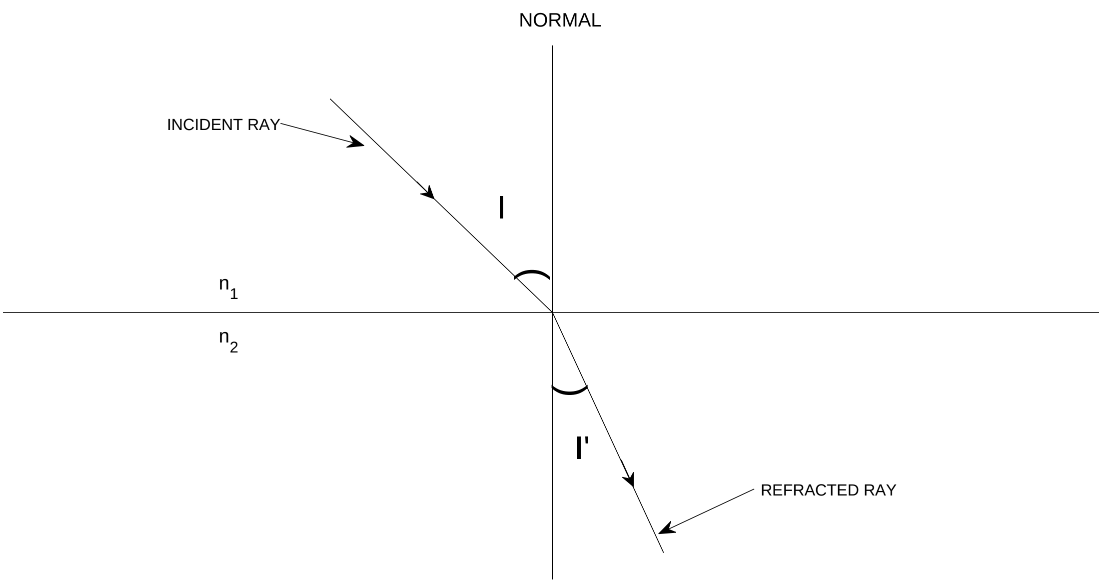

Figure 2.1 Refraction of the light ray at an interface between two different optical media.

2.1.3 Sign Conventions

Sign convention is very important to facilitate the ray tracing throughout in the optical system and it must be clearly defined for distances and angles. A single arrowhead is used to demonstrate whether distances and angles direction are positive or negative in the optical system.

- Light rays travel left to right and all refractive indices are positive [6].
- Distances measured to left of reference point are negative, to the right are positive.
- While focal length of the converging lens is positive, it is negative for the negative lens [7].
- While heights above the axis are positive, heights below the optical axis are negative [7].
- Surface radius is positive means that the center of curvature lies to right of the surface and radius is negative if the center of the curvature lies to left of the surface [6].
- Angles that are measured counterclockwise from a reference are positive; measured clockwise from a reference are negative [6].

2.1.4 Sag of Spherical Surfaces

In the paraxial approximation light rays refracts on the plane surface. Actually, light rays refracts on the spherical surface. Therefore, there is some separation vertex plane and spherical surface. The phenomenon known as sag of surface. The amount of separation increases if the light height move away from the paraxial region. Also the relation between the ray height on the spherical surface with radius and sag is important. The separation between the plane surface and lens surface can be derived from geometry of Figure 2.2 [6, 10].

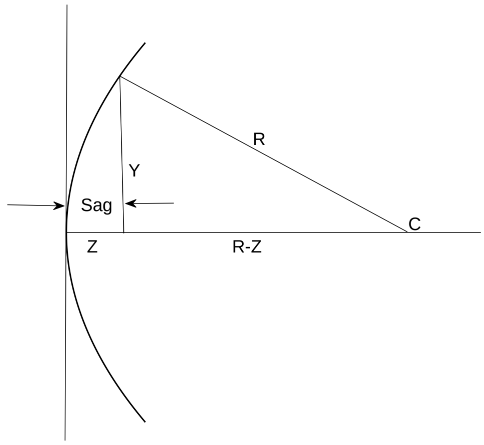

Figure 2.2 Plot of the spherical surface with the sag Z.

$$
\begin{aligned}
R^2 &= Y^2 + (R - Z)^2 \\
Z &= R - \sqrt{R^2 - Y^2}
\end{aligned}
\tag{2.3}
$$

where R is the radius of curvature, Y is the height of ray Z is the sag of the surface increases with the height of the light on the surface.

2.2 Paraxial Optics and Calculations

Paraxial optics is used to determine the location and size of images and pupils in the optical system [14]. It sometimes referred to first-order or Gaussian optic is known as the optics of perfect optical systems. Thus, there are not any aberrations. we will consider the properties of optical systems in the region close to the optical axis, usually known as the paraxial region that is a infinitesimal thin region [5]. As it is seen in the Figure 2.8.

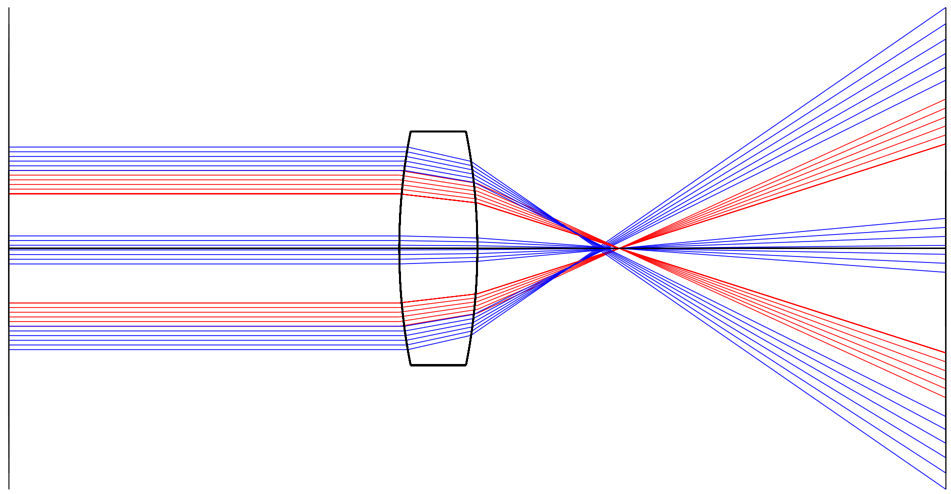

Figure 2.3 Paraxial region of the optical system is the thin region about the optical axis. Red light rays are defined paraxial rays that intersect the optical axis paraxial focus and blue light rays are defined as exact rays that intersect the optical axis at different points. But exact rays intersect the optical axis at the paraxial focus as rays close the optical axis.

Incident and refraction angles of light rays may be set equal their sine and tangent [5]. Paraxial equations are linear with respect to ray angles and heights. The sag of the surface is ignored in the this region.

2.2.1 Lenses and Lens Types

Lenses can be divided into two categories. One of them is positive lens that refracts rays convergently coming through on it and has positive focal length. An other type of lens is negative lens that refract rays divergently and has negative focal length. The optical axis of a lens is its rotational symmetric axis. When light rays parallel to the optical axis pass through a positive lens, they are refracted and intersect the optical axis at certain point. This point is known as focal points of this positive lens. For the negative lens, refracted light rays will not intersect the optical axis since this rays are diverted. But back extension of refracted rays will intersect the optical axis at a certain point that point is focal point for negative lens. There are two types of lenses that are thin lens and thick Lens. If the center thickness of a lens is so small compared to the radii of curvature of the lens surfaces, the lens can be considered as thin lens. The focal length of a thin lens can be write f [16]. The focal length of the thin lens is defined as the image distance for an object at infinity, giving

$$\frac{1}{f} = (n-1)\left(\frac{1}{R_1} - \frac{1}{R_2}\right) \tag{2.4}$$

where n is the refractive index of the lens material and R_1 *and* R_2 are the radii of the two surface curvatures of the lens and thin lens equation can be write in terms of focal length, object distance s and image distance s' [8].

$$\frac{1}{f} = \frac{1}{s} + \frac{1}{s'} \tag{2.5}$$

The lateral magnification (m) of an optical system is either given by ratio of image size to object size or image distance to object distance.

$$m = \frac{s'}{s} \tag{2.6}$$

When the center thickness of a lens is not so smaller than the two surface radii of the lens, the lens is called thick lens whose focal length f can be written

$$\frac{1}{f} = (n-1)\left(\frac{1}{R_1} - \frac{1}{R_2} + \frac{(n-1)d}{nR_1R_2}\right) \tag{2.7}$$

2.2.2 Cardinal Points of the Lenses

For the lens, there are tree types of cardinal points which are focal points, principal points and nodal points. If the light rays come from left to right or right to left from infinitely distant and parallel to optical axis, this rays undergo refraction on the lens and focuses a points on the optical axis. This points are called as second focal point, F_2, at the right side of the lens and first focal point, F_1, at the left side of the lens, respectively. If a bundle of rays entering the lens and after emerging from the lens are extended until they come across each other a point, this points on the lens look likes a planes according to paraxial approximation that known as principal planes. The intersection of this planes with the axis are the principal points (H_1 *and* H_2). Any ray directed the toward to first nodal point, N_1, emerge from the lens system parallel to incident ray and appears to come from the second nodal point N_2. For the thin lenses, the principal points and nodal points together with its planes are cojugate, unlike for the focal points.

The position of the all six cardinal points are showed in Figure 2.4. When the both side of the lens or optical system ia bounded by air, the nodal points are coincide with principal planes.

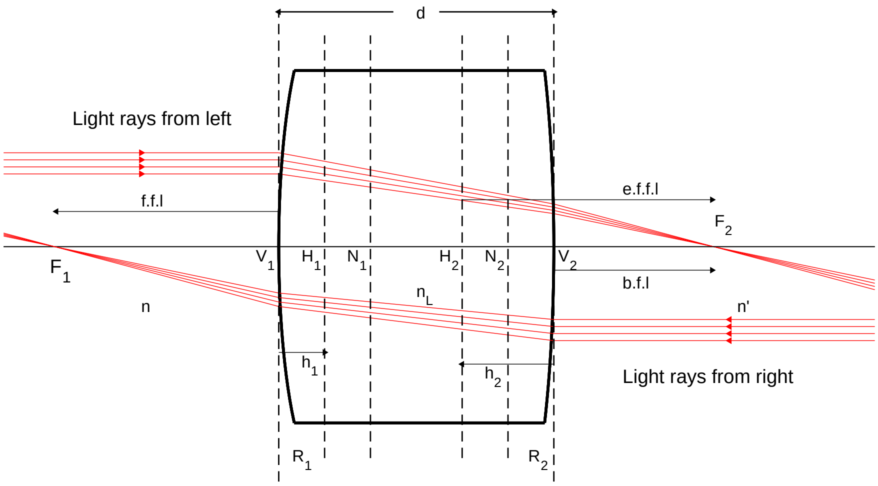

Figure 2.4 Light propagation and cardinal points of the lens.

where R_1 and R_2 radius of curvature, d center thickness of the lens, h_1 distance between V_1 *to* H_1, h_2 distance between H_2 *to* V_2, n_L refractive index of the lens. The effective focal length (effl) of the optical system is the distance between principal point to focal point thus (distance H_2 *to* F_2 *or* H_1 *to* F_1). The back focal length (bfl) is distance between last surface of the lens vertex (V_2) to second principal point. The front focal length (ffl) is the distance between first principal point to first surface vertex (V_1).

We can calculate this terms by using ray tracing that will be showed in the next chapter [5, 8].

2.3 Stops, Pupils and Windows

Optical systems have an aperture stop, field stop, entrance pupil, and exit pupil [16]. Light rays emerge from the object plane will not reach the image plane since some optical structure constrict to pass through the optical system. Thus, the only some usable light at the appropriate angles enters the system to reach the image location. These optical structure known as stops. There are two types of stops that are aperture stop that limits amount of light entering the system. It may be located in front or at intermediate the optical system as seen from the Figure 2.5 [6]. It constrict the some light rays. Another type of stop is field stop which determines the field of view or how much of the object can be seen through the optical system. It can be located at the object plane, image plane or in the middle of the optical system [6].

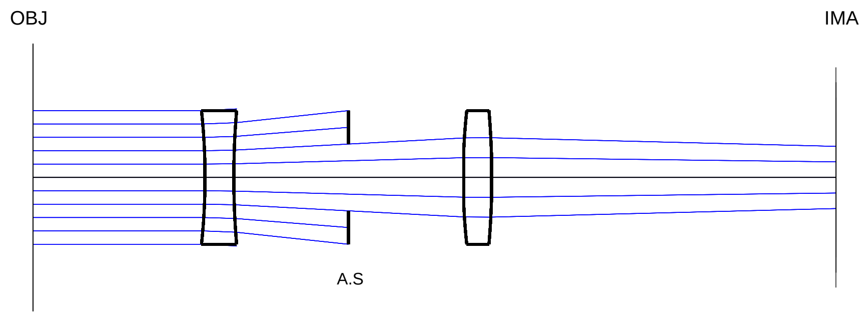

Figure 2.5 As bundle of rays propagates lens system from object plane to image plane, the aperture stop truncates the bundle of the rays at the upper and lower rim.

The image of the aperture stop as seen from the object space that is the left of the first surface of a system is known as entrance pupil (EnP) of the system. Similarly, image of the aperture stop as seen from the image space that is right of the last optical element is known as exit pupil (ExP) of the system as seen from the figure 2.6. If there are no lenses between object and aperture stop, in this case aperture stop is the entrance pupil. An other condition, if there are no lenses between aperture stop and image plane, then aperture stop serve as exit pupil [7, 16]. Aperture stop may be in the middle of the two or more lenses. We can find easily location of the entrance pupil and exit pupil. We assumed that an object is located at the edge of the aperture stop and by tracing some rays forward and rearward from periphery of the object can be found its images that are the entrance pupil and exit pupil of the system as it is seen from the object space and image space respectively [16]. It is seen from the Figure 2.6.

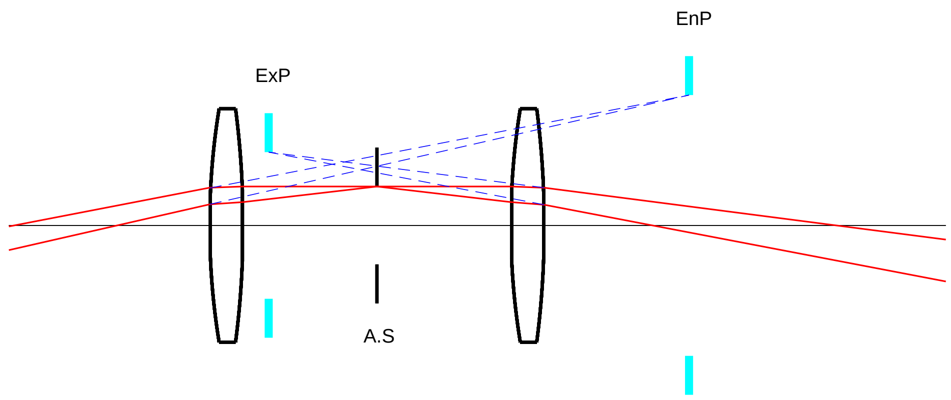

Figure 2.6 Aperture stop, entrance pupil and exit pupil for a two-lens system.

There are two important rays in the meridional plane which are marginal ray and chief ray are also known as meridional rays. Marginal ray or axial ray starts at the axial object point and proceeds to edge of the entrance pupil and determine the image location and image size. It also propagates the edge of the aperture stop and edge of the exit pupil [11]. As it is seen from the Figure 2.7

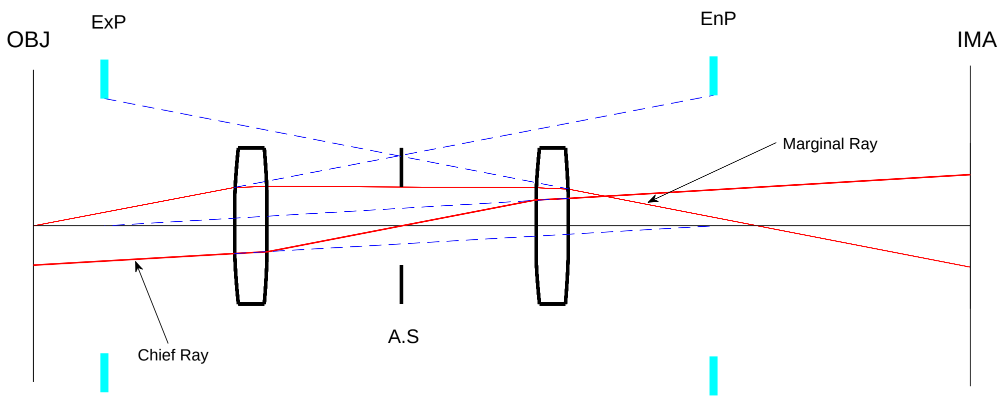

Figure 2.7 Location and width of entrance pupil and exit pupil in the optical system.

Another ray is chief ray or oblique ray that emanate from the edge of the object and goes the center of te entrance pupil and also define image height and pupil location. It propagates the center of the aperture stop and center of the exit pupil. Pupil location also can be determine tracing a chief ray center of the aperture stop through left and right side of the optical system [5, 11]. The ray intersection with the axis gives pupil location as shown in Figure 2.8.

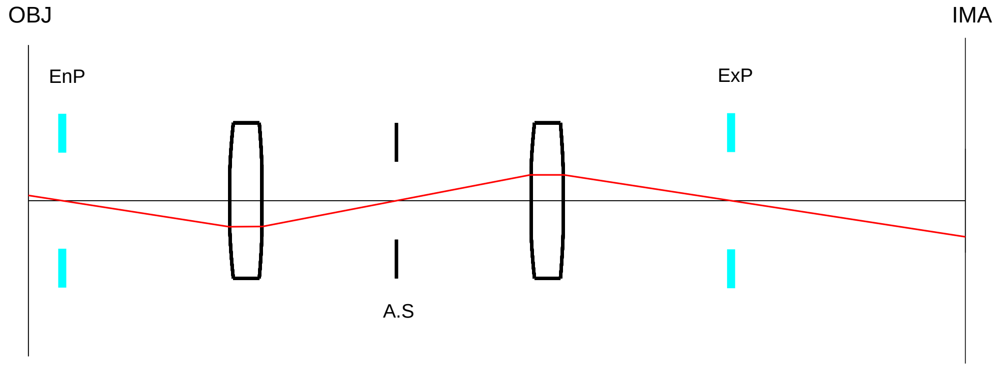

Figure 2.8 The location of the pupils of the optical system to trace a chief rays from middle of the aperture stop throught the its left and right sides optical system can be determined.

A chief ray is traced both direction from the aperture stop center. The ray appears to comes from center of the entrance pupil through the object space. It appears to emanate from center of the exit pupil through the image space. If the slope of the chief ray increase the edge of the first lens, the first lens is the field stop. Because the field stop is in the object space, it also known as entrance window. Extension of the chief ray from the first lens intersect the optical axis at the center of the entrance pupil. The angle between extension of two chief rays is called angular field of view of the system. The field of view is also defined as the maximum angular size of the object as seen from the entrance pupil. As it is seen from the Figure 2.9. The field stop image must be occurred through second lens. The image of the field stop is called as exit window of the system. Briefly, images of the field stop as seen from object and image space gives entrance window and exit window, respectively [7, 11].

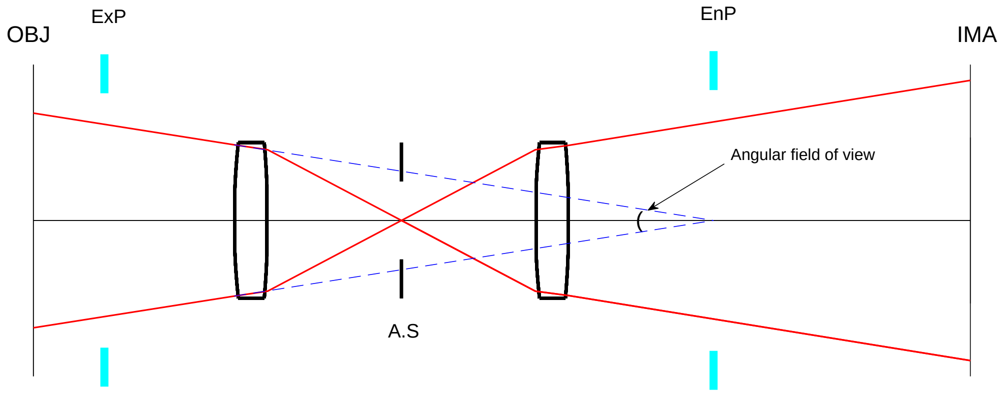

Figure 2.9 Angular field of view of the system

Figure 2.9 shows that first lens serve as both field stop and entrance window and its images gives exit window. The entrance and exit pupil, the aperture stop have important role both collecting light and decreasing the certain aberrations.

2.3.1 F-Number and Numerical Aperture

F/# is defined as the ratio of the effective focal length to clear aperture or diameter of the entrance pupil [5].

$$F/\# = \frac{\text{effective focal length}}{\text{clear aperture}} = \frac{f}{D} \tag{2.8}$$

where F/# is a single symbol. The f- number is also known as relative aperture. For example, a lens with 30 mm aperture and 60 mm focal length has a f-number 2, which is usually denoted by f/2 [15]. The numerical aperture (usually abbreviated as NA) is defined as index of refraction of the medium times by the sine of the largest entrance ray angle with respect to the optical axis.

$$NA = n \sin\ u \tag{2.9}$$

Numerical aperture and f-number are two methods of describing same characteristic of a system. While numerical aperture is used for systems which work at finite conjugate such as microscope objective, f-number is conveniently applied to systems for use distant object such as camera lens [5].

2.4 Aberrations

Any exact optical system includes various aberrations. The main purpose of optical design is to minimize these aberrations. There are five important aberrations such as spherical aberration, coma, astigmatism, field curvature, and image distortion. These five aberrations are monochromatic. We will also describe chromatic aberrations and optical path difference [15].

2.4.1 Spherical Aberration

Spherical aberration is a optical defect in the optical systems. It occurs that when all incoming light rays on the spherical surface of a lens, these rays intersect the optical axis different points. Because of this, spherical aberration affect quality of the images. We suppose that an exact ray coming from an object at infinity, while some of these rays intersect the optical axis near the lens, some of another rays intersect the optical axis very near the paraxial focus position. Once the rays height increases from the optical axis, rays intersection points with the axis move away the paraxial focus. Figure 2.10 shows that paraxial marginal rays which defined as red color intersect the axis paraxial focus. Exact rays which shown as blue rays intersect the axis different points [15].

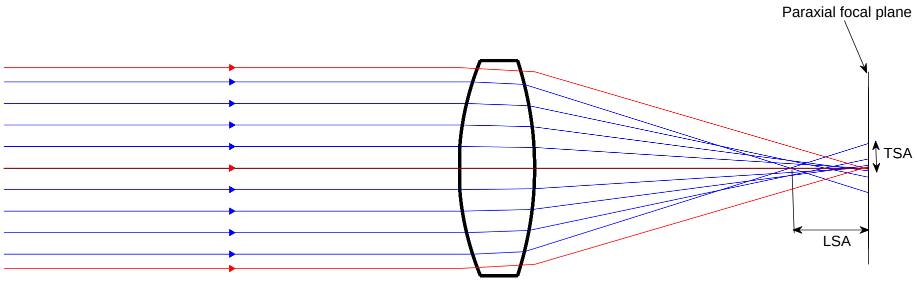

Figure 2.10 A simple converging lens with spherical aberration. The distance between the points where paraxial marginal rays (red rays) and exact rays (blue rays) cross the axis is defined as longitudinal spherical aberration (abbreviated LSA). The distance between paraxial focus point and exact ray in the paraxial focus plane is transverse spherical aberration (abbreviated TSA). Note that as the exact rays approach to the optical axis to intersect the optical axis near the paraxial focus. However, these rays away from the optical axis intersect the axis near the lens.

The distances between points where the paraxial rays and exact rays intersect the axis is called longitutinal spherical aberration is abbreviated LSA.

$$LSA = T - t \tag{2.10}$$

where T and t is the distance between last surface of the lens and their corresponding intersection point. Another type of spherical aberration transverse aberration can be defined as distance between the points where the rays cross the paraxial plane. Spherical aberrations are generally represented graphically. Longitudinal spherical aberration (LSA) is plotted against ray height at the lens. As shown in Figure 2.11.

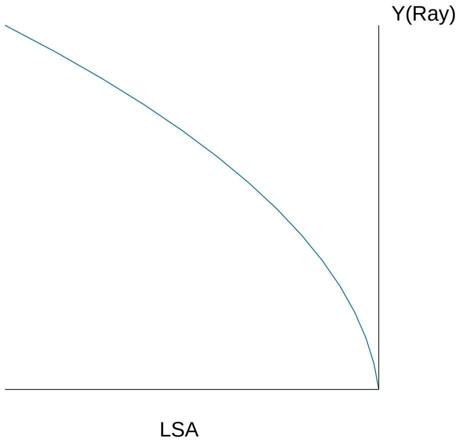

Figure 2.11 Graphical representation of the longitutinal spherical aberration (LSA) that is plotted against ray height of the last surface of lens, Y(Ray), at the lens.

Transverse spherical aberration can be calculated as to trace both exact rays and paraxial rays which emerge from at the same points and paraxial image plane is shifted toward left side from its initial point that may defined as shifted reference plane. These rays intersect the reference plane difference points. The differences between relative paraxial ray height and exact ray height gives transverse aberration [5, 7]. Spherical aberrations are affected by object position, width of aperture and lens shape but it can be eliminate by decreasing the aperture size and changing lens shape.

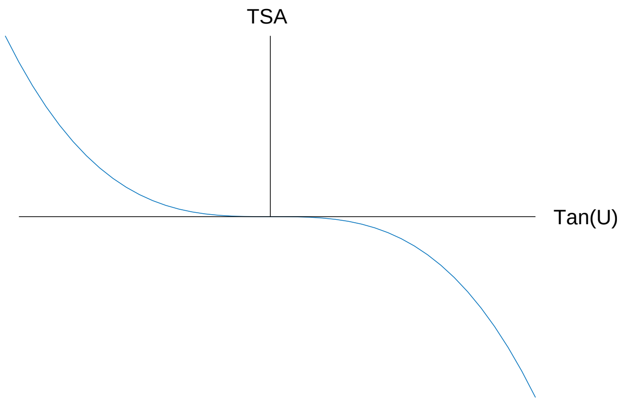

Figure 2.12 Plot of transverse aberration (TSA) versus final ray slope Tan(U).

2.4.2 Coma

When the light rays that parallel the optical axis and incident on the spherical lenses focuses different points on the optical axis, spherical aberration occurs. A bundle of ray about the chief ray with an angle will focus paraxial image point. But marginal rays passing through the edge portions of the lens do not focus on the chief ray. This kind of phenomenon is called coma aberration [7]. The upper and lower rim rays intersect the plane above the chief ray at same plane. The difference between intersection two rays point and chief ray is called tangential coma [15]. As it is seen from the Figure 2.14.

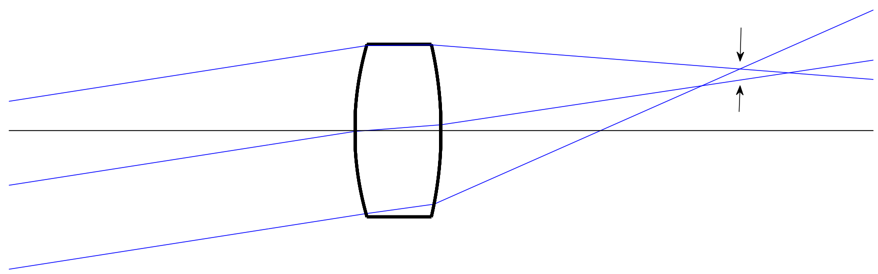

Figure 2.13 If parallel rays with an angle to the optical axis are focuses, coma appears. The distance head of arrows shows tangential coma.

The magnitude of coma aberration depends on the shape of the lens element, an aperture position and size and incident angle with axis. If incident angle of the rays and the lens aperture size increase, coma aberration increases. As the number of the light rays increases, the coma aberration looks like comet-shaped flare in the image plane. It can be seen spot diagram. Coma aberration affects sharpness of the image [16].

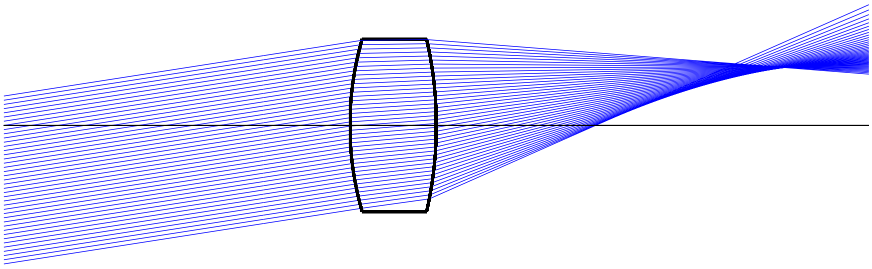

Figure 2.14 When the light rays away from the both side of the chief ray causes image blur.

2.4.3 Optical Path Difference (Wavefront Aberration)

Light waves radiating from the point source take a spherical form. After refraction in the optical system, light waves converging to form perfect image. In the Figure 2.15 represents spherical wavefront from the object point and aberrated wavefront emerging from the optical system. Spherical paraxial wavefront is denoted by dashed line that produces an image at the paraxial image. Exact wavefronts are denoted by solid line that represent actual solution of the optical system. Ray is the path of a point on the paraxial wavefront is defined as paraxial ray (red line) and path of a point on the exact wavefront is defined as exact ray (blue line). That the rays are also normal to the relative wavefront. As it seen in the Figure 2.15, paraxial ray and exact ray do not intersect the optical axis at the same point. Difference between paraxial and exact rays on the optical axis defined as longitudinal aberration and difference between these rays in the image plane defined as transverse aberration. These are ray aberrations. Alternatively, OPD aberration can be described in terms of the deviation of the exact wavefront from the paraxial wavefront at various distance from the optical axis known as optical path difference. Notice that if the rays from both wavefront near the optical axis, the rays reaches the same image point [5, 8].

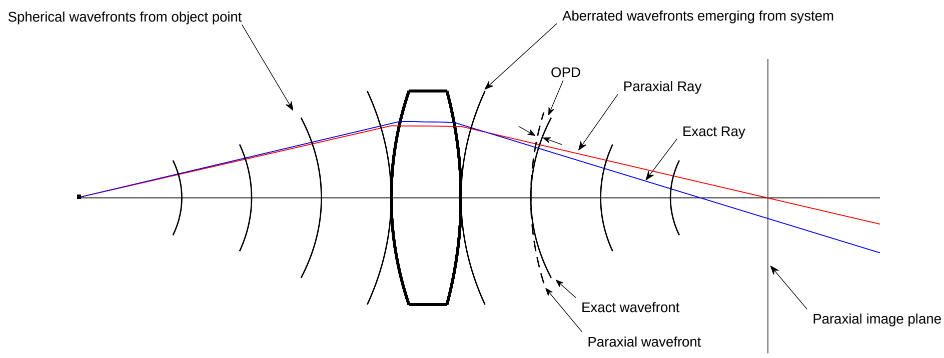

Figure 2.15 Optical path differnce OPD is the difference between exact wavefront and paraxial wavefront at various distance from the optical axis.

Optical path difference can be calculated according to fermat principles states that path is between two given points taken by light rays in the least amount of time. OPD can be calculated as either difference between the marginal ray optical path length and axial optical path length through the optical system or difference optical path lengths between paraxial ray path length and exact ray that emerges from same object point [6, 8].

$$OPD = (\text{Path along the reference ray}) - (\text{Path along the ray}) \tag{2.11}$$

where path along the reference ray is path along the paraxial ray, path along the ray is path along the exact ray [14].

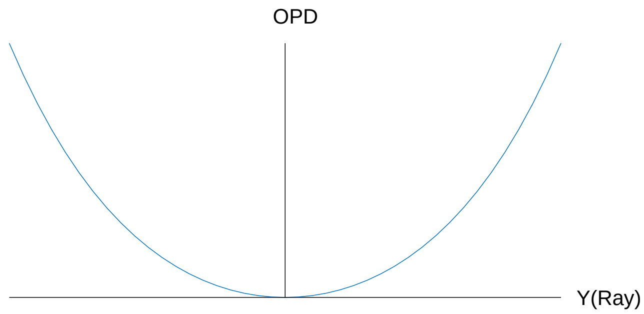

Figure 2.16 Graphical representation of the optical path difference aberration (OPD) that is plotted against ray height of the last surface of lens, Y(Ray), at the lens.

2.4.4 Astigmatism and Field Curvature

Figure 2.17 illustrate the astigmatic images of an off-axis object point. The image of an point source is formed by tangential fan of rays in the tangential plane will be a line image that is known as tangential focal line. This line perpendicular the tangential plane and lies sagittal plane. On the other hand, image is formed by sagittal fan of rays in the sagittal plane will be a line that is called sagittal focal line. This line lies on the tangential plane and perpendicular to sagittal plane. Astigmatism occurs when the tangential line and sagittal line do not coincide [5].

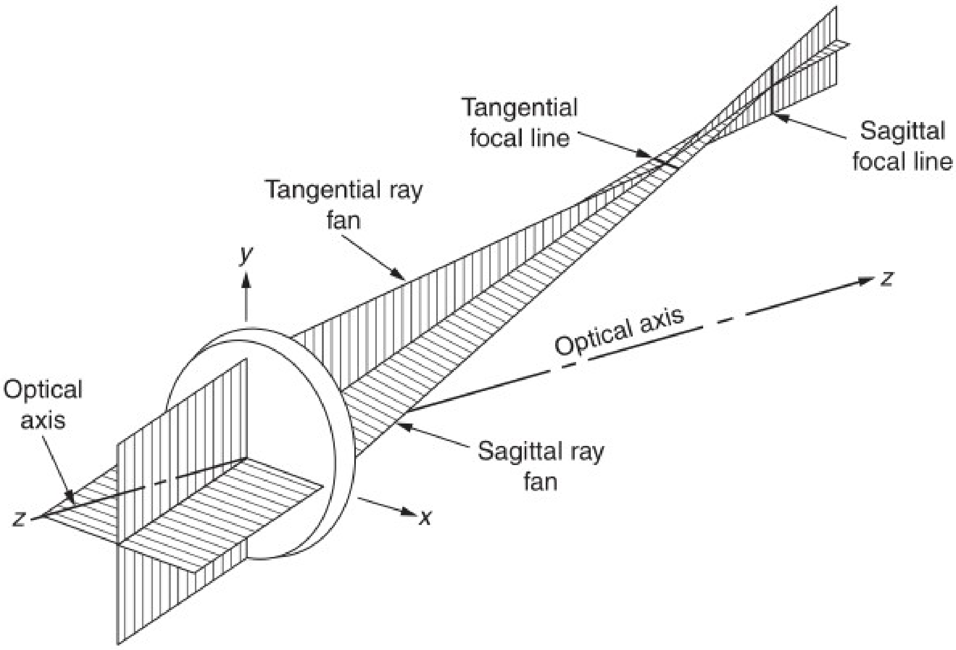

Figure 2.17 If the tangential line and sagittal line do not coincide, astigmatism occurs. Magnitude of the astigmatism changes with respect to the distance between the this lines [5, 16].

Every lens has a basic curved image surface known as Petzval surface (as shown in Figure 2.18), which is a function of the refraction index of the lens element and surface curvature. If the lens has no astigmatism, sagittal and tangential images surfaces coincide with each other and lie on petzval surface. For a simple thin lens, the longitudinal distance between the Petzval surface and the ideal planar image surface is given by $h^2/(2nf)$, where h is the image height, n is the index of refraction of the lens and f is focal length of the lens. If we assumed that light rays comes from an object have five different angles to the optical axis, the light rays with different angles will focuses different points from the ideal image plane. As it is seen in the Figure 2.18, longitudinal position of the focused points moves toward the lens while the field angle increases. This phenomenon is known as field curvature [16].

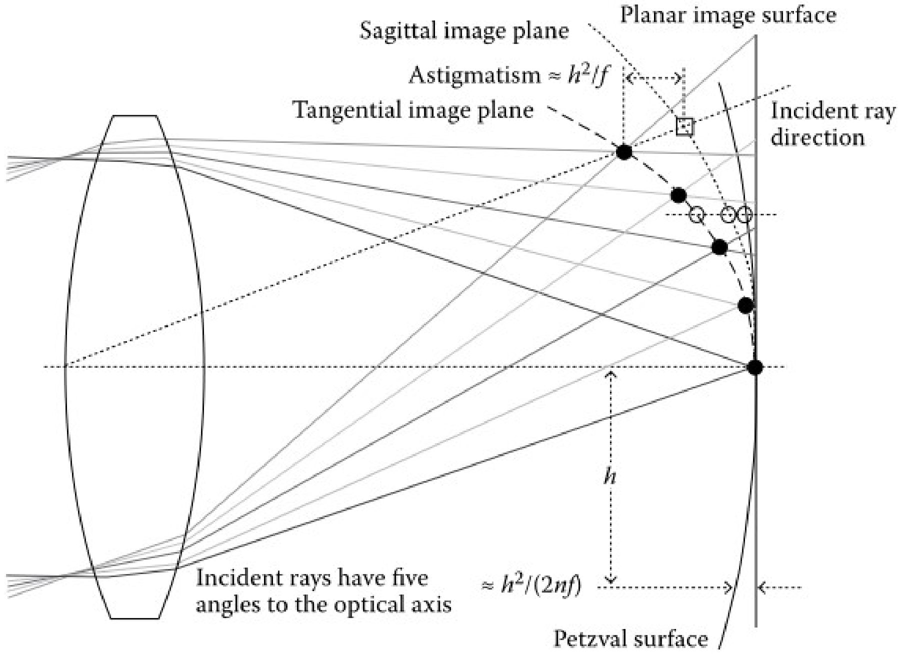

Figure 2.18 The incident rays have five different angles. Tangential image plane is curved more than sagittal image plane [16].

2.4.5 Distortion

Distortion aberration is the easiest to visualize. There are two types of distortion barrel distortion and pincushion distortion. Although object points are imaged as points, distortion occurs as variation in the lateral magnification. If the magnification increases with the from axis, the image appears as pincushion distortion. The image is stretched by its corners. If the magnification decreases with distance from the axis, the images appears as barrel distortion. The image with barrel distortion is compressed at its corners. While the Figure 2.19 shows the pincushion distortion, the Figure 2.20 shows barrel distortion [6, 8].

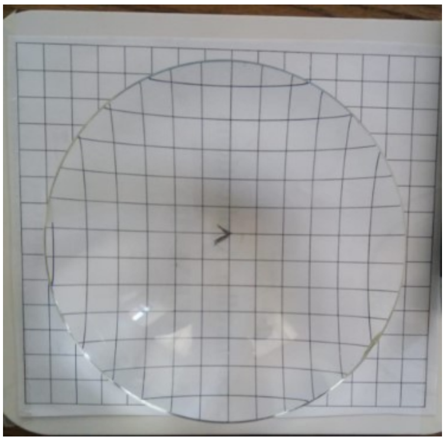

Figure 2.19 Pincushion distortion aberration

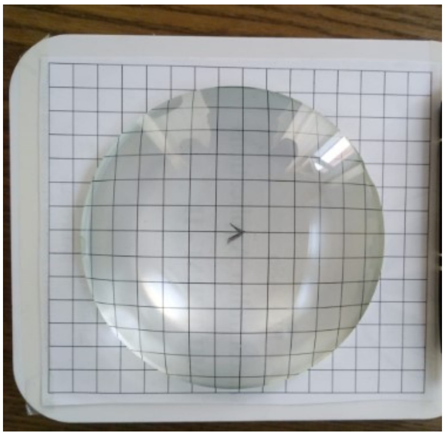

Figure 2.20 Barrel distortion aberrration

2.4.6 Chromatic Aberration

A lens will not focus different colors at the same place on the optical axis since focal length depends on refractive index of the material. Because the index of refraction varies as a function of a wavelength of the light, various colors in the light have different velocities within media. The index of refraction for blue light higher than that of red light since wavelength of the blue ray shorter than red light wavelength. In such a way that the blue light rays focus on the axis nearer the lens than the red light rays as seen from the Figure 2.22 [6, 5].

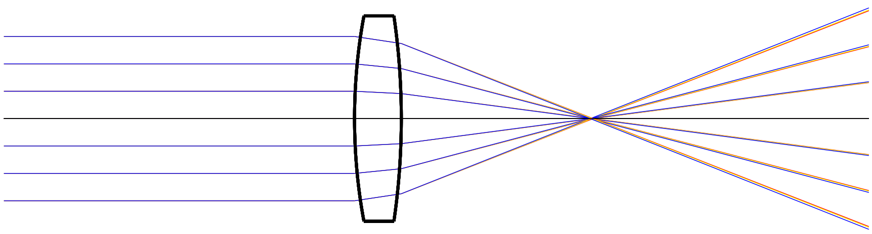

Figure 2.21 Light rays with different colors will not intersect the optical axis t the same point since they have different refractive indices.

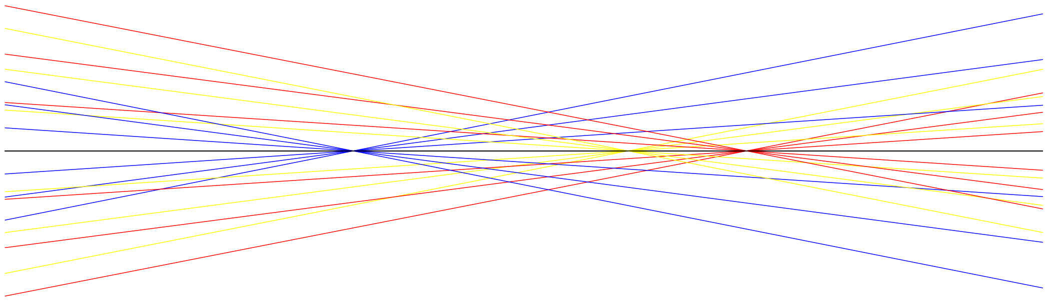

Figure 2.22 The distance along the axis between the blue rays focus and red rays focus is known as axial longitudinal chromatic aberration.

2.5 Optical Materials

The most common lens materials is optical glasses. However, crystals and plastics are can be used. The optical materials must have some properties. It should be able to accept a smoot polish, have homogeneous index of refraction, be chemically and mechanically stable, be free of undesirable artifacts [5].

2.5.1 Abbe Number

The Abbe number is a quantitative measure of the average slope of the dispersion curve. The Abbe number, sometimes called the glass factor [6]. Abbe number, V-number is defined as

$$V = \frac{n_d - 1}{n_F - n_C} \tag{2.12}$$

where n_d, n_F, n_C are the indices of the refraction for the helium d line 0.5876 μm, the hydrogen F line 0.4861 μm, and the hydrogen C line 0.6563 μm, respectively [5]. If the difference between refractive index at the F wavelength and the C wavelength is a small value, Abbe number is large and small dispersion. On the other hand, large dispersion glass has a low Abbe number. V-value or V number is greater than 55 glasses are classified as crown glass, V-value is less than 50 are called flint glass 1.55 μm [6].

2.5.2 Optical Glasses

Optical glasses are very useful material in the visual and near-infrared spectral region. They are easily fabricated, stable homogeneous and clear [5]. Optical glasses are classified as crown glasses and flint glasses. Crown glasses have low index of refraction that is below 1.6 and low dispersion, (high Abbe number, V-value of 55 or more). A common Schott glass, N-BK7, is a crown glass used in precision lenses (n_d = 1.517, V-value=64.7). Flint glasses have high index of refraction (n_d = 1.6) and high dispersion(low Abbe number), V-value less than 50 [6].

2.5.3 Optical Plastics

Plastic lens have become very popular in recent years, especially used for eye glasses. The advantages of plastic lenses are its low cost of raw material, easy and economic for the manufacture, high impact resistance and aspherical surfaces can be molded easily. Disadvantages of the plastic lenses are low heat resistance, high thermal expansion, high temperature coefficient of expansion, surfaces are less durable than glass [6, 5, 10]. The properties of the some used frequently optical materials are given in the Table 2.1.

Table 2.1 Properties of frequently used optical materials [10]

Properties	Acrylic (PMMA)	Polycarbonate (PC)	polystyrene (PS)
Refractive index			
N_F(486.1 nm)	1.497	1.599	1.604
N_D(587.6 nm)	1.491	1.585	1.590
N_C(656.3 nm)	1.489	1.579	1.584
Abbe value	57.2	34.0	30.8
Transmission(%)	92	85-91	87-92
Key advantages	Scratch resistance Chemical resistance High abbe Low dispersion	Impact strength Temperature resistance	Lowest cost Clarity

2.6 Doublet Achromatic Design

The doublet achromatic is design to eliminate chromatic aberrations for the singlet lenses. The achromatic doublet consist of convex lens element with high dispersion and concave lens element with low dispersion. Focal length and power of the lenses are the different in the doublet design. But compound lens has a net focal length Which provide reduction of the dispersion significantly. The general shape of the achromatic doublet is given in the Figure 2.23. The power of the convex and concave lenses for the yellow center of the visible spectrum demonstrated by the Fraunhofer wavelength $\lambda_D = 586.7$ nm [6].

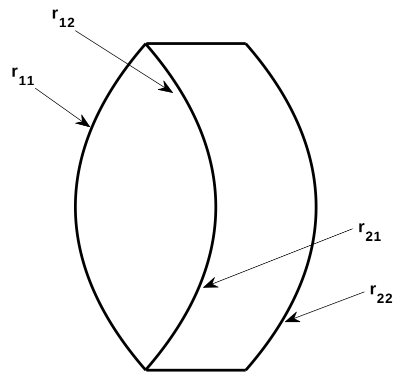

Figure 2.23 Achromatic doublet consist of positive crown glass equiconvex lens and negative flint glass lens. The radii of curvature of lenses are demonstrated.

$$\begin{aligned} P_{1D} &= (n_{1D}-1)+\left(\frac{1}{r_{11}}-\frac{1}{r_{12}}\right)=(n_{1D}-1)K_1 \\ P_{2D} &= (n_{2D}-1)+\left(\frac{1}{r_{21}}-\frac{1}{r_{22}}\right)=(n_{2D}-1)K_2 \end{aligned} \tag{2.13}$$

where the radius of curvatures showed in Figure 2.23. P_{1D} and P_{2D} are the power of the lenses. n_D refers to index of refraction each glass for the D fraunhofer line. K_1 and K_2 are abbreviation for the curvatures.[8].

$$\frac{1}{f}=\frac{1}{f_1}+\frac{1}{f_2}-\frac{L}{f_1 f_2} \tag{2.14}$$

f is the thin-lens doublet with lens separation $L = 0$. The power of the doublet is $P = 1/f$ and also defined as additive of two lens powers [8].

$$P = P_1 + P_2 \tag{2.15}$$

Dispersive constant V is defined as reciprocal of the dispersive power and given by

$$V = \frac{n_D - 1}{n_F - n_C} \tag{2.16}$$

where V is the abbe number and n_D, n_F, n_C are the indices of the refraction for the helium d line, the hydrogen F line, and the hydrogen C line, respectively. The power of the each elements can be expressed in terms of the desired power P_D of the combination:

$$\begin{gathered} P_{1D} = P_D \frac{-V_1}{V_2 - V_1} \\ P_{2D} = P_D \frac{V_2}{V_2 - V_1} \end{gathered} \tag{2.17}$$

The K curvature factors can be calculated as

$$\begin{gathered} K_1 = \frac{P_{1D}}{n_{1D} - 1} \\ K_2 = \frac{P_{2D}}{n_{2D} - 1} \end{gathered} \tag{2.18}$$

Finally, using as the values of K_1 and K_2, four radii of curvature of lens can be determined. The radii of curvature of two lenses satisfy

$$r_{12} = -r_{11} \quad r_{21} = r_{12} \quad and \quad r_{22} = \frac{r_{12}}{1 - K_2 r_{12}} \tag{2.19}$$

Radius of achromatic lens is demonstrated in Figure 2.23. In the design of an achromatic doublet, there are three indices of refraction for each of glasses and abbe number [8]. These values are taken from manufacturer's specification as shown in the Table 2.2.

Table 2.2 Sample of optical glasses

Type	Catalog code	V	n_C	n_D	n_F
		Abbe value	656.3 nm	587.6 nm	486.1
Borosilicate crown	517/645	64.55	1.51461	1.51707	1.52262
Borosilicate crown	520/636	63.59	1.51764	1.52015	1.52582
Light barium crown	573/574	57.43	1.56956	1.57259	1.57953
Dense barium crown	638/555	55.49	1.63461	1.63810	1.64611
Dense flint	617/366	36.60	1.61218	1.61715	1.62904
Flint	620/380	37.97	1.61564	1.62045	1.63198
Dense flint	689/312	31.15	1.68250	1.68893	1.70462
Dense flint	805/255	25.46	1.79608	1.80518	1.82771
Fused silica	458/678	67.83	1.45637	1.45846	1.46313

CHAPTER 3

RAY TRACING

3.1 Introduction

Ray tracing is a important technique for optical design and primary method used by optical engineers to determine optical system analysis. It is based on geometric optics that assumes light propagates like a straight ray and neglecting wave property of light. An other definition of ray tracing is a tracing a ray of light through a system by calculating the angle of refraction and height of ray on the optical axis at each surface. We can determine the location, size, and orientation of the image formed by the lens with tracing a few rays from the object through the lens [16].

3.1.1 y-u Trace for Thin Lenses

y-u trace is a great technique that use for evaluating complex optical system that consists of many thin lenses. we will use two different equation with derivation in this section; transfer equation and slope angle equation,respectively.

$$y_{k+1} = y_k + u_k t_k \tag{3.1}$$

$$u_{k+1} = u_k - y_{k+1}\phi_{k+1} \tag{3.2}$$

where y_k is the ray height, u_k is the ray angle, t_k is the distances, $\phi_{k+1} = 1/f_{k+1}$ is the powers. We know that light propagates rectilinear form in a vacuum with constant refractive index. We assume that the ray propagates right-handed coordinate system (in the y–z plane) as shown in figure 3.1 and choose the z-axis as the optical axis of the system [17]. We will proceed plane by plane from object plane to image plane that are perpendicular to the optical axis.

Firstly, we will look at ray between an object and lens surface. Ray is described by its distance from the optical axis, y_k, and by the angle u_k that is a angle with optical axis before refraction, and distance, t_k, is between two relative planes. As the ray propagates along the optical axis, it strikes each surfaces of the system and has different coordinates such as new angles, heights, distances that may change and take different values in different planes.
Derivation of a transfer equation: we select two reference planes separated by a distance t_k in homogeneous medium as seen from the Figure 3.1). There is a relationship between input values and output values. So, we can specify initial values because of ray starts with its initial coordinates (y_1, u_1). The angle remains constant but height will be change. In the paraxial region, each surfaces approaches a flat plane surface and all angles approach their sines and tangents [5]. Thus we can replace $\tan u_k$ by u_k.

$$y_{k+1} = y_k + u_k t_k \tag{3.3}$$

where t_k is the distance between the k and the $k+1$ surface and u_k is the ray angle defined earlier. This equation known as the transfer equation [7].

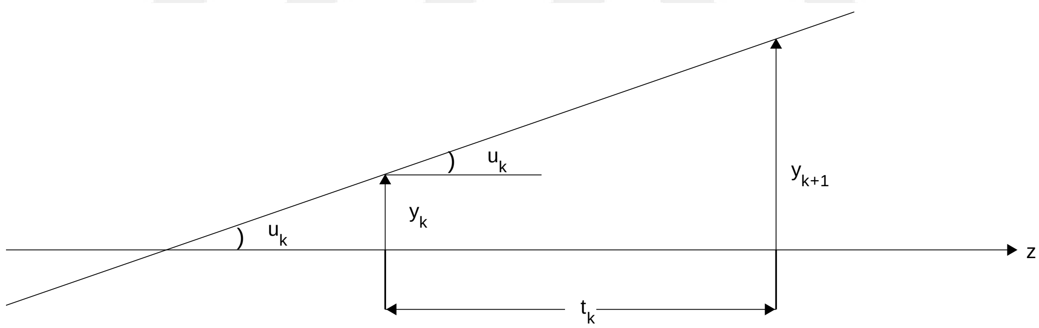

Figure 3.1 Ray propagation between two planes seperated by distance t_k [7].

When the light rays with an angle u_k with respect to the axis incident on a surface of thin lens at a height y_{k+1} from the axis it will be refracted to new angle u_{k+1} and intersect the axis distance t_k and $t_k + 1$ before and after refraction, as shown in Figure 3.2

we can apply thin lens equation

$$\frac{1}{t_{k+1}} = \frac{1}{f_{k+1}} + \frac{1}{t_k} \tag{3.4}$$

The slope angles of the rays are

$$\tan u_k = \frac{-y_{k+1}}{t_k} \qquad \text{and} \qquad \tan u_{k+1} = \frac{-y_{k+1}}{t_{k+1}}$$

We see minus sign in the two equation is because of the fact that the $\tan u_k$ and y_{k+1} are positive, but t_k is negative.On the other hand, y_{k+1} and t_{k+1} are positive, but u_{k+1} is negative [7].

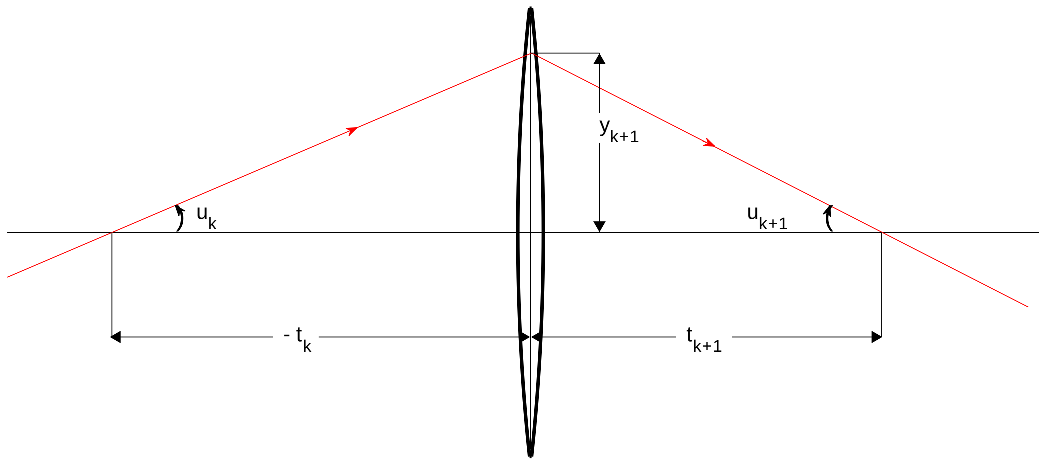

Figure 3.2 Ray propagation through a thin lens with focal length f_{k+1} [7].

$$t_k = \frac{-y_{k+1}}{u_k} \qquad \text{and} \qquad t_{k+1} = \frac{-y_{k+1}}{u_{k+1}}$$

where the paraxial approximation $\tan u_k = u_k$ and $\tan u_{k+1} = u_{k+1}$ substituting into in Equation 3.4

$$\frac{u_{k+1}}{y_{k+1}} = \frac{1}{f_{k+1}} - \frac{u_k}{y_{k+1}} \tag{3.5}$$

If we multiply Equation 3.5 by $-y_{k+1}$ we have an equation that provide the new angle by using ray height at the lens y_{k+1} and incident ray angle u_k

$$u_{k+1} = u_k - \frac{y_{k+1}}{f_{k+1}} \tag{3.6}$$

If we rearrange the equation to replace by $1/f_{k+1}$ and ϕ_{k+1}

$$u_{k+1} = u_k - y_{k+1}\phi_{k+1} \tag{3.7}$$

while f_{k+1} is the focal length of the lens, ϕ_{k+1} is known as power. Equation 3.7 may be called the slope angle equation [7].

3.1.2 y-nu Trace for Thick Lenses

y-nu trace method is similar to y-u trace that is used to thin lenses. But y-nu ray trace will be use thick lens. So we take into account radius of curvature, thickness of thick lens and other optical elements. y-nu tracing is more convenient to calculation rays of light as they interact with sequentially many optical surfaces. Therefore, we must define radius of curvature, refraction index and center of thickness of the lens. We will use two equation that are transfer equation and refraction equation. Transfer equation is the same as used in thin lens formula.

We will proceed surface by surface such as from surface k to surface $k+1$. The transfer equation is written as

$$y_{k+1} = y_k + u_k t_k \tag{3.8}$$

where y is the ray height at the surface, u is the slope angle, and t is the separation between the two surfaces.

3.1.3 Derivation of Refraction Equation

This equation must be derived to calculate the change height of the ray from axis and slope angle due to refraction at each plane [7]. According to paraxial approximation, light rays propagates very close the optical axis. In this approximation the sine and tangent angles are equal to its angle ($\sin u_k = u_k \ and \ \tan u_k = u_k$) and every lens surface approaches a flat plane surface [5]. In order to derive of refraction equation, we need to know refraction indices of two spaces, radius of curvature of the surface R and at the point where ray enters the second medium with slope angle u_k is changed to u_{k+1} and height of the ray y_{k+1} from the axis. Also at that point the ray incident on the surface at the angle I and refracted angle I' and local normal to the surface makes an angle α with the optic axis [7]. From the geometry shown in Figure 3.3. In the small angle approximation :

$$\begin{gathered}\alpha = \tan\alpha, \ u_k = \tan u_k, \ u_{k+1} = \tan u_{k+1} \\ u_k = \frac{u_{k+1}}{-t_k}, \quad -u_{k+1} = \frac{y_{k+1}}{t_{k+1}}, \quad -\alpha = \frac{y_{k+1}}{R_{k+1}}\end{gathered} \tag{3.9}$$

It can be observed from Figure 3.3 that

$$I = u_k - \alpha, \quad \alpha = u_{k+1} - I' \ or \ I' = u_{k+1} - \alpha. \tag{3.10}$$

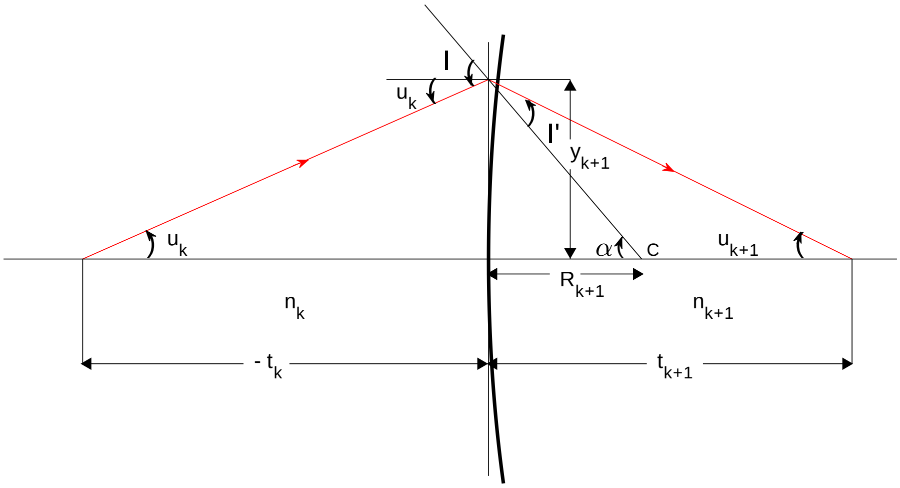

Figure 3.3 Refraction of a ray paraxial surface.

Snell's law $(n_{k+1} \ \sin I' = n_k \ \sin I)$ in the paraxial region decreases via the small angle approximation to

$$n_{k+1} \ I' = n_k \ I \tag{3.11}$$

substituting Equation 3.10 into Equation 3.12 one gets

$$n_{k+1}u_{k+1} = n_k u_k + (n_{k+1} - n_k)\alpha \tag{3.12}$$

and substituting for α, from Equation 3.9

$$n_{k+1}u_{k+1} = n_k u_k - (n_{k+1} - n_k)\frac{y_{k+1}}{R_{k+1}} \tag{3.13}$$

Equation 3.13 is known as refraction equation in paraxial optic [6]. There are some calculations that are very important in the paraxial optics [5]. One of them is effective focal length of the lens can be calculated as

$$\mathrm{e}ffl = f = \frac{\text{initial ray height}}{\text{final ray angle}} = \frac{-y_1}{u_{end}} \tag{3.14}$$

Another one is back focal length

$$\mathrm{b}fl = \frac{\text{final lens surface ray height}}{\text{final ray angle}} = \frac{-y_{end}}{u_{end}} \tag{3.15}$$

Reverse y-nu trace formula for the thick lens gives as follows

$$u_k n_k = u_{k+1}n_{k+1} + y_{k+1}P_{k+1} \tag{3.16}$$

$$y_k = y_{k+1} - u_k t_k \tag{3.17}$$

P is the power of a single refracting surface.

3.1.4 Meridional Exact Ray Tracing

Paraxial theory demonstrates perfect imagery by optical systems since all of rays each point on the object combine same image point. In fact, the paraxial image is no true representation of the object. And, as for real rays that will not intersect the image point at same point. Real ray tracing reveal aberration in an image. Accordingly, image will be blurred or distorted [9]. There are two ways to a apply exact ray tracing in the meridional plane(y-z) either use the equation developed for skew rays but x component approaches zero or the Q-U method developed by O'Shea, [7]. We will use different technique to determine exact ray tracing equations that contains line-circle intersection to find exact angle and height.

There are some assumptions for exact ray tracing:

- The ray starts with the initial coordinates $[y(k), u(k), z(k)]$ that are known.
- Apply transfer the ray equation between a tangent plane to the next actual surface. You should find actual surface coordinates such as [$y(k+1)$, $u(k+1)$, $z(k+1)$] as using line-circle intersection equation.
- you should find actual surface coordinates such as $[y(k+1), u(k+1), z(k+1)]$ as using line-circle intersection equation.
- Apply transfer equation to the next surface and repeat the sequence.

3.1.5 Intersection of a Line and a Circle

We will use two equation line equation and circle equation, respectively.
Line equation with respect to $k+1$ surface as shown in Equation 3.18

$$y_{k+1} = y_k + \tan u_k (z_{k+1} - z_k) \tag{3.18}$$

where $R_k + 1$ is of the curvature, $(z_{k+1} - z_k)$ is separation between object plane and a point where ray intercept the surface, y_{k+1} is a height at the actual surface, and $\tan u_k$ is the slope angle. If the center of the circle is at the origin, equation can be write $r^2 = y^2 + z^2$ where r is the radius. When the center of circle is the out of the origin at the point C(a,b), the equation becomes $(z-a)^2 + (y-b)^2 = r^2$. Figure 3.4 shows exact ray tracing and its calculation parameters.

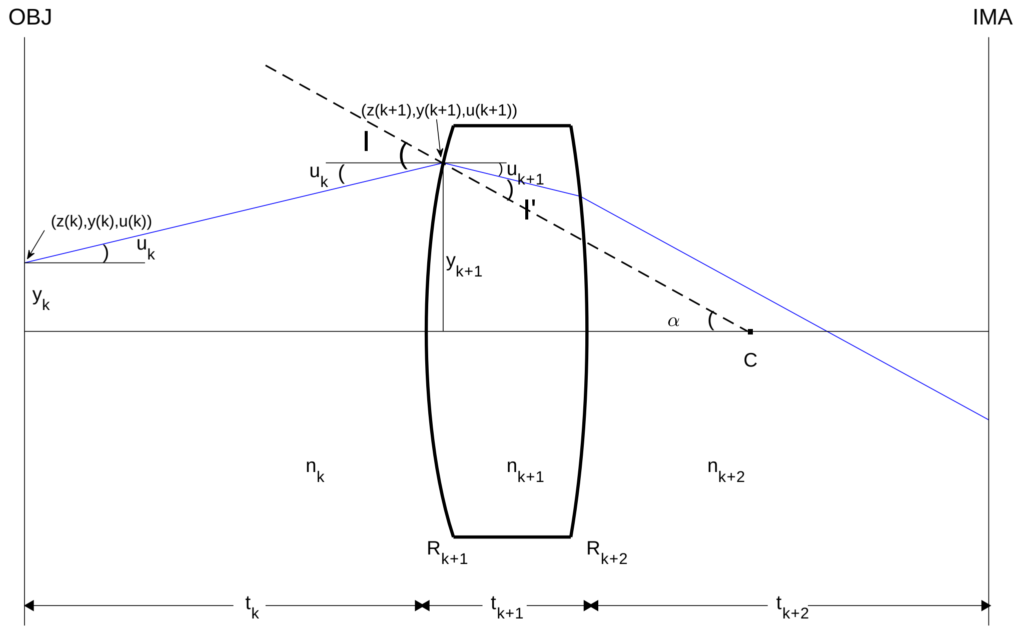

Figure 3.4 Exacat ray tracing with coordinates.

The geometry for the refraction of a ray at the actual surface is shown in Figure 3.4. We will look at the $k+1$ st surface where slope angle of the incident ray u_k is turn into u_k+1 after refraction and it has height y_{k+1} from the axis. The surface has radius R_{k+1} with the center of curvature at C. Incident angle I and refracted angle I' is measured from with respect to the normal to surface that makes an angle α with the optic axis and n_k, n_{k+1} are indices of refraction two different media.
It can be also observed from Figure 3.4 that

$\sin\alpha = \frac{y_{k+1}}{R_{k+1}}$, $I = u_k + \alpha$, $\alpha = I' - u_{k+1}$ or $u_{k+1} = I' - \alpha$

The circle equation :

$$(z_{k+1} - (R_{k+1} + t_k))^2 + (y_{k+1})^2 = (R_{k+1})^2 \tag{3.19}$$

One can substitute Equation 3.18 into Equation 3.19 to obtain for z_{k+1}, y_{k+1} as follows:

$$(z_{k+1} - (R_{k+1} + t_k))^2 + (y_k + \tan u_k(z_{k+1} - z_k))^2 = (R_{k+1})^2 \tag{3.20}$$

See Appendix A for the derivation of Equation 3.20. We have the values of I, u_k, α. To find I', we can apply Snell's law:

$$n_{k+1} \ \sin I' = n_k \ \sin(u_k + \alpha) \tag{3.21}$$

Finally, slope angle of refracted ray u_{k+1} is equal to difference $(I' - \alpha)$. Thereby, we have new coordinates, $(z_{k+1}, y_{k+1}, u_{k+1})$. The procedure can be repated iteratively until we reach image plane to find the intersection coordinates at the next surface.

CHAPTER 4

USE OF HEYSEM 1.0

4.1 Introduction

Heysem is the ray tracing program software developed in MATLAB 2016. MATLAB is a high-performance language for technical computing and known as matrix laboratory since its basic data element is matrix. It uses computations and algorithms to analyze large amounts of data and also include plotting of functions, application development, Graphical User Interface building (GUI). In this chapter the usage of the Heysem briefly is given.

4.2 Basic User Guide of Heysem Program

This program has different types of buttons, each of which serves a different purposes. A screenshot of the program is shown in Figure 4.1.

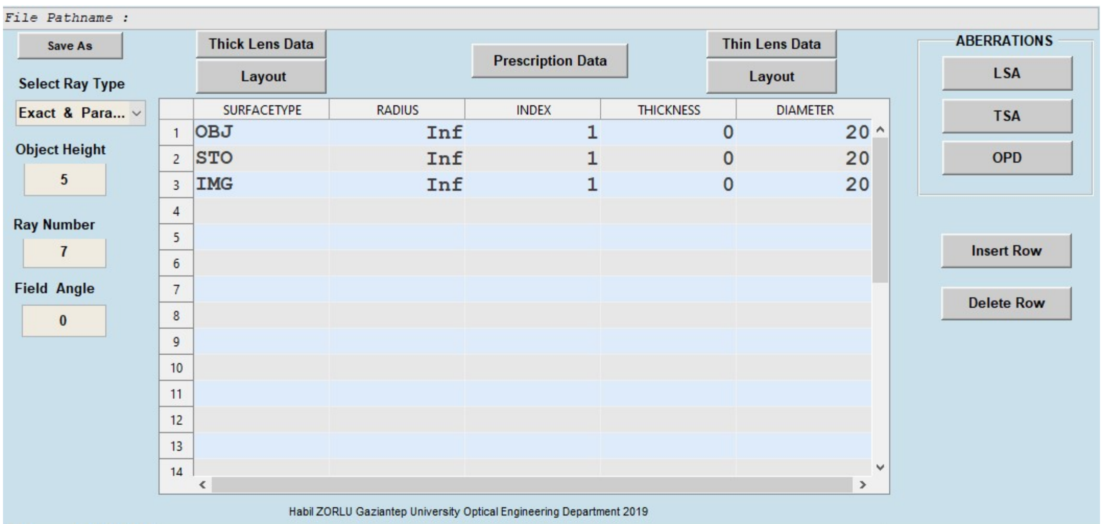

Figure 4.1 Graphical user interfaces of the Heysem program.

Basic manual is as follows:

- Selected ray type button provide the user three different ray layout. For example, user can choice both paraxial ray and exact ray at the same time or individually.
- Thick lens data button include values related to thick lens design that are radius of curvature, index of refraction, thickness between surfaces, diameter of each surfaces. The data is taken from a default file called lens.txt stored in the same folder as the program executable file.
- Thick lens layout button draws the figure related to thick lenses
- Prescription data generates some data as shown in Figure 4.8
- Thin lens data button include values related to thin lens which are focal length of the lens, thickness and diameter.
- Thin layout button draws the figures related to thin lenses
- User can enter any object height, number of rays and field angle. Default values related to object height, number of rays and field angle are defined as 5, 7, 0 respectively.
- Insert row button adds new surfaces anywhere in the table but firstly user should select the relevant surface.
- Delete row button removes selected surface.
- Aberration parts include tree buttons that are LSA button, TSA button and OPD button. LSA button draws a figure longitudinal spherical aberration versus ray height at he last surface of the system. TSA button draws a figure transverse aberration (TSA) versus final ray slope Tan(u). OPD button draws a figure optical path difference (OPD) versus ray height Y(Ray) at the last surface of the system.
- Gui table includes different parameters which are surface radius, index, thickness, diameter. Surface type consist of object plane(OBJ), image plane(IMG), aperture stop surface(STO) and standard surface(STD). Radius is define as radius of all lens surface. Object, aperture stop, image that are taken infinity(inf). Index is define as index of refraction of the lens and space index, image plan, object plane, aperture stop are taken as one. Thickness is defined as distance between surfaceses. User should take object distance as inf, if object at infinity. Otherwise, any value can be used object at finite distance.

- Save as button saves output into the file.

4.3 Example Applications

Example 1: A small biconvex lens has a center thickness 5 mm and an index of 1.5, and it is surrounded by by air. Assume that its first surface has a radius of 20 mm and its second surface a radius of 10 mm. We can determine some values about the optical system as using heysem program [15].

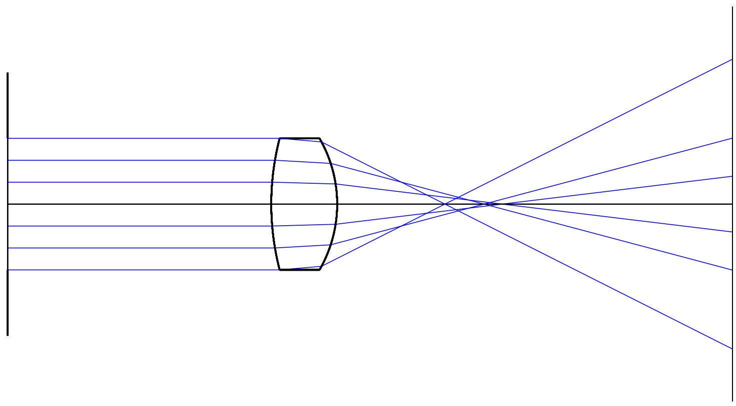

Figure 4.2 Layout related to example 1

Solution of the optical system as shown Figure 4.2 is given in general lens data as shown below Figure 4.3.

```
                GENERAL LENS DATA
Effective Focal Length   = 14.117647
Back Focal Length        = 12.941176
Front Focal Length       = -11.764706
Total Track              = 55.000000
Image Space F/#          = 1.411765
Object Space NA          = 1.00e-09
Stop Radius              = 5.00
Entrance Pupil Diameter  = 10.000000
Entrance Pupil Position  = 0.000000
Exit Pupil Diameter      = 17.142857
Exit Pupil Position      = 7.142857
Lens Units               = milimeters
Field Types              = Angle in degrees
Principal Plane OBJ      = 2.352941
Principal Plane IMJ      = -1.176471
Exact Ray Trace Data
------------------------------------------
Z-values          Y-values        U-values
------------------------------------------
0.000000e+00     5.000000e+00     0.000000
0.000000e+00     5.000000e+00     0.000000
2.063508e+01     5.000000e+00     -0.085232
2.381136e+01     4.728621e+00     -0.467538
5.500000e+01     -1.101764e+01    -0.467538
Paraxial ray Trace Data
------------------------------------------
z-values          y-values        u-values
------------------------------------------
0.000000e+00     5.000000e+00     0.000000
0.000000e+00     5.000000e+00     0.000000
2.000000e+01     5.000000e+00     -0.083333
2.500000e+01     4.583333e+00     -0.354167
5.500000e+01     -6.041667e+00    -0.354167
```

Figure 4.3 Prescription data related to example 1.

Example 2: Figure 4.5 and Figure 4.7 shows a typical problem that is designed both in zemax and heysem program. The optical system consist of three surfaces and have radii, thicknesses and indices. The object is located at infinity distance left of the first surface 10 mm above the axis. The lens is immersed in air [5]. We have the values related to optical seystem as follows:

$R_1 = 50 \quad t_1 = inf \quad n_1 = 1$

$R_2 = -50 \quad t_2 = 10 \quad n_2 = 1.5$

$R_3 = -50 \quad t_3 = 2 \quad n_3 = 1.6$

$R_4 = plano \quad t_4 = 60 \quad n_4 = 1$

All calculations are calculated both in zemax and heysem program. Results are given in the prescription data for two programs as shown in Figures 4.6 and 4.8, respectively.

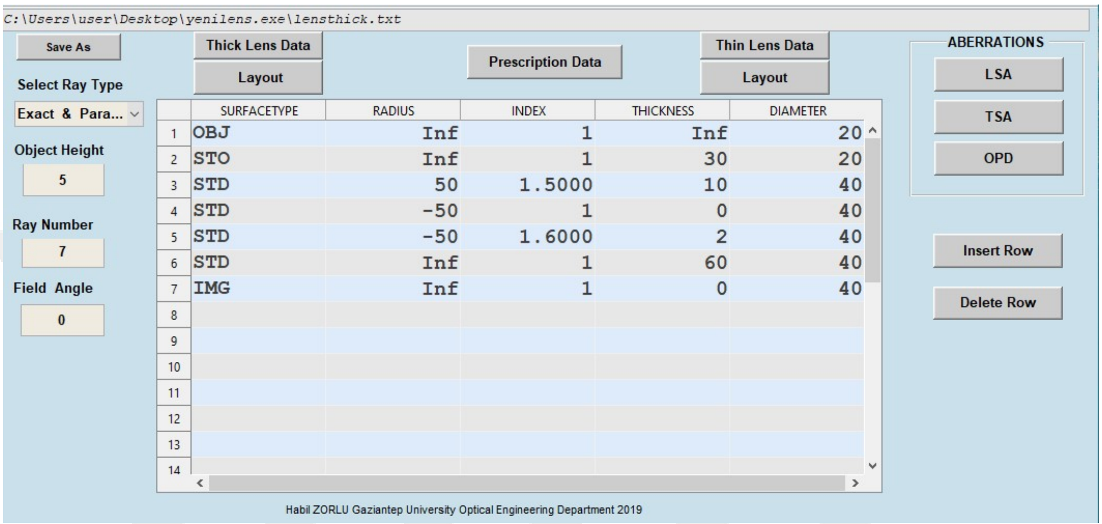

Figure 4.4 Lens design graphical user interfaces. All values related to given example.

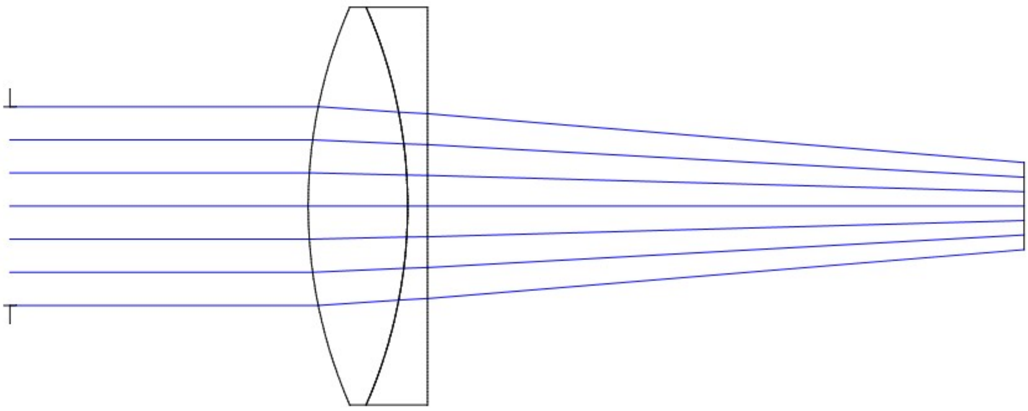

Figure 4.5 Layout related to example 2 in zemax program.

```
GENERAL LENS DATA:

Surfaces                   :                 6
Stop                       :                 1
System Aperture            : Entrance Pupil Diameter = 20
Glass Catalogs             : SCHOTT
Ray Aiming                 : Off
Apodization                : Uniform, factor =   0.00000E+000
Temperature (C)            :    2.00000E+001
Pressure (ATM)             :    1.00000E+000
Adjust Index Data To Environment  : Off
Effective Focal Length     :        122.9508 (in air at system
Effective Focal Length     :        122.9508 (in image space)
Back Focal Length          :        113.5041
Total Track                :             102
Image Space F/#            :        6.147541
Paraxial Working F/#       :        6.147541
Working F/#                :         6.10844
Image Space NA             :      0.08106565
Object Space NA            :          1e-009
Stop Radius                :              10
Paraxial Image Height      :               0
Paraxial Magnification     :               0
Entrance Pupil Diameter    :              20
Entrance Pupil Position    :               0
Exit Pupil Diameter        :        25.99653
Exit Pupil Position        :       -106.3107
Field Type                 : Angle in degrees
Maximum Radial Field       :               0
Primary Wavelength         :            0.55 µm
Lens Units                 :   Millimeters
Angular Magnification      :               0
```

Figure 4.6 The prescription data related to example 2 in zemax program.

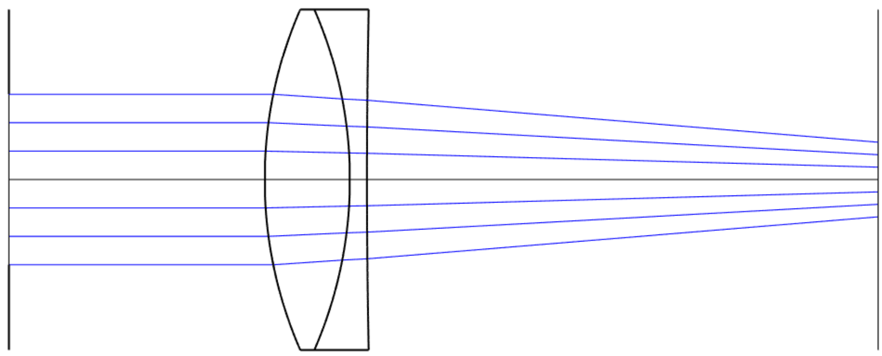

Figure 4.7 Layout related to example 2 in heysem program.

```
                GENERAL LENS DATA
Effective Focal Length    = 122.950820
Back Focal Length         = 113.504098
Front Focal Length        = -124.590164
Total Track               = 102.000000
Image Space F/#           = 6.147541
Object Space NA           = 1.00e-09
Stop Radius               = 10.00
Entrance Pupil Diameter   = 20.000000
Entrance Pupil Position   = 0.000000
Exit Pupil Diameter       = 25.996534
Exit Pupil Position       = -106.310659
Lens Units                = milimeters
Field Types               = Angle in degrees
Exact Ray Trace Data
-------------------------------------------
Z-values          Y-values        U-values
-------------------------------------------
0.000000e+00     1.000000e+01    0.000000
0.000000e+00     1.000000e+01    0.000000
3.101021e+01     1.000000e+01    -0.067626
3.909843e+01     9.452187e+00    -0.202260
3.909843e+01     9.452187e+00    -0.051181
4.200000e+01     9.303552e+00    -0.081946
1.020000e+02     4.375779e+00    -0.081946
Paraxial ray Trace Data
-------------------------------------------
z-values          y-values        u-values
-------------------------------------------
0.000000e+00     1.000000e+01    0.000000
0.000000e+00     1.000000e+01    0.000000
3.000000e+01     1.000000e+01    -0.066667
4.000000e+01     9.333333e+00    -0.193333
4.000000e+01     9.333333e+00    -0.050833
4.200000e+01     9.231667e+00    -0.081333
1.020000e+02     4.351667e+00    -0.081333
```

Figure 4.8 The prescription data related to example 2 in heysem program.

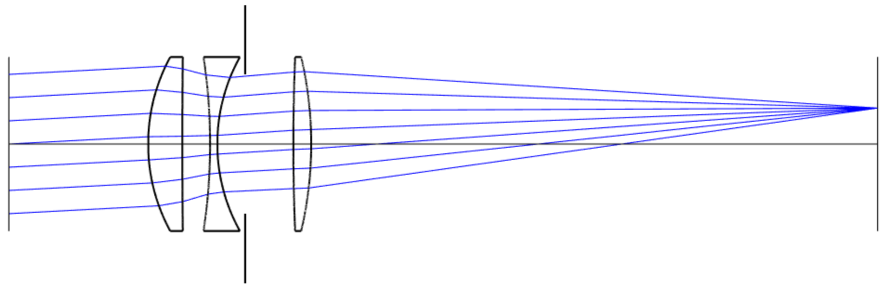

Figure 4.9 Layout related to cooke triplet camera lens.

Example 3: This example is related to Cooke triplet camera lens layout as shown Figure and determine some values [5]. Required values about the system are given as follows:

$R_1 = 26.160$	$t_1 = 4.916$	$n_1 = 1.678$
$R_2 = 1201.700$	$t_2 = 3.988$	$n_2 = 1$
$R_3 = -83.460$	$t_3 = 1.033$	$n_3 = 1.648$
$R_4 = 25.670$	$t_4 = 4.00$	$n_4 = 1$
$R_5 = STOP$	$t_5 = 6.925$	$n_5 = 1$
$R_6 = 302.610$	$t_6 = 2.567$	$n_6 = 1.651$
$R_7 = -54.790$	$t_7 = 81.433$	$n_7 = 1$

Results of optical system calculations are as indicated below :

Effective focal length (EFFL) = 98.657496

Back focal length (BFL) = 81.518244

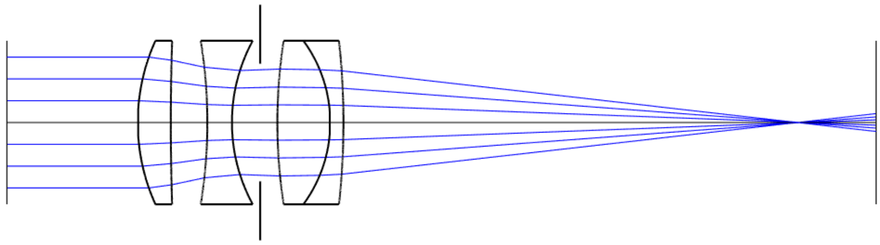

Figure 4.10 Layout related to tessar design.

Example 4: This example is related to layout and compute some values Tessar lens design as shown Figure 4.10 [5]. Required values about the system are given as follows:

$R_1 = 30.322$	$t_1 = 5.054$	$n_1 = 1.620$
$R_2 = 390.086$	$t_2 = 5.579$	$n_2 = 1$
$R_3 = -78.533$	$t_3 = 3.760$	$n_3 = 1.575$
$R_4 = 26.128$	$t_4 = 4.320$	$n_4 = 1$
$R_5 = STOP$	$t_5 = 2.634$	$n_5 = 1$
$R_6 = 82.072$	$t_6 = 8.076$	$n_6 = 1.1.639$
$R_7 = -21.128$	$t_7 = 2.021$	$n_7 = 1$
$R_8 = -21.128$	$t_8 = 0$	$n_8 = 1.523$
$R_9 = -114.906$	$t_9 = 81.484$	$n_9 = 1$

Effective focal length (EFFL) = 99.755850
Back focal length (BFL) = 81.307107

Example 5: We assumed that an optical system (as seen from Figure)is made up of a positive thin lens that has diameter 6 cm and focal length 6 cm. Another lens is negative lens that has 6 cm diameter and its focal length -10 cm. The aperture is located 3 cm in front of first lens. The distance between two lenses is 4 cm [8].

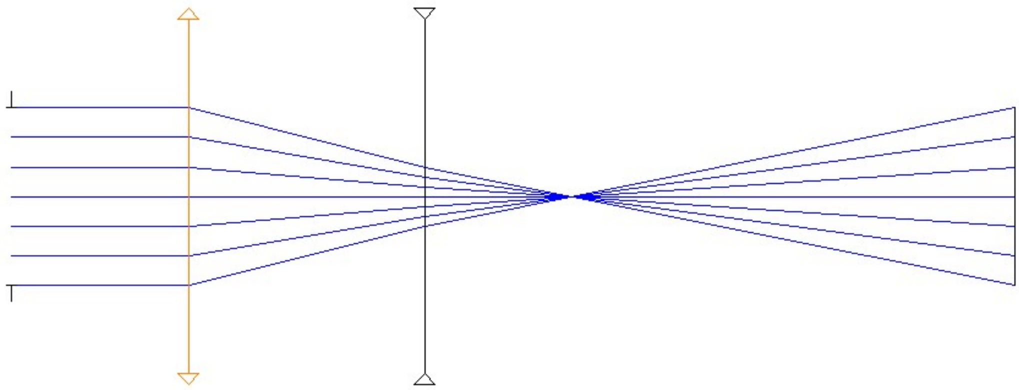

Figure 4.11 Layout related to example 5 in zemax program.

Calculations due to the example 5 as shown Figure 4.11 are given in general lens data as shown below Figure 4.12.

```
GENERAL LENS DATA:

Surfaces                  :                 4
Stop                      :                 1
System Aperture           : Entrance Pupil Diameter = 3
Glass Catalogs            : SCHOTT
Ray Aiming                : Off
Apodization               : Uniform, factor =   0.00000E+000
Temperature (C)           :    2.00000E+001
Pressure (ATM)            :    1.00000E+000
Adjust Index Data To Environment  : Off
Effective Focal Length    :               7.5 (in air at system temperature and pressure)
Effective Focal Length    :               7.5 (in image space)
Back Focal Length         :               9.5
Total Track               :                17
Image Space F/#           :               2.5
Paraxial Working F/#      :               2.5
Working F/#               :           2.54951
Image Space NA            :         0.1961161
Object Space NA           :          1.5e-010
Stop Radius               :               1.5
Paraxial Image Height     :                 0
Paraxial Magnification    :                 0
Entrance Pupil Diameter   :                 3
Entrance Pupil Position   :                 0
Exit Pupil Diameter       :                 3
Exit Pupil Position       :               -15
Field Type                : Angle in degrees
Maximum Radial Field      :                 0
Primary Wavelength        :              0.55 µm
Lens Units                :   Millimeters
Angular Magnification     :                 0
```

Figure 4.12 The prescription data related to example 5 in zemax program.

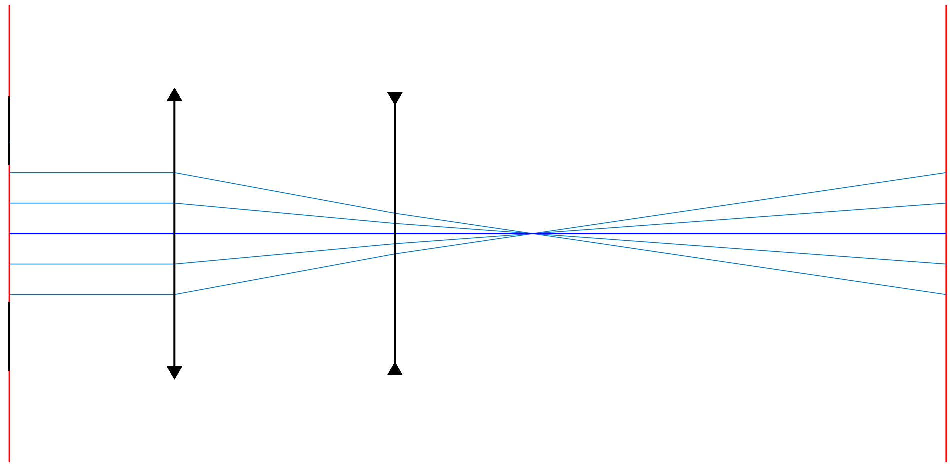

Figure 4.13 Layout related to example 5 in heysem program.

Calculations with the example 5 are given in the heysem program lens data as shown in the Figure 4.14

```
                  GENERAL LENS DATA
Effective Focal Length     = 7.500000
Back Focal Length          = 2.500000
Total Track                = 17.000000
Image Space F/#            = 2.500000
Object Space NA            = 1.00e-09
Stop Radius                = 1.50
Entrance Pupil Diameter    = 3.000000
Entrance Pupil Position    = 0.000000
Exit Pupil Diameter        = 3.000000
Exit Pupil Position        = -15.000000
Lens Units                 = milimeters
Field Types                = Angle in degrees
Exact Ray Trace Data
-------------------------------------------
z-values          y-values        u-values
-------------------------------------------
0.000000e+00      2.000000e+00    0.000000
0.000000e+00      2.000000e+00    0.000000
3.000000e+00      2.000000e+00    -0.333333
7.000000e+00      6.666667e-01    -0.266667
1.700000e+01      -2.000000e+00   -0.266667
Paraxial Ray Trace Data
-------------------------------------------
z-values          y-values        u-values
-------------------------------------------
0.000000e+00      2.000000e+00    0.000000
0.000000e+00      2.000000e+00    0.000000
3.000000e+00      2.000000e+00    -0.333333
7.000000e+00      6.666667e-01    -0.266667
1.700000e+01      -2.000000e+00   -0.266667
```

Figure 4.14 The prescription data related to example 5 in heysem program.

CHAPTER 5

CONCLUSION

A basic paraxial and exact ray tracing program called Heysem 1.0 for the optical lens design has been developed by using MATLAB 2016. The program may especially be used for educational purpose for undergraduate students. Optical and geometrical system data (such as lens surfaces and indices of refraction of lenses) are input to Heysem having a simple lens data editor similar to Zemax. The outcomes (layout, prescription data) of Heysem and Zemax are found to be the same.

Heysem is only performing meridional ray tracing and related calculations. However, traditional ray tracing software programs (Zemax, Code V, TracePro and etc) are good at calculations of skew rays, tolerancing, thermal analysis and physical optics.

In the next version of Heysem, first, it is suggested to include skew ray tracing and suitable spot diagram at image surface.

REFERENCES

[1] Zemax, Version June 9 2009. https://www.zemax.com.

[2] Optical design software.
https://www.synopsys.com/optical-solutions/codev.html, 20.07.2019.

[3] MathWorks, User's Guide(R2016a). https://www.mathworks.com/help/pdf_doc/matlab/index.html, 20.03.2019.

[4] Ibn al-Haytham, Arab astronomer and mathematician. https://www.britannica.com/biography/Ibn-al-Haytham,18.07.2019

[5] Smith, W. J. (2008). *Modern optical engineering.* Tata McGraw-Hill Education.

[6] Dereniak, E. L., Dereniak, T. D. (2008). *Geometrical and trigonometric optics.* Cambridge University Press.

[7] O'shea, D. C., C'Shea, D. C. (1985). *Elements of modern optical design* (**Vol. 51)**. Wiley: New York.

[8] Pedrotti, F. L., Pedrotti, L. M., Pedrotti, L. S. (2006). *Introduction to optics*, 3rd Edition.

[9] Katz, M. (2002). *Introduction to geometrical optics.* World Scientific Publishing Company.

[10] Kingslake, R., Johnson, R. B. (2009). *Lens design fundamentals.* Academic Press.

[11] Greivenkamp, J. E. (2003). Field Guide to Geometrical Optics (**SPIE Vol. FG01**).

[12] Kidger, M. J. (2002). *Fundamental optical design* (**Vol. 92**). Bellingham, SPIE Press: WA.

[13] DiMarzio, C. A. (2011). *Optics for engineers.* Crc Press.

[14] Shannon, R. R. (1997). *The art and science of optical design.* Cambridge University Press.

[15] Hecht, E. (2017). *Optics.* 5^{nd} edition. Pearson Education.

[16] Sun, H. (2016). *Lens Design: A Practical Guide*. Crc Press.

[17] Kloos, G. (2007). *Matrix methods for optical layout* (**Vol. 77**). Bellingham, SPIE Press: Washington.

APPENDIX A

LINE AND CIRCLE INTERSECTION

Equation of a line

$$y = mz + n \tag{A.1}$$

The basic equation for a straight line is shown in Equation A.1, where n is the height of the line at $z = 0$ and m is the gradient.

If the center of the circle is at he origin of the coordinate system, the equation of the circle is

$$(z)^2 + (y)^2 = R^2 \tag{A.2}$$

where R is the radius of the circle.

If the center of the circle is at the point (a,b), the equation of circle becomes

$$(z - a)^2 + (y - b)^2 = R^2 \tag{A.3}$$

firstly, we substitute Equation A.1 into Equation A.3

$$(z - a)^2 + (mz + n - b)^2 = R^2 \tag{A.4}$$

Next, if we expand all brackets and bring the R over to the left

$$(z^2 + a^2 - 2az + m^2 z^2 + (n - b)^2 + 2m(n - b)z - R^2 = 0 \tag{A.5}$$

Rearranging the equation, we get the following quadratic equation:

$$(1 + m^2)z^2 + 2(m(n - b) - a)z + a^2 + (n - b)^2 - R^2 = 0 \tag{A.6}$$

This equation look like mess, so if we replace $(1+m^2)$ by capital A , $2(m(n-b)-a)$ by capital B,$a^2 + (n - b)^2 - R^2$ by capital C
The new equation is

$$Az^2 + Bz + C = 0 \tag{A.7}$$

Then we can apply the quadratic formula to find the roots of this equation. Recall that a quadratic equation

$$z_{1,2} = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \tag{A.8}$$

If $B^2 - 4AC < 0$ then the line misses the circle. There are no roots.
If $B^2 - 4AC = 0$ then the line is tangent to the circle. There is single root for the equation.
If $B^2 - 4AC > 0$ then the line meets the circle in two distinct points. There are two real roots to the equation.
since $y = mz + n$. If we put z values in the line equation, we have two different y values. However, we must select just one value of z and y.

APPENDIX B

RAY TRCING MATLAB CODES

```
 % yu ray trace function  for thin lens

function [z y u] = yuthin(z,y,u,t,P,ASindex)
if nargin == 6
i = ASindex;
z(i) = z(ASindex);
else
i = 1;
z = zeros(1,length(y));
end

for k = i:length(t)
z(k+1) = z(k) + t(k);
y(k+1) = y(k) + u(k)*t(k);
u(k+1) = u(k) - y(k+1)*P(k+1);
end
end
% ryu ray trace function for thin lens

function [z y u] = ryuthin(z,y,u,t,P,ASindex)
for k = ASindex-1:-1:1
z(k) = z(k+1) - t(k);
u(k) = u(k+1) + y(k+1)*P(k+1);
y(k) = y(k+1) - u(k)*t(k);
end
end


% plotting thin lens function

function plottingthin(z,y,D,t,P,ASindex)
N = length(t);
xmax = sum(t);
ymax = max(D)/2;
yf   = abs(D(end)/2);
yi   = D(1)/2;
if yf > ymax
ymax = yf;
end
drawGeometry(xmax,ymax,yi,yf);
```

```
% draw lenses and aperture stop
for j=2:N
if j==ASindex
drawAperture(z(j),D(j));
else
drawLens(z(j), D(j), P(j));
end
end

for k = 1:N
if abs(y(k)) > D(k)/2
break;
end
line([z(k) z(k+1)], [y(k) y(k+1)]);
end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function drawGeometry(xmax, ymax, yi, yf)
axis([-xmax*0.1 xmax*1.1 -ymax*1.5 ymax*1.5]);
line([0 xmax], [0 0], 'Color','b');        % optical axis
line([0 xmax], [0 0], 'Color','b');        % optical axis
line([0 0],[-yi yi],'Color','r');          % OBJ plane
line([xmax xmax],[-yf yf],'Color','r');    % IMA plane
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function drawLens(z, diameter, power)
r = diameter/2;
line([z z], [-r r], 'LineWidth',1, 'color','k');
hold on
if power > 0
plot(z, r,'k^','MarkerFaceColor','k','MarkerSize',7);
plot(z,-r,'kV','MarkerFaceColor','k','MarkerSize',7);
else
plot(z, r,'kV','MarkerFaceColor','k','MarkerSize',7);
plot(z,-r,'k^','MarkerFaceColor','k','MarkerSize',7);
end
hold off

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function drawAperture(z, y)
line([z z], [-y/2 -y], 'LineWidth',1, 'color','k');
line([z z], [+y/2 +y], 'LineWidth',1, 'color','k');
end


% this function returns intersection of two lines for thin lens

function [z y] = lineintersectthin(x1,y1,x2,y2, X1,Y1,X2,Y2)
m1 = (y2-y1)/(x2-x1);
m2 = (Y2-Y1)/(X2-X1);
n1 = -m1*x1+y1;
n2 = -m2*X1+Y1;
z  = (n1-n2)/(m2-m1);
y  = m1*z + n1;
end

% this function returns Exit pupil position and height for thin lens
```

```
function [ExPx ExPy] = getExPupilthin(z,D,t,P,ASi)
Y(ASi) = D(ASi)/2;
U(ASi) = 0;
X(ASi) = z(ASi);
[X Y U] = yuthin(X,Y,U,t,P,ASi);
x1=X(end-1); x2=X(end);
y1=Y(end-1); y2=Y(end);

Y(ASi) = D(ASi)/2;
U(ASi) = 0.05;
X(ASi) = z(ASi);
[X Y U]= yuthin(X,Y,U,t,P,ASi);

X1=X(end-1); X2=X(end);
Y1=Y(end-1); Y2=Y(end);

[ExPx ExPy] = lineintersectthin(x1,y1,x2,y2, X1,Y1,X2,Y2);
end



% This function returns Entrance  pupil position and height

function [EnPx EnPy] = getEnPupilthin(z,D,t,P,ASi)
if(ASi<=2)
EnPx = z(ASi);
EnPy = D(ASi);
return
end

Y = zeros(1,ASi);
U = zeros(1,ASi);
X = zeros(1,ASi);

Y(ASi) = D(ASi)/2;
U(ASi) = 0;
X(ASi) = z(ASi);
[X Y U] = ryuthin(X,Y,U,t,P,ASi);
x1=X(1); y1=Y(1);
x2=X(2); y2=Y(2);

Y(ASi) = D(ASi)/2;
U(ASi) = 0.05;
X(ASi) = z(ASi);
[X Y U] = ryuthin(X,Y,U,t,P,ASi);

X1=X(1); Y1=Y(1);
X2=X(2); Y2=Y(2);

[EnPx EnPy] = lineintersectthin(x1,y1,x2,y2, X1,Y1,X2,Y2);
end


reverse ynu trace function
function [z,y,u]= rynu(z,y,u,R,t,n,ASi)

for k = ASi:-1:1
```

```
p(k+1) = (n(k+1)-n(k))/R(k+1);
end


for k = ASi-1:-1:1
z(k) = z(k+1) - t(k);
u(k) = (u(k+1)*n(k+1) + y(k+1)*p(k+1))/n(k);
y(k) = y(k+1) - u(k)*t(k);
end

end

% this fuction returns plotting thick lens

function  plotting(D,R,t,n,z,Z,y,Y,ASi,handles)

N= length(t);
zmax = sum(t);
drawGeometry(zmax,D);

for j=1:N

if  j==ASi
drawAperture(z(j),D(j));
else
drawLens(D,R,t,n,ASi);
end
end
light = get(handles.btn_lightselection,'value');
if light==1
paraxialray(z,y,D);
exactray(Z,Y,D);
elseif light==2
exactray(Z,Y,D);
elseif light==3
paraxialray(z,y,D);
end
end




function paraxialray(z,y,D)
N=length(z)-1;

for k=1:N
if abs(y(k))>abs(D(k)/2)
break;
else
line([z(k) z(k+1)], [y(k) y(k+1)],'color','r');%paraxial  ;
end
end
end

function exactray(Z,Y,D)
N=length(Z)-1;
for k=1:N
if  abs(Y(k))>abs(D(k)/2)
break;
```

```
else
line([Z(k) Z(k+1)], [Y(k) Y(k+1)],'color','b');%exact
end
end
end


function drawLens(D,R,t,n,ASi)
N= length(t);

for j=2:length(D)-1
if R(j)==inf
R(j)=1.e3;
R(ASi)=inf;
end
[a b] = arc( R(j),sum(t(1:j-1)),D(j) );
v(j-1,:)=a;
d(j-1,:)=b;
end

w= length(v);
for j=1:w
if  n(j+1)>1
line([v(j) v(j+1)],[d(j,1)   d(j,1)],'Color','k','LineWidth',1);
line([v(j) v(j+1)],[-d(j,1) -d(j,1)],'Color','k','LineWidth',1);
end
end
end %function drawLens end



function drawAperture(z, D)
line([z z], [-D/2 -D], 'LineWidth',1, 'color','k');
line([z z], [+D/2  D], 'LineWidth',1, 'color','k');
end



function drawGeometry(zmax,D)
% optical axis
line([0 zmax], [0 0], 'Color','k');
% OBJ plane
line([0 0],[-D(1)/2 D(1)/2],'Color','k');
% IMA plane
line([zmax zmax],[-D(end)/2 D(end)/2],'Color','k');
% determine axis size
xlim([-zmax/10 zmax*1.1])
axis equal
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%paraxial ray trace function for thick lens
function [z,y,u] = paraxial(z,y,u, R, t, n,ASi)

% Number of surfaces
N = length(t);
p = zeros(1,N+1);
```

```
for k = 1:N
p(k+1) = (n(k+1)-n(k))/R(k+1);
end

if nargin == 7
i=ASi;
z(i)=z(ASi);
else
i=1;
z = zeros(1,length(y));
end

for k = i:N
z(k+1) = z(k) + t(k);
y(k+1) = y(k) + u(k)*t(k);
u(k+1) = (n(k)*u(k)-p(k+1)*y(k+1))/n(k+1);
end

end

% optical path length function for a ray

function [opd,b] = OPL(z,y,Z,Y,n,ASi)
N = length(Z);
for i=1:N-1
S(i) = n(i).*sqrt((Z(i+1)-Z(i))^2+(Y(i+1)-Y(i))^2);
s(i) = n(i).*sqrt((z(i+1)-z(i))^2+(y(i+1)-y(i))^2);

end
opd = sum(s)-sum(S);
if ASi==N-1
b=(Y(end-2));
else
b=(Y(end-1));
end

end

% LSA function for a ray

function [lsa,tsa,a,uu] = LSA(z,y,u,Y,U,ASi)
N=length(z);
l=y(end)/tan(u(end));
L=Y(end)/tan(U(end));
lsa=l-L;
if ASi==N-1
a=abs(Y(end-2));
else
a=abs(Y(end-1));
end
tsa=y(end)-Y(end);
uu=tan((U(end)));
end
% line-circle intersection fuction

function [Z,Y]= intercept(z0,y0,u0,R,t,b)
if nargin==5
b=0;
end
```

```
m=tan(u0);
n=-tan(u0)*z0+y0;

a=R+t;
A=1+m^2;
B=2*(m*(n-b)-a);
C=a^2+(n-b)^2-R^2;
Z1=(-B-sqrt(B^2-4*A*C))/(2*A);
Z2=(-B+sqrt(B^2-4*A*C))/(2*A);
Y1=m*Z1+n;
Y2=m*Z2+n;

D1=sqrt((Z1-t)^2+Y1^2);
D2=sqrt((Z2-t)^2+Y2^2);
if D1<D2
Z=Z1;
Y=Y1;
else
Z=Z2;
Y=Y2;
end %end of if
if abs(R)>1e10
Z = t;
Y=m*Z+n;
end
end %end of function

%Returns Exit pupil position and height for thick lens

function [ExPx ExPy] = getExPupil(z,D,t,R,n,ASi)

yy = zeros(1,ASi);
uu = zeros(1,ASi);
zz = zeros(1,ASi);
yy(ASi) = D(ASi)/2;
uu(ASi) = 0;
zz(ASi) = z(ASi);

[zz yy uu] = paraxial(zz,yy,uu,R,t,n,ASi);
[zz' yy' uu'];

x1=zz(end-1); x2=zz(end);
y1=yy(end-1); y2=yy(end);
yy(ASi) = D(ASi)/2;
uu(ASi) = -0.1;
zz(ASi) = z(ASi);
[zz yy uu]=paraxial(zz,yy,uu,R,t,n,ASi);
[zz' yy' uu'];

X1=zz(end-1); X2=zz(end);
Y1=yy(end-1); Y2=yy(end);

[ExPx ExPy] = lineintersect(x1,y1,x2,y2, X1,Y1,X2,Y2);
end

%Returns Entereance pupil position and height for thick lens
function [EnPx EnPy] = getEnPupil(z,D,t,R,n,ASi)
```

```
if(ASi<=2)
EnPx = z(ASi);
EnPy = D(ASi);
return
end

yy = zeros(1,ASi);
uu = zeros(1,ASi);
zz = zeros(1,ASi);
yy(ASi) = D(ASi)/2;
uu(ASi) = 0;
zz(ASi) = z(ASi);
[zz yy uu] = rynu(zz,yy,uu,R,t,n,ASi);%rynu(z,y,u,R,t,n,ASi)
[zz' yy' uu'];

x1=zz(1); y1=yy(1);
x2=zz(2); y2=yy(2);

yy(ASi) = D(ASi)/2;
uu(ASi) = 0.1;
zz(ASi) = z(ASi);
[zz yy uu] = rynu(zz,yy,uu,R,t,n,ASi);
[zz' yy' uu'];
X1=zz(1); Y1=yy(1);
X2=zz(2); Y2=yy(2);


[EnPx EnPy] = lineintersect(x1,y1,x2,y2, X1,Y1,X2,Y2);
end

%exact ray tracing function

function [Z,Y,U] = exact(z0,y0, u0, R, t, n)
% Number of surfaces
N = length(t);
% Initial values (height and angle)
Z(1) = z0;
Y(1) = y0;
U(1) = u0;
b=0;
%exact ray tracing
for k=1:N
[Z(k+1),Y(k+1)]= intercept(Z(k),Y(k),U(k),R(k+1),sum(t(1:k)),b);
q  = asin(Y(k+1)/R(k+1));%q is a center angle
Ip = asin((n(k)/n(k+1))*sin(U(k)+q));
U(k+1) = Ip-q;
end

end

%this functions draws arcs

function [a,b] = arc(Radius,t,D)

n = 500;
theta1 = pi-asin(0.5*abs(D/Radius));
theta2 = pi+asin(0.5*abs(D/Radius));
theta  = theta1:1/n:theta2;
x = Radius*cos(theta) + Radius + t;
```

```
y = Radius*sin(theta);
a = x(1);
b = y(1);
plot(x,y,'k-','LineWidth',1);
hold on
end
```

```
%Matlab gui program
```

```
function varargout = lensdesign(varargin)
% UNTITLED MATLAB code for untitled.fig
%      UNTITLED, by itself, creates a new UNTITLED or raises the existing
%      singleton*.
%
%      H = UNTI TLED returns the handle to a new UNTITLED or the handle to
%      the existing singleton*.
%
%      UNTITLED('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in UNTITLED.M with the given input arguments.
%
%      UNTITLED('Property','Value',...) creates a new UNTITLED or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before untitled_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to untitled_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help untitled

% Last Modified by GUIDE v2.5 12-Jul-2019 16:32:28

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @untitled_OpeningFcn, ...
'gui_OutputFcn',  @untitled_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% clc;
% clear;
```

```
% --- Executes just before untitled is made visible.
function untitled_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to untitled (see VARARGIN)

% Choose default command line output for untitled
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes untitled wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = untitled_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;




% --- Executes on selection change in btn_lightselection.
function btn_lightselection_Callback(hObject, eventdata, handles)
% hObject    handle to btn_lightselection (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns btn_lightselection contents as cell array
%        contents{get(hObject,'Value')} returns selected item from btn_lightselection


% --- Executes during object creation, after setting all properties.
function btn_lightselection_CreateFcn(hObject, eventdata, handles)
% hObject    handle to btn_lightselection (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end


% --- Executes when entered data in editable cell(s) in uitable1.
function uitable1_CellEditCallback(hObject, eventdata, handles)
% hObject    handle to uitable1 (see GCBO)
```

```
% eventdata  structure with the following fields (see MATLAB.UI.CONTROL.TABLE)
%       Indices: row and column indices of the cell(s) edited
%       PreviousData: previous data for the cell(s) edited
%       EditData: string(s) entered by the user
%       NewData: EditData or its converted form set on the Data property. Empty if Data was not chang
%       Error: error string when failed to convert EditData to appropriate value for Data
% handles    structure with handles and user data (see GUIDATA)


function btn_rays_Callback(hObject, eventdata, handles)
% hObject    handle to btn_rays (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,’String’) returns contents of btn_rays as text
%        str2double(get(hObject,’String’)) returns contents of btn_rays as a double


% --- Executes during object creation, after setting all properties.
function btn_rays_CreateFcn(hObject, eventdata, handles)
% hObject    handle to btn_rays (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,’BackgroundColor’), get(0,’defaultUicontrolBackgroundColor’))
set(hObject,’BackgroundColor’,’white’);
end



function btn_height_Callback(hObject, eventdata, handles)
% hObject    handle to btn_height (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,’String’) returns contents of btn_height as text
%        str2double(get(hObject,’String’)) returns contents of btn_height as a double


% --- Executes during object creation, after setting all properties.
function btn_height_CreateFcn(hObject, eventdata, handles)
% hObject    handle to btn_height (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,’BackgroundColor’), get(0,’defaultUicontrolBackgroundColor’))
set(hObject,’BackgroundColor’,’white’);
end



function btn_angle_Callback(hObject, eventdata, handles)
% hObject    handle to btn_angle (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of btn_angle as text
% str2double(get(hObject,'String')) returns contents of btn_angle as a double


% --- Executes during object creation, after setting all properties.
function btn_angle_CreateFcn(hObject, eventdata, handles)
% hObject    handle to btn_angle (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in btn_main.
function [lsa, YY,tsa,UU,opd,yy,D,R,t,n,z,Z,y,Y,ASi,y0,Nray,z0,u0]=btn_main_Callback(hObject, eventda
% hObject    handle to btn_main (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
figure
d=(get(handles.uitable1,'data'));

Name= d(:,1);
R   = cell2mat(d(:,2));
n   = cell2mat(d(:,3));
t   = cell2mat(d(:,4));
D   = cell2mat(d(:,5));

for i = 1:length(R)
fprintf('%2d: %6s %8.3f %8.3f %8.3f %8.3f\n',i,Name{i},R(i),n(i),t(i),D(i));
end
t= [t(1:end-1)];
ASi  = 0;

for i=1:length(R)
if strcmp(Name(i),'STO')==1
ASi = uint32(i);
end
end
if ASi==0
disp('Program stoped since Asi = 0');
return
end

Nray = str2double(get(handles.btn_rays,'String'));
y0= str2double(get(handles.btn_height,'String'));
uo = str2double(get(handles.btn_angle,'String'));
u0=uo*pi/180;
N=length(t);
z0=0;
k=1;
if t(1)==inf
t(1)=0;
end
```

```
for i=linspace(-y0,y0,Nray)
y0=i;
[z,y,u] = paraxial(z0,y0,u0,R,t,n);
[Z Y U] = exact(z0,y0,u0,R,t,n);
if  abs(y(1))<=D(ASi)/2
[opd(k) yy(k)] = OPL(z,y,Z,Y,n,ASi);
[lsa(k)  tsa(k) YY(k) UU(k)]=LSA(z,y,u,Y,U,ASi);
[lsa' tsa' YY' UU'];
k=k+1;
end
end
ry(N+1) = y0;
ru(N+1) = 0;
rz(N+1) =z(end);
[rz,ry,ru]= reverseynu(rz,ry,ru,R,t,n);
[rz' ry' ru'];
[EnPx EnPy] = getEnPupil(z,D,t,R,n,ASi);
[ExPx ExPy] = getExPupil(z,D,t,R,n,ASi);

effl = -y(1)/u(end);  %Effective focal length
efl=-ry(end)/ru(1);
if ASi==N
bfl=-y(end-2)/u(end);  % back focal length
else
bfl=-y(end-1)/u(end);   % back focal length
end
if ASi==2
ffl=ry(3)/ru(1);    % front focal length
else
ffl=ry(2)/ru(1);   % front focal length
end
h1=ffl-efl;      % distance front vertex to front plane of thick lens
h2=bfl-effl;     % distance back vertex to seconday plane of thick lens
F=effl/D(ASi);   %f/number
TR=z(end);       %total track
STR=D(ASi)/2;    % STR is stop radius
expx=ExPx-TR;    %exit pupil position
if t(1)==0
NAo=1e-9;
else
NAo=(D(1)/2)/sqrt(t(1)^2+(D(1)/2)^2);
end
NAi=u(end)/4;
z(ASi+1);

predata = fopen('prescriptiondata.txt','w');
if u0==0
fprintf(predata,'\t \tGENERAL LENS DATA  \r\n');
fprintf(predata,'Effective Focal Length   = %4.6f \n',effl);
fprintf(predata,'Back Focal Length        = %4.6f \n',bfl);
fprintf(predata,'Front Focal Length       = %4.6f \n',ffl);
fprintf(predata,'Total Track              = %6.6f \n',TR);
fprintf(predata,'Image Space F/#          = %4.6f \n',F);
fprintf(predata,'Object Space NA          = %4.2e\n',NAo);
fprintf(predata,'Stop Radius              = %4.2f \n',STR);
fprintf(predata,'Entrance Pupil Diameter  = %4.6f \n',abs(EnPy));
fprintf(predata,'Entrance Pupil Position  = %4.6f \n',EnPx);
fprintf(predata,'Exit Pupil Diameter      = %4.6f \n',2*abs(ExPy));
```

```
fprintf(predata,'Exit Pupil Position      = %4.6f \n',expx);
fprintf(predata,'Lens Units               = milimeters   \n');
fprintf(predata,'Field Types              = Angle in degrees \n');
fprintf(predata,'Principal Plane OBJ      = %6.6f \n',h1);
fprintf(predata,'Principal Plane IMJ      = %4.6f \n',h2);
fprintf(predata,'Exact Ray Trace Data \n-------------------------------------------\n');
fprintf(predata,'Z-values\t  Y-values\t U-values\t \n-------------------------------------------\n');
fprintf(predata,'%2.6e \t %2.6e \t %2.6f \n',[Z;Y;U]);
fprintf(predata,'Paraxial ray Trace Data \n-------------------------------------------\n');
fprintf(predata,'z-values\t  y-values\t u-values\t \n-------------------------------------------\n');
fprintf(predata,'%2.6e \t %2.6e \t %2.6f \n',[z;y;u]);
else
fprintf(predata,'If you see true values, you should take field angle as zero \n');
end
fclose(predata);

% --- Executes on button press in btn_layoutthin.
function btn_layoutthin_Callback(hObject, eventdata, handles)
% hObject    handle to btn_layoutthin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
figure
d=(get(handles.uitable1,'data'));
S= d(:,1);
P   = 1./cell2mat(d(:,2));
t   = cell2mat(d(:,3));
D   = cell2mat(d(:,4));

for i = 1:length(P)
fprintf('%2d: %6s %8.3f %8.3f %8.3f\n',i,S{i},P(i),t(i),D(i));
end
t= [t(1:end-1)];
Asi  = 0;

for i=1:length(P)
if strcmp(S(i),'STO')==1 | strcmp(S(i),'*')==1 | strcmp(S(i),'AS')==1
ASi = uint32(i);
end
end
if ASi==0
disp('Program stoped since Asi = 0');
return
end
if t(1)==inf
t(1)=0;
end
Nray = str2double(get(handles.btn_rays,'String'));
y0= str2double(get(handles.btn_height,'String'));
uo = str2double(get(handles.btn_angle,'String'));
u0=uo*pi/180;
z0 =0;
N=length(t);
if t(1)>0
if ASi==2
y0=0;
u1=atan((D(3)/2)/(t(1)+t(2)))-0.009 ;
else
u1=atan((D(2)/2)/t(1))-0.009;
y0=0;
```

```
end

for i=linspace(-u1,u1,Nray)
u1=i;
[z y u] = yuthin(z0,y0,u1,t,P);
[z' y' u']

plottingthin(z,y,D,t,P,ASi);
end
else
for i=linspace(-y0,y0,Nray)
y0=i;
[z y u] = yuthin(z0,y0,u0,t,P);
[z' y' u'];

plottingthin(z,y,D,t,P,ASi);
end
end

[EnPx EnPy] = getEnPupilthin(z,D,t,P,ASi);
[ExPx ExPy] = getExPupilthin(z,D,t,P,ASi);
effl = -y(1)/u(end);
if ASi==N
bfl=-y(end-2)/u(end);    % back focal length
else
bfl=-y(end-1)/u(end);    % back focal length
end
F=effl/D(ASi);    %f/number
TR=z(end);        %total track
STR=D(ASi)/2;     % STR is stop radius
expx=ExPx-TR;     %exit pupil position
if t(1)==0
NAo=1e-9;
else
NAo=(D(1)/2)/sqrt(t(1)^2+(D(1)/2)^2);
end

predatathin = fopen('prescriptiondata.txt','w');
if u0==0
fprintf(predatathin,'\t \tGENERAL LENS DATA      \r\n');
fprintf(predatathin,'Effective Focal Length    = %4.6f \n',effl);
fprintf(predatathin,'Back Focal Length         = %4.6f \n',bfl);
fprintf(predatathin,'Total Track               = %6.6f \n',TR);
fprintf(predatathin,'Image Space F/#           = %4.6f \n',F);
fprintf(predatathin,'Object Space NA           = %4.2e\n',NAo);
fprintf(predatathin,'Stop Radius               = %4.2f \n',STR);
fprintf(predatathin,'Entrance Pupil Diameter   = %4.6f \n',abs(EnPy));
fprintf(predatathin,'Entrance Pupil Position   = %4.6f \n',EnPx);
fprintf(predatathin,'Exit Pupil Diameter       = %4.6f \n',2*abs(ExPy));
fprintf(predatathin,'Exit Pupil Position       = %4.6f \n',expx);
fprintf(predatathin,'Lens Units                = milimeters \n');
fprintf(predatathin,'Field Types               = Angle in degrees \n');
fprintf(predatathin,'Exact Ray Trace Data \n-------------------------------------------\n');
fprintf(predatathin,'z-values\t  y-values\t u-values\t \n-------------------------------------------\
fprintf(predatathin,'%2.6e \t %2.6e \t %2.6f \n',[z;y;u]);
fprintf(predatathin,'Paraxial Ray Trace Data \n-------------------------------------------\n');
fprintf(predatathin,'z-values\t  y-values\t u-values\t \n-------------------------------------------\
fprintf(predatathin,'%2.6e \t %2.6e \t %2.6f \n',[z;y;u]);
else
```

```
fprintf(predatathin,'If you see true values, you should take field angle as zero \n');
end
fclose(predatathin);


% --- Executes on button press in btn_prescriptiondata.
function btn_prescriptiondata_Callback(hObject, eventdata, handles)
% hObject    handle to btn_prescriptiondata (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
system('notepad.exe prescriptiondata.txt');

guidata(hObject, handles);

% --- Executes on button press in btn_layoutthick.
function btn_layoutthick_Callback(hObject, eventdata, handles)
% hObject    handle to btn_layoutthick (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[lsa ,YY,tsa,UU,opd,yy,D,R,t,n,z,Z,y,Y,ASi,y0,Nray,z0,u0]=btn_main_Callback(hObject, eventdata, handl

light = get(handles.btn_lightselection,'value');

if t(1)>0
if ASi==2
y0=0;
u1=atan((D(3)/2)/(t(1)+t(2)))-0.009 ;
else
u1=atan((D(2)/2)/t(1))-0.009;
y0=0;
end

for i=linspace(-u1,u1,Nray)
u1=i;
if light==1
[Z Y U] = exact(z0,y0,u1,R,t,n);
[z,y,u] = paraxial(z0,y0,u1,R,t,n);
elseif light==2
[Z Y U] = exact(z0,y0,u1,R,t,n);
elseif light==3
[z,y,u] = paraxial(z0,y0,u1,R,t,n);
end
plotting(D,R,t,n,z,Z,y,Y,ASi,handles);
end

else
k=1;

for i=linspace(-y0,y0,Nray)
y0=i;
if light==1
[Z Y U] = exact(z0,y0,u0,R,t,n);
[z,y,u] = paraxial(z0,y0,u0,R,t,n);
elseif light==2
[Z Y U] = exact(z0,y0,u0,R,t,n);
elseif light==3
[z,y,u] = paraxial(z0,y0,u0,R,t,n);
end
plotting(D,R,t,n,z,Z,y,Y,ASi,handles);
```

```
end
end


% --- Executes on button press in btn_lsa.
function btn_lsa_Callback(hObject, eventdata, handles)
% hObject    handle to btn_lsa (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[lsa,YY,tsa,UU,opd,yy]=btn_main_Callback(hObject, eventdata, handles);
plot(lsa,YY);
title('Longitudinal Spherical Aberration')
xlabel('LSA');
ylabel('Y(Ray)');

% --- Executes on button press in btn_tsa.
function btn_tsa_Callback(hObject, eventdata, handles)
% hObject    handle to btn_tsa (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[lsa, YY,tsa,UU,opd,yy]=btn_main_Callback(hObject, eventdata, handles);
plot(UU,tsa);
title('Transverse Spherical Aberration')
xlabel('TAN(U)');
ylabel('TSA');

% --- Executes on button press in btn_opd.
function btn_opd_Callback(hObject, eventdata, handles)
% hObject    handle to btn_opd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[lsa, YY,tsa,UU,opd,yy]=btn_main_Callback(hObject, eventdata, handles);
plot(yy,opd);
title('Optical Path Difference')
xlabel('Y(RAY)');
ylabel('OPD');


% --- Executes on button press in btn_addvaluethick.
function btn_addvaluethick_Callback(hObject, eventdata, handles)
% hObject    handle to btn_addvaluethick (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[FileName,pathname] = uigetfile('*.txt') ;
if ~ischar(FileName)
disp('User aborted the dialog');
return;
end
filepath=fullfile(pathname,FileName) ;
set(handles.btn_filename,'String',filepath)
file = fopen(FileName);
txt = textscan(file,'%s %f %f %f %f');
Name= txt{1};
R   = num2cell(txt{2});
n   = num2cell(txt{3});
t   = num2cell(txt{4});
D   = num2cell(txt{5});
```

```
fclose(file);
A=[Name,R,n,t,D];
str2double(set(handles.uitable1,'data',A));
set(handles.uitable1,'ColumnName',{'SURFACETYPE' 'RADIUS' 'INDEX' 'THICKNESS' 'DIAMETER' })
set(handles.uitable1,'CellSelectionCallBack',@(h,e) set(h,'UserData',e));

% --- Executes on button press in btn_addvaluethin.
function btn_addvaluethin_Callback(hObject, eventdata, handles)
% hObject    handle to btn_addvaluethin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Read datga from file
[FileName,pathname] = uigetfile('*.txt') ;
if ~ischar(FileName)
disp('User aborted the dialog');
return;
end
filepath=fullfile(pathname,FileName);
set(handles.btn_filename,'String',filepath);
fid = fopen(FileName);
txtt = textscan(fid,'%s %f %f %f');
S   = txtt{1};    % surfaceType
P   = num2cell(txtt{2}); % Power = 1/f, for OBJ and IMA planes P=0
t   = num2cell(txtt{3});     % Thickness
D   = num2cell(txtt{4});    % Diameter
fclose(fid);
B=[S,P,t,D];
set(handles.uitable1,'ColumnName',{'SURFACETYPE' 'FOCAL LENGTH'  'THICKNES' 'DIAMETER' })

str2double(set(handles.uitable1,'data',B));
set(handles.uitable1,'CellSelectionCallBack',@(h,e) set(h,'UserData',e));

%--- Executes on button press in btn_insert.
function btn_insert_Callback(hObject, eventdata, handles)
set(handles.uitable1,'CellSelectionCallBack',@(h,e) set(h,'UserData',e));
Data = handles.uitable1.Data
ncol=length(Data(1,:));
newrow1=[{'STD'} {[inf]} {[0]} {[20]}];
newrow2=[{'STD'} {[inf]} {[1]} {[0]} {[20]}];
Index = handles.uitable1.UserData;
selected = Index.Indices(1);
if ncol==4
Data = [ Data(1:selected-1,:); newrow1; Data(selected:end,:) ];
else
Data = [ Data(1:selected-1,:); newrow2; Data(selected:end,:) ];
end
set(handles.uitable1,'Data',Data);

% --- Executes on button press in btn_delete.
function btn_delete_Callback(hObject, eventdata, handles)
% hObject    handle to btn_delete (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.uitable1,'CellSelectionCallBack',@(h,e) set(h,'UserData',e));
Data = handles.uitable1.Data;
Index = handles.uitable1.UserData;
Data(Index.Indices(:,1), :) = [];
set(handles.uitable1,'Data',Data);
```

```
guidata(hObject, handles);

% % --- Executes during object deletion, before destroying properties.
function uitable1_DeleteFcn(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function uitable1_CreateFcn(hObject, eventdata, handles)

function btn_filename_Callback(hObject, eventdata, handles)
% hObject    handle to btn_filename (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of btn_filename as text
%        str2double(get(hObject,'String')) returns contents of btn_filename as a double

% --- Executes during object creation, after setting all properties.
function btn_filename_CreateFcn(hObject, eventdata, handles)
% hObject    handle to btn_filename (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in button_save.
function button_save_Callback(hObject, eventdata, handles)
% hObject    handle to button_save (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename, pathname] = uiputfile('*.txt');
if ~ischar(filename)
disp('User aborted the dialog');
return;
end
file=fullfile(pathname,filename);
d=(get(handles.uitable1,'data'));
fileID=fopen(file,'wt');
[nrows,ncols] = size(d);
if ncols==5
formatSpec = '%4s %4.f %6.1f %6.1f %6d\n';
else
formatSpec = '%4s %4.f %6.1f  %6d\n';
end
for row = 1:nrows
fprintf(fileID,formatSpec,d{row,:});
end
set(handles.btn_filename,'String',file)

fclose(fileID);
```