Master of Science Thesis

Optical System Optimization

Using

Genetic Algorithms

By:

Neda Hesam Mahmoudi Nezhad

May 2014

Supervisors:

Dr. Jos Thijsen
Dr. Florian Bociort

Optics Research Group & Theoretical Physics Group
Department of Imaging Physics & Department of Quantum Nano Science
Delft University of Technology- Lorentzweg 1 NL-2628 CJ
Delft. The Netherlands
Optical System Optimization

Using

Genetic Algorithms

Thesis report submitted for the degree of
Master of Science in Applied Physics

Work carried out by:
Neda HesamMahmoudiNezhad
June 2014

Supervisors:
Dr. Jos Thijsse
Dr. Florian Bociort

Faculty of Applied Physics
Delft University of Technology
To My Mohammad and Dorsa,

And

My Dear Parents
Preface

This report is written as a conclusion of my master thesis project which was carried out in the Applied Physics department of Delft University of Technology. The research project was on optical system optimization using Genetic Algorithms (GA).

The report begins with scientific backgrounds on the optics, the GA and the Artificial Bee colony methods including the mathematics theories needed in our research. My own contribution can be mainly found after this introduction, namely in chapter 4, 5 and 6, where I discuss the implementation of the presented theories on the simulation and programming for the optimization in addition to the result analysis and ideas on some possible future works.

My grateful thanks go to both my supervisors, Florian Bociort and Jos Thijssen, for their continuous kind supports and helps, for expertly guiding me through my research and for sharing their great knowledge and experience with me in optical design and Genetic Algorithms fields which were definitely the keys to successfully finishing this project.

I would also like to thank Mohsen Davarynejad for giving me the permission to use his Artificial Bee Colony code in my optimization code and for sharing his knowledge on this field with me, also for his time he kindly devoted for the discussions and questions.

I am also grateful to Ali Mohammadi-Gheidari for giving me constructive and stimulating advices and answering my questions during the length of my master program despite being very busy at his work.

I would also like to express my gratitude towards all my colleagues in the optics group. From Hamed Ahmadpanahi, for running my simulation on his computer when I needed more than one fast P.C. to execute my optimization codes and from Zhe Hou who has always kindly answered my questions no matter how busy he was.

I would also like to thank my kind aunt, i.e. Minoo Hesam, for giving me her invaluable comments on the preface page.

I am in debt to my dear parents, for their always invaluable support, love and blessing which although was from far distance, helped me not to be frustrated in difficult situations.
At the end, I would like to express my sincere thanks to a person who made all my education possible with his unconditional emotional and financial supports, i.e. my dear husband, Mohammad. Thank you Mohammad for all your continuous love, helps and supports! And a special thanks and hug to my daughter, Dorsa, for being patient while I was too busy with my study and was unable to spend time with her.

Neda HesamMahmoudiNezhad

June 2014, Delft
Abstract

The goal of this project is to investigate the performance of the Genetic Algorithms (GA) and the influence of their parameters on optical system optimization. We have developed our own code on optical system optimization using the MatLab GA module to accomplish this task. To evaluate the optical part of our code we checked the outcomes of each step with the commercial lens-design software-package Zemax.

We tested different tuning parameters of the GA. We found that the mutation and the crossover parameters are the most critical parameters. Choosing inappropriate values of these parameters causes the optimization routine to never reach a good result, even by increasing the population size and the number of generations to high numbers.

As an alternative to GA, we studied the Artificial Bee Colony (ABC) method. This is one of the newest methods for global optimization which is claimed by some authors to perform better than GA. We combined an existing ABC code with our optical code. According to the results, for the optical system we consider, we found the GA to be superior over the ABC method.
# Table of Contents

PREFACE ................................................................................................................................................ III

ABSTRACT ............................................................................................................................................... V

TABLE OF CONTENTS ........................................................................................................................... I

1 INTRODUCTION .................................................................................................................................. 1

2 GEOMETRICAL OPTICS BACKGROUND ON OPTICAL SYSTEM DESIGN .............................................. 3

2.1 INTRODUCTION .................................................................................................................................. 3

2.2 OPTICAL SYSTEM DESIGN .................................................................................................................. 3

2.3 ABERRATIONS OF OPTICAL IMAGING SYSTEMS ............................................................................. 4

2.4 REAL RAY TRACING ............................................................................................................................ 5

2.5 OPTICAL SYSTEM PARAMETERS INVOLVED IN THE OPTICAL DESIGN ............................................. 10

2.5.1 Principal Planes ............................................................................................................................. 10

2.5.2 Focal Length .................................................................................................................................. 11

2.5.3 Aperture stop, Entrance & Exit Pupil .............................................................................................. 16

2.5.4 Marginal and Chief Ray ................................................................................................................... 17

2.5.5 Field angles and heights ................................................................................................................... 18

2.5.6 Spot Diagram .................................................................................................................................. 19

3 OPTIMIZATION ...................................................................................................................................... 21

3.1 INTRODUCTION .................................................................................................................................. 21

3.2 LOCAL AND GLOBAL OPTIMIZATION .............................................................................................. 21

The method of Steepest Descent ............................................................................................................. 22

3.3 GENETIC ALGORITHM (GA) .............................................................................................................. 23

3.3.1 Fitness Scaling .................................................................................................................................. 26

3.3.2 Selection .......................................................................................................................................... 26

3.3.3 Reproduction (Crossover and Elitism) .............................................................................................. 27


1 Introduction

Optimization is the process of finding the configuration of a system that is best at performing a specified task. In the present project, we studied optimization for optical imaging systems as a means to arrive at the best possible optical system design, based on particular criteria.

In the past, before having today’s fast computers optical design was very challenging. After the technological revolution resulting in high speed computers, optical design is mostly done using powerful optimization algorithms. Computer optimization techniques play a major role in modern optical system design [1].

The optical imaging system which we mainly worked on consisted of six spherical optical surfaces. This choice was made in order to study particular optimization strategies in a simple system. Extension to more surfaces is straightforward.

There are different optimization techniques available, each having its own merits and drawbacks, depending on the system they are applied to. In general, the optical systems’ merit functions are highly nonlinear [2]. This results in having many local minima in the merit function landscape. To optimize the optical system a global optimizer should be therefore used.

The goal of our project is to analyze the performance of genetic algorithms (GA’s) as a global optimizer for an optical design. The effects of different GA parameters such as the population size, number of generations, the method of crossover and the type of mutation are studied to assess their influence on the performance and to find the best options among them for the defined optical system.

We also include local optimization into the global search in order to improve the result and to see to what extent it influences the GA outcome.

Finally, a new global optimization method, namely the Artificial Bee Colony is introduced and compared with the GA.

To apply optimization to optical systems, knowledge of geometrical optics, such as ray-tracing, optical design parameter calculation and spot size derivation, is required. The second chapter therefore includes the necessary geometrical optics background. To give an idea of how the Genetic
Algorithm (GA) and the local optimizers work, the third chapter introduces these concepts and techniques.

In the Fourth chapter we describe how the discussed theories have been implemented in MatLab to optimize an optical design. The results and the corresponding analysis are presented in chapter five. In the sixth chapter, we summarize the results and present conclusions together with ideas on future possible work.
2 Geometrical Optics Background on Optical System Design

2.1 Introduction

In the following section, we introduce the concept of optical system design. In the section 2.3 we shall have a brief look at the aberrations of optical imaging system. Section 2.4 is devoted to ray-tracing calculations. In the last section, essential concepts involved in optical system design such as principal planes, focal length, aperture stop, marginal & chief ray and field angles & height are introduced in addition to the mathematics needed for their numerical calculation.

2.2 Optical System Design

‘Optical System Design’, is defined as the process of finding the values of the design parameters for which the system has optimal performance, respecting constraints satisfied by these parameters. For an optical lens system, the parameters would include the curvatures of the spherical surfaces, the elements positions in the system and the type of the material used in the system. The constraints may be image-related factors (e.g. magnification, numerical aperture, and field of view) and economical requirements which are generally asked by the consumers such as the size, weight and cost of the system elements.

The quality of imaging in an optical system is determined by the aberrations. The optical system which suffers less from aberration has therefore better imaging quality and by definition is a better optical system.

The task of the optical designer is to find the best optical system within the domain of its pre-defined specifications.

There exists a long-standing practice in optical design. In early days, all optical design steps had to be performed by hand which made the task very time consuming and even sometimes impossible. Today however optical designs are mainly done using computers and based on powerful optimization algorithms tailored to performing the optical design task.
2.3 Aberrations of Optical imaging systems

Ideally a perfect point image is the image in which all the rays coming from one object point focus precisely into one single point.

To find the image of a point, the rays coming from one object point should be traced surface by surface using fundamental rules of reflection and refraction (Snell’s law). The Snell’s law relates the angle of incidence \( i_1 \) of a ray to the angle of refraction \( i_2 \) when the ray passes through an optical surface separating two media with index of refraction \( n_1 \) and \( n_2 \) respectively.

\[
n_1 \sin(i_1) = n_2 \sin(i_2)
\]  

(2.3.1)

In paraxial optics, the rays coming from object points are supposed to travel close to the optical axis. The rays thus intersect the surfaces within small incidence angles \( i_1 \) and \( i_2 \to 0 \). The sine functions can then be Taylor-expanded. In the paraxial optics region the higher order terms in expansion series can be omitted and the sine of angles be approximated by the angle itself. This simplification, which is the basis of paraxial optics, leads to the precise focusing of one object point into one single image point and therefore yields an ideal image.

However, this situation is only valid for the case of optical systems with small field and aperture\(^1\) and cannot be generalized for all circumstances. In real cases, where field and aperture may be large, the rays which are travelling with large angles and heights cannot be traced using paraxial optics formulas and deviations from ideal image formation will arise. These deviations are called monochromatic aberrations. The main five monochromatic aberrations (which are also known as third order aberrations) are the spherical aberrations, coma, astigmatism, field curvature and distortion [26].

In the above discussion it is assumed that the coming ray is monochromatic. Considering the fact that the refractive index is a function of wavelength the chromatic aberration should be incorporated into the system’s aberrations as well.

---

\(^1\) The definition of field and aperture can be found in section 2.5.5.
An ideal imaging system, by definition, is the system which does not suffer from any kind of aberration. In practice, the aberrations cannot be eliminated completely from the system. However, by changing the parameters of the system they can be balanced, controlled and made small.

In the figures below, a system without having aberration, performing perfect point image (fig2-1) and a system which suffers from spherical aberration (fig2-2) are shown.

![Figure 2-1: An optical system without aberration performing ideal point imaging.](image1)

![Figure 2-2: An optical system which suffers from spherical aberration. Rays which hit the lens in different heights will have different focusing places and strike the image plane at different heights. The image is thus a wide spread than a single point.](image2)

2.4 **Real Ray Tracing**

In this section we will present the equations and the mathematics behind the real ray tracing method which enables us to trace any ray passing through optical systems. The optical systems which we consider are symmetrical optical systems consisting of a set of spherical surfaces with their center of curvature on the optical axis.

In this analysis, we neglect diffraction. The tracing of the rays can thus be accomplished merely by using geometrical optics theories and goniometry.
Ray tracing can be done by a moving coordinate system and by implementing the basic assumptions of the geometrical optics:

- Rays travel along straight lines with the constant velocity in homogenous media.
- At the interface of two homogenous media having different refractive indices, rays change their direction according to Snell’s law.

The real ray tracing can be iteratively performed by pursuing the following steps [3, 6]:

I. Start from a point placed at the first surface and trace the ray from that surface to the tangent plane at the vertex of the next surface.

II. Transfer the ray from the tangent plane to the actual optical surface and find the intersection point of the ray with that surface.

III. Find the new direction cosine for the refracted ray from the optical surface by applying Snell’s law of refraction.

IV. Transfer to the next surface by repeating steps I to III.

The Sign Convention

The curvature of the spherical surface of Radius R is defined as \( c = \frac{1}{R} \). We use the sign convention where \( c > 0 \) when the center of the curvature is placed to the right of the spherical surface and \( c < 0 \) for the other case, i.e. when the curvature is to the left of the surface (shown in figure 2-3) [12]. Below we discuss the ray traveling steps in more detail:

\[ \begin{align*}
R > 0 & \quad & C > 0 \\
R < 0 & \quad & C < 0
\end{align*} \]

\( \text{Centre of curvature} \)

\( \text{Centre of curvature} \)

**Figure 2-3: The Sign Convention**

I. Transfer between a spherical surface and the tangent plane of the next spherical surface:

The ray tracing can be started from an initial point \( P_{1} \), placed at the first spherical surface (shown as surface \( j_{1} \) in the figure 2-4), having position \((x_{1}, y_{1}, z_{1})\) with the cosine directions \((L, M, N)\). The coming ray hits the tangent plane (surface \( j_{0} \) in the figure 2-4) at \( P_{0} = (x_{0}, y_{0}, z_{0}) \). The position of this
point can be mathematically derived using the direction cosines through the equations (2.4.1)-
(2.4.3):

\[ x_0 = x_1 + \frac{L}{N} (d - z_1) \]  \hspace{1cm} (2.4.1)

\[ y_0 = y_1 + \frac{M}{N} (d - z_1) \]  \hspace{1cm} (2.4.2)

\[ z_0 = d + z_1 \]  \hspace{1cm} (2.4.3)

where, \( d \) is the distance between the origins of the coordinate systems of the normal plane to the
surface \( j_1 \) and the tangent plane \( j \)

**Figure 2-4:** Ray tracing from the surface \( j_1 \) to the tangent plane \( j_0 \) and from the tangent plane \( j_0 \) to the next
surface \( j \). (after Geometrical Optics and Lens Design Lecture Notes, Imperial College [3]).

II. Transfer between the tangent plane of a spherical surface to its related spherical surface:

Continuing from point \( P_0 = (x_0, y_0, z_0) \), placed at the tangent plane of the second spherical surface
(shown as \( j_0 \) in figure 2-4), the ray would travel the distance \( \Delta \) until it strikes its corresponding
spherical surface (surface \( j \) in figure 2-4) at the intersection point \( P = (x, y, z) \). The position of this new
transformed point can be calculated using:
\[ x = x_0 + L \Delta \]  \hspace{1cm} (2.4.4)

\[ y = y_0 + M \Delta \]  \hspace{1cm} (2.4.5)

\[ z = z_0 + N \Delta \]  \hspace{1cm} (2.4.6)

In these equations, the parameters \( x_0 \), \( y_0 \) and \( z_0 \) are known parameters which have been already determined using equations (2.4.1)-(2.4.3), while \( \Delta \) is a yet unknown parameter which should be derived. To find this parameter, we use the equation which describes the spherical surface:

\[ x^2 + y^2 + (z-R)^2 = R^2 \]  \hspace{1cm} (2.4.7)

By substituting the equation (2.4.6) into equation (2.4.7), after mathematical simplification, we obtain:

\[ c(\Delta^2) - 2(N - c(Lx_0 + My_0))\Delta + c(x_0^2 + y_0^2) = 0 \]  \hspace{1cm} (2.4.8)

where \( F \) and \( G \) are defined as below:

\[ F = c(x_0^2 + y_0^2) \]  \hspace{1cm} (2.4.9)

\[ G = (N - c(Lx_0 + My_0)) \]  \hspace{1cm} (2.4.10)

Solving the quadratic equation (2.4.8), leads to the two solutions for \( \Delta \):

\[ \Delta = \frac{G \pm (G^2 - cF)^{1/2}}{c} \]  \hspace{1cm} (2.4.11)

For very small \( (x_0^2 + y_0^2) \), the value of \( \Delta \) should approach tends to zero. This condition is satisfied by the minus sign in equation (2.4.11) which thus determines the solution for \( \Delta \) to be:

\[ \Delta = \frac{G - (G^2 - cF)^{1/2}}{c} \]  \hspace{1cm} (2.4.12)

With all parameters known, the equations (2.4.4)-(2.4.6) can be solved to find the position of the intersection point of the incoming ray with second spherical surface.
III. Calculate the direction for the ray after its refraction from second spherical surface:

Assume the ray \( P_2 P \) (having direction cosines of \( L, M \) and \( N \) and a unit vector \( \hat{r} \) (i.e. \( \hat{r} = (L, M, N) \)) travels in the medium with refractive index of \( n \). The ray, intersecting the second spherical boundary at point \( P(x, y, z) \), will be refracted due to the change in its traveling medium and will have a new unit vector \( \hat{r}' = (L', M', N') \) (shown in figure 2-5).

\[
\hat{r} = \cos i \hat{a} + \sin i \hat{b}
\]

(2.4.13)

\[
\hat{r}' = \cos i' \hat{a} + \sin i' \hat{b}
\]

(2.4.14)

By applying the Snell’s law on coming ray (\( n \sin i = n' \sin i' \)), the refracted ray can be determined in terms of coming ray and normal unit vector as:

\[
n' \hat{r}' = n \hat{a} + (n' \cos i' - n \cos i) \hat{a}
\]

(2.4.15)

To calculate the refracted direction cosines ((\( L', M', N' \) = \( \hat{r}' \))), we need to calculate the terms \( \cos i \), \( \cos i' \) and the direction of \( \hat{a} \).

The function which describes a spherical surface with the radius of \( R \) has the form:

\[
f(x, y, z) = R^2 - x^2 - y^2 - (z - R)^2 = 0
\]

(2.4.16)
The normal unit vector \( \hat{a} \) to this surface can be determined by taking the gradient of this function. After some mathematical simplification it gives

\[
\hat{a} = \frac{\nabla f}{|\nabla f|} = \left(\frac{-x}{R} \hat{\imath}, \frac{-y}{R} \hat{j}, \frac{1 - z}{R} \hat{k}\right) = (cx, cy, (1 - cz))
\] (2.4.17)

Knowing the direction of normal unit vector, the terms \( \cos i \) and \( \cos i' \) can be determined:

\[
\cos i = \hat{r} \cdot \hat{a} = (L, M, N) \cdot (cx, cy, (1 - cz)) = (-cxL - cyM + (1 - cz)N)
\] (2.4.18)

where \( x, y \) and \( z \) can be taken from equations (2.4.4)-(2.4.6).

Using Snell’s law \( \cos i' \) can also be derived as below:

\[
\cos i' = \sqrt{(n'^2 - n^2 (1 - \cos^2 i))}
\] (2.4.19)

Implementing all these results in equation (2.4.15), the new direction cosines of \( L', M', N' \) can be calculated:

\[
n'L' = nL - x B
\] (2.4.20)

\[
n'M' = nM - y B
\] (2.4.21)

\[
n'N' = nN - z B + B/c
\] (2.4.22)

where \( B = (n' \cos i' - n \cos i) \)

### 2.5 Optical System Parameters Involved in the Optical Design

There are essential concepts involved in optical system design, which are used in our optimization design programming such as field and aperture, principal planes, marginal and chief ray, effective focal length (efl) and back focal length (bfl). Due to their importance we devoted this section to briefly introduce these parameters.

#### 2.5.1 Principal Planes

In reality, any lens has a thickness. For zero thickness the simple thin lens equations and rules can be applied for defining the system parameters and tracing the rays (illustrated in figure 2-6). However, in real ray tracing, the thickness of lenses is a main factor which could not be discarded and the thin lens approximation rules are no longer valid. For this case, to simplify the representation of system parameters and tracing the rays, similar to that of thin lens approximation, principal planes are introduced. The principal planes are two hypothetical planes such that all refractions are supposed to
be occurred there. The incoming rays are thus assumed not to face refraction between these two planes and thus to cross both planes in the same height with respect to the optical axis. These planes are shown in figure 2-7.

![Figure 2-6: Thin lens ray-tracing. All rays are supposed to intersect the rear and front planes of lens with the same height with respect to the optical axis.](image)

![Figure 2-7: Ray tracing in an Optical design. P1 and P2 are two hypothetical planes (called the first and second principal planes) on which all rays assumed to intersect with the same height with respect to the optical axis.](image)

### 2.5.2 Focal Length

**Effective focal length (efl) and back focal length (bfl)**

The focal length of optical systems, is defined based on the paraxial approximation rules (i.e. assuming the rays travel at small angles with respect to the optical axis ($\sin \alpha \cong \alpha$)) [4].

The back focal point of an optical system is, by definition, the point to which all rays parallel to the optical axis in the object space, converge. The front focal point has the reverse property, namely, all rays passing through this point would travel parallel to the optical axis after refraction from optical system. The planes perpendiculars to the optical axis at these points are called the back focal and front focal planes, respectively [23].
The back focal length (bfl) is defined as the distance between the back focal plane and the vertex of the last lens surface of the system. The effective focal length (efl) is the distance of the back focal plane, measured from the second principal plane (These properties and definitions are illustrated in figure 2-8 [24]).

Figure 2-8: focal length definitions of optical systems. F is the paraxial back focal point of the optical system. P1 and P2 are the first and second principal points. The planes perpendicular to the optical axis at P1 and P2 (shown by dashed lines) are the so-called first and second principal planes respectively. BFL is, by definition, the distance between the back focal point to the vertex of last lens surface. FFL is the distance from the front focal plane to the vertex of first lens surface. EFL is defined as distance from the back focal point when measured from the second principal plane.

efl and bfl calculation

Implementing the definitions of the effective focal length and back focal length, these parameters can be formulated such that to be numerically calculable. In the paraxial approximation, according to the above discussion, a ray with height 1 coming from infinity while travelling parallel to the optical axis, strikes the second principal plane at the same height 1. The ray crosses the optical axis at the back focal plane. Assume the ray refracted from the last optical surface with an angle $\alpha'$ having the height $y_n$ (illustrated in figure 2-9), the efl and bfl parameters could be formulated using trigonometry as below equations [3]:

\[
\text{in the triangle } DP_2 F: \tan(-\alpha') = \frac{DP_2}{FP_2} \quad (2.5.1)
\]

paraxial approximation $\rightarrow \tan(-\alpha') \equiv -\alpha'$ \quad (2.5.2)

efl definition $\rightarrow$ efl $= FP_2$ \quad (2.5.3)
→ \( efl = -1/\alpha' \) \hspace{1cm} (2.5.4)

in the triangle CEF: \( \tan(-\alpha') = \frac{CE}{EF} \) \hspace{1cm} (2.5.5)

\( bfl \) definition \( \rightarrow BF = bfl \) \hspace{1cm} (2.5.6)

paraxial approximation \( \rightarrow \tan(-\alpha') \equiv -\alpha' \) \& \( BF \equiv EF \rightarrow bfl = CE/EF \) \hspace{1cm} (2.5.7)

→ \( bfl = -\gamma_n/\alpha' \) \hspace{1cm} (2.5.8)

Figure 2-9: finding efl and bfl of an optical system.

There is also another way based on a matrix method which can be used to calculate the bfl [5].

In the matrix method, a ray is represented by a two-component column-vector (denoted here by \( v \)) consists of height and propagation angle of the ray:

\[ v = \begin{pmatrix} n \mu \\ y \end{pmatrix} \] \hspace{1cm} (2.5.9)

where \( y \) is the height of the ray and \( \mu \) is the angle which the ray makes with the positive optical axis.

The ray passes through the homogenous media and the refraction elements. The transformations of the ray’s vector through these media are calculated using the transfer and refraction matrices:

For the transfer:

\[ M_t = \begin{pmatrix} 1 & 0 \\ d/n & 1 \end{pmatrix} \] \hspace{1cm} (2.5.10)
\[
\begin{pmatrix}
    n_{k+1} & u_{k+1} \\
    y_{k+1}
\end{pmatrix} = M_t \begin{pmatrix}
    n_k & u_k \\
    y_k
\end{pmatrix} \tag{2.5.11}
\]

For the refraction:

\[
M_r = \begin{pmatrix}
    1 & c(n_k - n_{k+1}) \\
    0 & 1
\end{pmatrix} \tag{2.5.12}
\]

\[
\begin{pmatrix}
    n_{k+1} & u_{k+1} \\
    y_{k+1}
\end{pmatrix} = M_r \begin{pmatrix}
    n_k & u_k \\
    y_k
\end{pmatrix} \tag{2.5.13}
\]

where \(d\) is the distance which ray travels through a homogenous medium with refractive index of \(n\). The indices \(k\) and \(k + 1\) denotes the parameters before and after the transformation, respectively.

Considering an optical system with \(k\) different media, the transformation of the ray vectors through the whole system, starting from the object space and ending at the image plane is then written as:

\[
\begin{pmatrix}
    n_i & u_i \\
    y_i
\end{pmatrix} = G \begin{pmatrix}
    n_0 & u_0 \\
    y_0
\end{pmatrix} \tag{2.5.14}
\]

\[
G = M_{t,1} M_{tot} M_{t,0} \tag{2.5.15}
\]

\[
M_{tot} = M_{r,k} M_{t,k-1} M_{r,k-1} \ldots M_{r,2} M_{t,1} M_{r,1} \tag{2.5.16}
\]

\[
M_{t,i} = \begin{pmatrix}
    1 & 0 \\
    d_i/n_i & 1
\end{pmatrix} \tag{2.5.17}
\]

\[
M_{t,o} = \begin{pmatrix}
    1 & 0 \\
    d_o/n_o & 1
\end{pmatrix} \tag{2.5.18}
\]

Indices ‘\(i\)’ and ‘\(o\)’ indicates the parameters related to the object and image space respectively. \(d_o\) is the distance from the object plane to the first spherical surface and \(d_i\) is the distance from the last spherical surface to the image plane (a scheme of matrix method configuration is represented in the figure 2-10).

Replacing equations (2.5.17) and (2.5.18) in equation (2.5.15) gives

\[
G = \begin{pmatrix}
    1 & 0 \\
    d_i/n_i & 1
\end{pmatrix} \begin{pmatrix}
    M_{tot11} & M_{tot12} \\
    M_{tot21} & M_{tot22}
\end{pmatrix} \begin{pmatrix}
    1 & 0 \\
    d_o/n_o & 1
\end{pmatrix} \tag{2.5.20}
\]

writing \(G\) matrix as its elements

\[
G = \begin{pmatrix}
    G_{11} & G_{12} \\
    G_{21} & G_{22}
\end{pmatrix} \tag{2.5.19}
\]
and using matrices multiplication gets the G matrix elements as

\[ G_{11} = M_{tot11} + M_{tot12} \frac{d_o}{n_0} \]  
\[ G_{12} = M_{tot12} \]  
\[ G_{22} = M_{tot22} + M_{tot12} \frac{d_i}{n_i} \]  
\[ G_{21} = M_{tot12} \]  

\[ (2.5.21) \]  
\[ (2.5.22) \]  
\[ (2.5.23) \]  
\[ (2.5.24) \]

**Figure 2.10:** A scheme of Matrix method configuration. The optical system includes k spherical surfaces. \( d_o \)\( ( d_i ) \) is the distance from the first (last) spherical surface to the object (image) plane. \( Mr, (i=1: k) \), are the refraction matrices related to k spherical surfaces. \( Mt, (i=1:k-1) \) are the transformation matrices related to k-1 intervals between k spherical surfaces.

Using the equations (2.5.14)-(2.5.19), the height of the ray in the image plane is then:

\[ y_i = G_{21} n_o u_o + G_{22} y_o \]  
\[ (2.5.25) \]

For sharp imaging the height in the image plane should be independent of the rays\' propagation angle. Then in paraxial optics, transverse magnification is constant.

Applying these conditions in equation (2.5.25) gives:
\[ G_{22} = \beta = y_i/y_o \]  \hspace{2cm} (2.5.26)

\[ G_{21} = 0 \]  \hspace{2cm} (2.5.27)

where \( \beta \) is the transverse magnification.

The matrix \( G \) is then

\[ G = \begin{pmatrix} \beta^{-1} & M_{tot12} \\ 0 & \beta \end{pmatrix} \]  \hspace{2cm} (2.5.28)

Using this equation and equations (2.5.21)-(2.5.24), one gets

\[ \beta^{-1} = M_{tot11} + M_{tot12} d_o/n_o \]  \hspace{2cm} (2.5.29)

\[ \beta = M_{tot22} + M_{tot12} d_i/n_i \]  \hspace{2cm} (2.5.30)

Which results

\[ d_i = -\frac{n_i}{M_{tot12}} \left( M_{tot22} - 1/ \left( M_{tot11} + M_{tot12} \frac{d_o}{n_o} \right) \right) \]  \hspace{2cm} (2.5.31)

According to the definition of the back focal length (i.e. \( bfl = \) the distance between the last spherical surface and the back focal plane), \( bfl \) is the parameter \( d_i \) when \( d_o \) is infinite.

\[ bfl = -\frac{n_i}{M_{tot12}} \left( M_{tot22} - 1/ \left( M_{tot11} + M_{tot12} \frac{d_o}{n_o} \right) \right) \text{ where } d_o = \infty \]  \hspace{2cm} (2.5.32)

\[ \rightarrow bfl = -\frac{n_i}{M_{tot12}} (M_{tot22}) \]  \hspace{2cm} (2.5.33)

### 2.5.3 Aperture stop, Entrance & Exit Pupil

The aperture stop is an important element of the optical system which limits the amount of light that can enter to the system. The aperture stop can be an opening or a lens edge, where the solid angle of rays coming from an on-axis object point is physically restricted. Based on the optical system requirements it can be placed anywhere within the system (two cases are illustrated in figure 2.11).

In the top figure, the incoming rays are limited by the first opening, which is then the aperture stop of the system. In the figure shown at the bottom, the restriction of the incoming rays is done by the first lens which therefore caused the first lens to be the aperture stop of the system.
Figure 2-11: Aperture stop in two different cases. In this figure, at the top, the aperture stop is the hole which is placed before the first lens, while, for the optical system shown in the bottom, the aperture stop is the first lens.

The entrance pupil is defined as the image of the aperture stop as observed from the object side. It is formed by the optical elements placed before the aperture stop. Depending on the place of aperture stop, this image can be virtual or real. The exit pupil is the image of the aperture stop when it is seen with the optics beyond it from the image side. This image also can be a real or a virtual image.

2.5.4 Marginal and Chief Ray

The image of an object in paraxial approximation can be found using only two rays. One possible set of rays is the marginal and the chief ray. These rays have substantial importance in the ray tracing as they are associated with the aperture and field\(^2\).

The marginal ray refers to the ray which comes from the axial object point while it hits the edge of the entrance/exit pupils in its path to the paraxial image plane. It determines the entrance and exit pupil sizes as well as the image location. The chief ray is the ray which is transmitted from an off-axis object point (i.e. at the edge of the field) to the center of entrance/exit pupils to strikes the image plane. The chief ray determines the image size and the location of the entrance and exit pupils [5].

These relationships are illustrated in the figure 2-12.

---

\(^2\) The definition of the field is presented on the section 2.5.5.
Figure 2.12: Illustration of marginal & chief ray relationships with entrance and exit pupils.

2.5.5 Field angles and heights

The ‘Field’ or ‘Field of View’ (abbreviated by FOV), is used to specify the extent of the object which is observable by the optical system. The FOV for an optical system can be the angle formed by the ray coming from the two edges of object to the center of entrance pupil of the system. This is shown in figure 2-13 for a multi lens system.

However, for the sake of simplicity in our later data comparison with Zemax, we use the same terms to specify the object characteristic in the system, namely ‘field heights’ and ‘field angles’, as Zemax does[7].

‘Field height’ refers to the height of object in the object space. This term is used when the object is placed at a finite distance with respect to the optical system. ‘Field angles’ can be used either when the object is at infinity or at a finite distance far from optical system. This term is defined as the angle equal to the half field of view (HFOV).
Figure 2.13: Field of view for a composite optical system. Field of view can be determined by the angle formed by two rays coming from the edges of object to the center of entrance pupil (the pair of chief rays originated from two object top points) [8].

2.5.6 Spot Diagram

Rays coming from an object point intersect the paraxial image plane at some points, which can be found using the real ray tracing methods. As it is discussed before, in non-ideal optical systems the image of this point is not a single point, but a spread spot. The collection of the intersection points of the traced rays from an object point with the paraxial image plane is called ‘spot diagram’ [21] (See figure 2-14).
Optical System Optimization Using Genetic Algorithm

Figure 2-14: A schematic illustration for spot diagram. A set of rays from an object point is numerically traced until they cross the paraxial image plane. The intersection points which do not form a single point, but instead a spread spot are the spot diagram. Zoomed area shows an example of a spot diagram. The spot diagram formed by the rays coming form an on axis object point at infinity passed through an optimized optical system (taken from our Matlab optimization code).

The shape of the spot diagram can indicate the type of aberrations present in the optical system. The size of this diagram, which is called ‘spot size’, is a measure of the amount of aberration.
3 Optimization

3.1 Introduction

Optimization is a collection of mathematical principles and techniques used to find the optimum solution of quantitative problems. In many technological areas, there are quantitative problems which require optimization. Thus, optimization is applied not only in optics but also in other disciplines such as business, engineering, biology, economics etc.

The technique consists of constructing a mathematical function (called objective function, error function or merit function) which represents the performance of the design or process. This mathematical function should be calculated at each step by using the input values, within a defined domain. The optimum point relates to the maximum or minimum (depends on our goal) value of the merit function.

There are two types of optimization namely, local and global optimization. To give an idea on how they work, in the following sections, we briefly describe the overall approach of local and global optimization. First a simple technique of local optimization (i.e. Steepest Descent method) is described. The section 3.3 is then devoted to explain in detail the Genetic Algorithm as a global optimizer. There are many parameters involved in the GA. The most important ones are presented at sections 3.3.1-3.3.4. The Artificial Bee Colony (ABC) is another type of global optimization method. This method is briefly described in section 3.4.

3.2 Local and Global Optimization

Considering a function with several minima, local minima are defined as points where the function values are smaller than the function values of the nearby points but can be higher than function values of further points. In contrast, the global minimum is the point where the function has the minimum value of all arguments in the domain of the function (figure 3-1).
Local optimization is a type of optimization which starts from an initial point and reaches the local minimum in the basin of attraction of that point. In global optimization, more basins of attraction are explored to find the global minimum. Global optimization thus finds the global minimum while local optimization finds the local minimum [9].

**The method of Steepest Descent**

A simple local optimization method is the method of steepest descent. The steepest descent method (also called gradient descent) is applied to a function $F(X)$ to find the local optimum starting from a point $X_0$ if the function is defined and differentiable in the neighborhood of this point. The algorithm moves from point $X_0$ by taking the steps $\lambda_n$ in the direction of the negative (positive) gradient of the function. The process is then repeated until it reaches the minimum (maximum) of the function [20].

$$X_{n+1} = X_n - \lambda_n \nabla F(X_n), \ n \geq 0$$  \hspace{1cm} (3.2.2)

where $X_n$ is the local minimum point of the function. $\lambda_n$ is found by a one-dimensional minimization (maximization) along the direction of the gradients.

We obtain a sequence of the points $X_0$ to $X_n$ for local minimization:

$$F(X_0) \geq F(X_1) \geq F(X_2) \geq ... F(X_n)$$  \hspace{1cm} (3.2.1)

This process is illustrated in figure 3-2. By starting from two different initial points, the method found two different local minima.
Optical System Optimization Using Genetic Algorithm

3.3 Genetic Algorithm (GA)

According to Darwin's theory, in the natural evolution process, the creatures have improved their compatibility characteristics with the environment through natural selection among their genetic variations over the course of many generations. Therefore new generations tend to be more successful in their survival than previous ones.

Genetic Algorithm (GA) is a heuristic search technique with a logic which mimics this natural evolution process and can therefore be used as an optimization tool for real complex problems in science and technology.

The algorithm starts with a set of systems (represented by chromosomes in Darwin's Theory) called initial population. The initial population can be a randomly generated state denoted by $P(x_1,...,x_n)$, where the ‘$x$’ are vectors representing the chromosomes of different individual systems.

Figure 3-2: Illustration of gradient descent method. The method is applied to two different starting points ending in different local minima. After [10].
This initial population gradually evolves towards set of chromosomes which better satisfy the conditions of the desired problem. The objective or merit function is called \( f(x) \), where ‘\( x \)’ corresponds to the chromosomes on which the evolutionary processes are applied. The evolution leads to a new population of chromosomes \( P_{i+1}(x_1,..,x_n) \) which concentrates the population near the optima of the objective function. The chromosomes are in practice often described by a set of binary bit-string representations.

The next step is to create a new population. For the sake of simplicity we only present the simplest form of genetic algorithm which involves three operators: Crossover, Mutation and Selection operators to apply the main procedures in the system and Elitism as an additional operator which could be added to the process to enhance the result of the GA [22]. This so-called Simple Genetic Algorithm (SGA) is explained by a Schematic Algorithm below:
There are different methods of scaling, reproduction and mutation. Here we introduced those which are used in the 2013 version of Matlab. The implementation methods are taken from Matlab manual toolbox [11].
3.3.1 Fitness Scaling

The objective function (fitness function in GA) determines the ‘quality’ of the individuals. For each individual the mathematical fitness function returns a raw fitness score. The raw scores are converted to values in a range which is suitable for the selection step. This conversion step (the so-called fitness scaling step) can be performed through different methods such as, ‘rank’, ‘proportional’, ‘top’ and ‘shift linear scaling’.

In the ‘rank fitness scaling’ method, the scaling function scales the individual’s raw scores based on their rank (i.e. the individual position in the stored scores) instead of their real scores. The function returns a score proportional to $1/\sqrt{r}$ for an individual with the rank $r$. The best individual therefore gets the scaled score proportional to 1. The ‘proportional scaling’ method, gives a scaled score to the individual proportional to their raw fitness score. The ‘top scaling’ function, gives all the top individuals an equal score. The method needs therefore an additional step to first select the top individuals. This step is performed specifying a so-called ‘Quantity’ field, which determines the numbers of individuals that are assigned as top individuals. In the ‘shift linear scaling’, the scaling function converts the raw scores so that the scaled fitness value is equal to a constant multiplied by the average score.

3.3.2 Selection

The selection operator uses the scaled fitness values to choose the fitter individuals in the population as the parents to generate offspring for the next generation. To reach this goal, the members with the lower scaled fitness values should get greater probability to be excluded from the next generation, while the reverse holds for those with higher fitness, namely, individual having higher scaled fitness values would be assigned a higher probability of selection to enter to the next generation.

There are many different selection techniques such as “roulette”, “stochastic uniform”, “remainder”, “uniform”, and “tournament” which can be determined based on the defined problem.

To give an idea of how this step is applied on the members, we briefly explain the first method as an example of the selection method.

The “roulette” selection method consists of giving the members the opportunity to be selected to enter to the next generation proportional to their scaled fitness values. In this method, each
individual would get a slice of a circular roulette wheel with an area which is equal to their scaled fitness value [25]. The roulette wheel then is spun N times, where N is the number of individuals in each population. The wheel rotation stops on one edge-shaped slice and the corresponding individual under that wheel’s marker will be selected as the parent to enter to the next generation.

3.3.3 Reproduction (Crossover and Elitism)

In the next step, the chosen parents entered from the selection step should create their offspring. The Reproduction operator is the process determines how the genetic algorithm creates offspring from the selected parents for the next generation. The reproduction can be performed using crossover and elitism operations. The fraction of the next generation members that are produced by crossover or by elitism can be determined specifically by the GA user.

3.3.3.1 Crossover

The power of Genetic Algorithm arises primarily from the two operators, namely crossover and mutation. These operations serve as mechanism to introduce diversity in the population

Crossover is the operator which combines two individuals (parents) to create a new individual (offspring). The idea behind using this operator is to create new individuals with the better characteristics with respect to their parents by inheriting the best characteristics from each parent. Crossover is applied on the population during evolution based on the user definable crossover probability [12]. There are different methods of crossover implementation such as “single-point crossover”, “two-point crossover”, ”scattered crossover”, “intermediate crossover”, “uniform crossover”, “arithmetic crossover” and “heuristic crossover”.

To explain different methods we consider two member of population, for instance p1 and p2, to be the selected parents for mating in crossover process. The individuals are represented as strands of elements which could be bits or real numbers. Presuming a general case of strands p1= [a b c d e f g h i] and p2 = [1 2 3 4 5 6 7 8 9] to be two selected parents, different types of crossover are explained below. [11]

In the “single point crossover”, one crossover point is randomly selected by the crossover operator. All elements beyond this point are then swapped between two parents to produce new individuals (i.e. offspring). As an example, if the crossover point is chosen to be placed after the third element, the offspring p’1 and p’2 which is returned by the single-point crossover function would be:
Offspring 1: \( p'1 = [a \ b \ c \ 4 \ 5 \ 6 \ 7 \ 8 \ 9] \) and Offspring 2: \( p'2 = [1 \ 2 \ 3 \ d \ e \ f \ g \ h \ i] \).

The “two-point crossover” operator, however, randomly selects two positions as the crossover indicator points instead of one. Every element between these two positions is then exchanged to create the new offspring. For instance if the two indicator points are selected to be placed after the second and fifth elements, the children would have the form of:

Offspring 1: \( p'1 = [a \ b \ 3 \ 4 \ 5 \ f \ g \ h \ i] \) and Offspring 2: \( p'2 = [1 \ 2 \ c \ d \ e \ 6 \ 7 \ 8 \ 9] \).

The ‘Scattered’ Crossover function first randomly creates a binary indicator vector. Based on this vector, then it produces the children. The children are formed such that for each parent the elements are stayed unchanged where the indicator vector has a 1 element and changed with the other parent’s element where it has a 0 element. Assume the indicator vector to be \([0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0]\), the offspring would be: Offspring 1: \( p'1 = [a \ b \ c \ 4 \ e \ 6 \ 7 \ 8 \ i] \) and Offspring 2: \( p'2 = [1 \ 2 \ c \ d \ 5 \ f \ g \ h \ 9] \).

For the ‘Intermediate’ crossover function, there is a so-called ‘Ratio’ factor involved. This ‘Ratio’, which can be introduced as a scalar or a row vector of number of variables length, specifies the weights of the individuals. The operator then creates the offspring taking a weighted average of the parents based on this ratio and the following formula:

\[
\text{Offspring 1: } p'1 = p1 + \text{rand} \times \text{Ratio} \times (p2 - p1)
\]

The Heuristic Crossover operator first uses the parents’ fitness values to evaluate them and indicate the best and worst among them. It then returns one offspring based on the following equation:

\[
\text{Offspring 1 = Best Parent + Ratio} \times (\text{Better Parent – Worst parent})
\]

The created child therefore would be placed between the ranges of two parents values. The distance of the child from the best and the worst parent can be determined by the factor ‘ratio’ which is a random number that introduced by the operator.

The ‘Arithmetic’ crossover function, return the offspring by taking the average of two parents. In this way, the offspring are always placed in the regions limited to the upper and lower pre-defined bounds.

### 3.3.3.2 Elitism

This method can be implemented in the GA in addition to the main selection operator to enforce the Genetic Algorithm selection process to keep the best individuals at each generation. If this method is not added to the system, there is a risk to lose some of the best individuals, as they could be
destroyed through the crossover and mutation process. This operator eliminates this danger and consequently significantly improves the performance of GA.

### 3.3.4 Mutation

The mutation plays an important role in the GA. This procedure is applied in the algorithm in addition to the crossover process to give each member the opportunity to reach the full space of possible solutions. Without introducing this step to the GA body, there is a risk that some solutions can never be generated by just implementing the crossover step. As a simple example, consider an element which has the same value in all individuals of the initial population. Just using cross-over and reproduction, this element will always have the same value for all individuals through all generations. Since in such case, the members can be trapped in a limited area without ever escaping from that region, therefore the GA might be able just to find a local optimum and unable to reach the main goal point which is to reach the global optimum. Applying mutation eliminates this defect by enhancing the diversity of the population and giving the members the chance to escape from a trapped area.

There are different ways which mutation can be implemented such as ‘Gaussian’, ‘Uniform’ and ‘Adaptive Feasible’.

In the Gaussian mutation method, there are two main factors (i.e. ‘scale’ and ‘shrink’) which control the mutation procedure. The ‘scale’ parameter determines the amount of deviation introduced at the first generation, while the ‘shrink’ parameter specifies how this deviation would be diminished in the course of different generations. The operator adds a random number selected from a Gaussian distribution to each element of the parents. The Gaussian distribution centered at zero with the width defined based on the ‘scale’ parameter which gets smaller by the ‘shrink’ factor as the generations proceed.

The ‘Uniform Mutation’ applies the mutation in the elements of a generation with a probability determined by the so-called ‘Rate’ factor. The selected elements then will be replaced by random numbers which are selected uniformly within the pre-defined range for each element.

In adaptive feasible, the mutation occurs for each element with a probability proportional to the inverse of its fitness value. This means the elements which have less fitness values are more probable to be mutated compare to those with higher fitness values. In this way, the better members would
be less mutated and thus less be discarded. The direction of implementation of mutation is therefore said to be in the direction adaptive with the worst populations.

3.4 Artificial Bee Colony (ABC)

The Artificial Bee Colony (ABC) is a global optimization algorithm inspired by the intelligent foraging-behavior of the honey bees. The main components of the algorithm include the food sources, employed bees, scouts and onlookers. The bees try to find the best food source using their intelligent colony behavior.

In Nature, for real honey bees, the food sources have values (i.e. the profitability) relating to the factors such as the proximity of the source to the nest, richness of the food and ease of its extraction.

In the artificial bee colony, the value factors of the food sources are the quality criteria of the defined system. The position of the food sources are thus the possible solutions of the function which is aimed to be optimized. The profitability of a food source can be represented as a single quantity, i.e. objective function value related to each solution.

The employed bees are the group of bees who are sent to the food sources to exploit and evaluate the value of the sources. Finishing their evaluation, they go to an area called ‘dancing area’. This is the area in which all bees exchange their information by showing a so-called ‘waggle dance’ to the onlooker bees. The onlookers are responsible to watch the dance of the employed bees and choose the best source depending on the dances and keep the place in their memory.

The employed bees, finishing this step, when the food source is abundant are ready to move to the new sources and exploit them.

The bees that explore the space to find a better source are called scouts. Scouts start to search the new places among the search space to find food sources with higher values respect to the previous evaluated ones found by employed bees. When all scouts finished their search, the employed bees are sent to the founded sources to evaluate the new food-sources. The information is transferred by the waggle dancing and the procedure will be repeated.

For the artificial bees at the first step, an initial population (food source positions) is randomly generated by the operator. The new spaces are also the randomly-created populations. It is also
assumed that to each food source there is one corresponding employed bee. In other words, the number of employed bees is the same as the number of food sources. The population is evaluated and replaced based on the evaluation of the employed, onlooker and scout bees respectively.

The numbers of trials for the scouts to find better food sources are limited to a value defined by the ABC user (called ‘limit’). If within that limit they could not find a better source, they keep their previous source. The procedure of dancing, onlookers ‘evaluation and scouts search is a cycle which will be repeated until the termination criteria(i.e. reaching to the acceptable optimum objective function value, maximum number of iteration or maximum run time) is fulfilled [13].
4 Modelling & Optimization of Optical Imaging System in MatLab

4.1 Introduction

The concept of “optimization” as stated before is comparatively simple, while its implementation can be sometimes fairly complicated. As stated before, a better optical imaging system, is by definition, a system that suffers less from the geometrical aberrations. Geometrical aberrations strongly depend on the geometric shape of the lenses and their arrangement in an optical system. The optimization routines can thus be used to tune the shape of the lenses and their arrangement in order to minimize the contributions from geometrical aberrations.

To apply the optimization process the objective function (here a function which describes the deviation from the ideal aberration-free image) should be therefore determined.

The objective function is obviously related to the aberration factors. The aberrations are included in the real ray-tracing calculations. In particular, the aberrations result in the image of the incoming bundle not being focused in a single point, but rather in a spot of finite size.

Since, the larger this spot size is, the further we are from our ideal optical system, this spot size can be used as the objective function. Consequently, minimizing this function forms our optimization problem. In summary, our optimization problem is thus essentially that of spot size minimization at the paraxial image plane.

For the optimization problem, we need to define an optical system and determine its parameters. We fix the positions of the surfaces and their refractive indices. The curvatures are then left as the variables of the optimization problem.

To validate our data we use the Zemax optical software. For the sake of comparison, we tried to make our optical system conditions as close as possible to those in this optical software package.
The optimized system shape of a six-spherical-surface optical system found by Zemax has a well-known optimal shape, called the ‘Cooke Triplet’ (‘CT’) (A CT system consists of two positive lenses and a negative one located around halfway in between which are separated from each other with sizable air spaces. The aperture stop is placed at the third surface). Figure 4-1(a) shows a schematic of this system. For the sake of simplicity, we placed the aperture stop at the first surface. An example of this type of CT, found by our optimization code, is shown in figure 4-1(b).

Regarding our case study, i.e. optimization of six-spherical-surfaces optical system, we expect to find CTs among the results. Finding CTs with low objective function value, i.e. on the order of e-3 (the same order as similar optimized systems in Zemax) is therefore one of our major challenges.

![Figure 4-1: Cooke Triplet System Shape](image)

In the following section, we model the real ray tracing for an optical system having six spherical surfaces. The essential parameters of optical system needed for optimization such as the EFL and BFL are calculated using programming in MatLab. The details of the program are represented in section 4.3. Section 4.4 is devoted to the numerical calculation of the spot size. The optimization of optical system using GA is the subject of section 4.5.

### 4.2 Real Ray Tracing

To perform the real ray tracing in MatLab, we use the formulas mentioned in section 2.4. An example of our real ray tracing program in MatLab is presented in appendix B. A representative result illustrated in figure below. In this example, the incoming bundle of rays is parallel, i.e. the object point is at infinity.
The path of incoming rays (shown in pink in the figure 4-2) with their known direction cosines is determined using the equations (2.4.1)-(2.4.3). The ray hitting the vertex of the tangent plane to the first spherical surface travels the path Δ (shown by the left green line) to cross the spherical surface. The path Δ and the intersection position are derived from equations (2.4.4)-(2.4.12). Thereafter, the ray changes its directions as it enters a medium with a different refractive index (i.e. traveling inside the lens medium which is shown in blue rays). Formulas (2.4.20)-(2.4.22) are used to calculate the direction cosine changes at the interface between two different media. Having a new set of direction cosines, the steps can just be repeated for any number of surfaces.

![Figure 4-2: Real-Ray-Tracing from an optical system having six spherical surfaces in MatLab. An incoming bundle of parallel rays is traced through the optical system.](image)

To check the correctness and accuracy of our ray tracing codes, we used Zemax. The comparisons show high compatibility in finding both the positions and the direction cosines (at least 4-5 digit compatibility) [14]. An example of the ray-tracing results showing its compatibility with Zemax is presented in Appendix A.

### 4.3 Numerical calculation of EFL and BFL

There are two ways to numerically calculate the paraxial focal length namely, the real ray tracing and the matrix methods (both discussed in section 2.5.2). We first used the method based on the real ray
tracing to calculate both EFL and BFL. To validate our data, we compared the results with Zemax software [14] for a number of optical systems. The comparison shows high accuracy on the EFL case. For the BFL, however, the results are less accurate. We then checked the matrix methods for the BFL calculation. This technique yielded better agreement. (The results are illustrated for a test system in Appendix A)

In our optimization codes we therefore implemented the matrix methods for the BFL calculation and the real ray tracing for the EFL calculation. The related code can be found in appendix B.

4.4 Numerical Calculation of Spot Size

The spot size is estimated using different techniques. One simple way is to calculate the center of the spot diagram and calculate the root-mean-square (RMS) spot size with respect to the center as the reference point (i.e. taking the root mean square of the distances between the intersection points and the center of the spot diagram). If the optical axis is assumed to be oriented in the z direction, for a bundle of N rays propagating from an object point, the spot diagram center and the mean-square spot size (MSS) can be determined as [15, 16]:

\[
\begin{align*}
    x_c &= \left( \frac{1}{N} \right) \sum_{i=1}^{N} x_i \\
    y_c &= \left( \frac{1}{N} \right) \sum_{i=1}^{N} y_i \\
    MSS &= \left( \frac{1}{N} \right) \sum_{i=1}^{N} \left( (x_i - x_c)^2 + (y_i - y_c)^2 \right)
\end{align*}
\]

The root-mean-square spot size is the square root of the MSS. Generally, the RMS spot size is used in the optical design field instead of the MSS.

In MatLab we calculated the spot diagram for a bundle of N rays coming from infinity at the paraxial image plane. One example of the spot diagram resulting from tracing the rays originating from an on-axis object point is shown in figure 4-2. We compared the spot diagram patterns and the sizes for
many optical systems with Zemax. The results show good agreement. The data and configurations for some optical systems are represented in appendix A.

Figure 4.3: Real Ray-Tracing in MatLab. The figure shows a spot diagram (part(c)) at the paraxial image plane. The diagram is formed by a bundle of N rays coming from an on-axis object point placed at infinity (part (a)) passing through the optical system with six spherical surfaces (part (b)).

4.5 Optimization

We examine a simple case of an optical system which can be easily checked by Zemax. We optimize the optical system having six spherical surfaces for objects placed at infinity. We use three points from the object (i.e. an on-axis point, a point in 2/3 object height and one at the top). In Zemax the aperture stop is placed at the third surface. Since our main goal is to investigate the GA performance, we simplified the situation by changing the place of stop to the first surface. In MatLab, we thus place a grid just before the first spherical surface. The rays coming from three object points hitting N grid points are traced through the entire optical system until they strike the paraxial image plane.

We aim to position the paraxial image plane (here is the same as the back focal plane) within a defined range. For that, the code first calculates BFL for each system and checks whether the BFL lies within the defined range. If this is the case it will proceed to the next step, which is calculating the objective function.

The objective function is the weighted summation of the three spot size functions related to each object points. We applied the same weight factors (i.e. 1) for all three spot size functions.
Objective function = $w_1*\text{Spot Size}_1 + w_2*\text{Spot Size}_2 + w_3*\text{Spot Size}_3$ \hfill (4.5.1)

where $w_1 = w_2 = w_3 = 1$.

We aim to fix the effective focal length at a constant value with small tolerance. To implement this constraint in the programming we added the difference between the aimed EFL and the system’s calculated EFL to the objective function, weighted by a factor (i.e. 0.1).

To optimize the system using Genetic Algorithm, MatLab has a built-in function, called ‘ga’, with the possibility of tuning factors involved in Genetic Algorithm such as initial population, number of generations, crossover, mutation and many others. To combine local optimization with Genetic Algorithm, MatLab has some standard built-in functions intended for numerical or analytical local optimization problems. We have used these built-in functions in our ray-tracing codes and implemented the above mentioned theories in our programming to optimize the optical systems. Some examples can be found in the codes in Appendix B.
5  The Performance of the GA

5.1  Introduction

Using our codes developed in MatLab, we can now investigate the performance of the GA on optimization of optical systems.

The influences of the main parameters of GA are assessed in section 5.2. The parameters that were varied are the number of generations, population size, crossover and mutation as well as effective factors in Gaussian mutation (i.e. scale and shrink) and the rate of the crossover.

The GA outcomes are fed into a local optimization operand in MatLab to improve the results and to see how much the GA outcome can be influenced by the local optimization. The resulting data are represented in section 5.3. The optical system’s shapes related to the best systems found by GA are shown and discussed in section 5.4.

5.2  The influence of different GA parameters

5.2.1  Generation

Generation is a stopping criterion of GA which determines after how many numbers of iterations the algorithm terminates.

In this section we assess the effect of increment of generations in the performance of GA. The best suited value of this option for our case study is also aimed to be found. This option has a direct effect on the execution time of the algorithm. It is thus important to evaluate the results improvement together with its associated time.

The GA default value of this option is 100 [11]. We tested 9 different generation numbers (i.e. 30, 50, 100, 150, 200, 300, 400, 500, and 600) with the population size of 20 (the default value of GA). The rest of the options are also left at the GA defaults.

Figure 5-1 represents the average of the objective function values for 100 runs for each case. Looking at the figure, it appears that for less than 500 generations, the objective function fluctuates substantially with a minimum at 400 generations. These fluctuations decrease beyond 500
generations. However, it turns out that none of these generation options causes the algorithm to reach the aimed objective function values. This shows that the GA with default options, even with many generations, cannot achieve the desired target.

We therefore changed the GA crossover and mutation methods and did similar tests with new options to find the optimum value of generations for other combinations of mutation and crossover. We found that by incrementing the number of generations up to 500, an overall improvement in the results was achieved. Further incrementing above 500 shows the same behavior as the system with the GA defaults, i.e. no improvement is observed in the results upon increasing the generation numbers beyond 500. The run time is shown in figure 5-2. As the run time is not dramatically increasing in terms of generation increments, it is wise to choose the number of generations that gives the best results for most of tested options (i.e. 500).

![objective function value versus number of generations](image)

**Figure 5-1:** The average of the objective function versus the number of generations. The population size $p$ is taken to be 20. The x axis shows the number of generations changing from 30 to 600. The y axis gives the related average objective function values.
5.2.2 Population Size

The population size is the number of individuals present in each generation.

Increasing this parameter, the GA will search the space more extensively. This increases the probability for the GA to reach the global solution and to avoid getting trapped in local maxima. However, increasing the population size to large values will significantly increase the execution time of the GA. Thus it is very important to find an optimum range of this parameter regarding the run time and the results.

In the previous section we found out by just increasing the generation for the default GA options that we never get a good enough fitness value. We thus changed the population size to see whether changing this option can help the GA routines (with its other parameters left as default) to reach to the range of desired objective function values.

The default value of the population size in MatLab is 20 [11]. We examined nine different population sizes (i.e. 5, 10, 20, 30, 40, 60, 80, 100 and 200) for 100 runs with 500 generations. The rest of parameters are left at the GA default values. The figure 5-3 shows the relationship between the population size and the average of objective function values over 100 runs.

It is clear from the figure that incrementing the population size from 5 to 60 causes a noticeable improvement on the result; beyond this size the improvements in the results due to increasing the

\[\text{Figure 5-2: The execution time versus the number of generations. The x-axis shows the changing of the number of generations from 30 to 600. The y-axis gives the related running time for each case.}\]
population is modest. We therefore conclude that a population size around 60 is a reasonable optimum value.

However, figure 5-5 shows that for the GA default options incrementing the population still does not result in either the CTs or other system shapes with the desired objective function values in a reasonable time (figure 5-4).

Therefore, we changed the GA crossover and mutation methods also in this case and did some tests with new options to find the optimum population size range. We found that considering the results and execution time, the optimum population size is around 50. We therefore take the combination of population size 50 and generation numbers of 500 for the rest of our study.

Figure 5-3: The objective function value for different population sizes. The x-axis shows the changing of the populations from 5 to 200. The y-axis gives the related objective-function average-values for each case.
5.2.3 Crossover and Mutation

We tested 18 combinations of mutations and crossover methods available in MatLab. There are 6 methods of crossover (i.e. Heuristic, Scattered, Intermediate Single point, Two point and Arithmetic) and 3 type of mutations (i.e. Uniform, Adapt feasible and Gaussian). We run the code 100 times for each case (for the generation = 50 and population size = 500).
The histograms in figure 5-6 contain the results for each case. Each histogram indicates the number of occurrences per objective function values. It appears from the figure that the smallest values of objective function are achieved by the method of Gaussian mutation and heuristic crossover. Therefore this option seems the best for finding the systems with the smallest value of the objective function.

**Figure 5-6:** The histogram for 18 different Mutation and Crossover options. The options include 18 combinations of Crossover and Mutation GA methods (i.e. 3 type of mutation (Uniform, Adapt feasible and Gaussian and 6 type of crossover (Heuristic, Scattered, Intermediate, Single point, Two point, Arithmetic)). The x-axis indicates the objective function values. The y-axis shows the number of occurrence. The result is taken from 100 runs for each option. The population size is 50 and the number of generations is 500.
The average objective function values for all different 18 options are presented in the figure 5-7. From the results it is concluded that the mutation Gaussian method gives the lowest average values for all crossover methods. Moreover, the Heuristic crossover has the lowest average among all crossovers. Therefore the Gaussian mutation with Heuristic crossover gives also the best results regarding the lowest average value.

**Figure 5-7**: The bar graph representing the average objective function values for 18 different combinations of Crossover and Mutation GA methods (i.e. 3 type of mutation (Uniform, Adapt feasible and Gaussian and 6 type of crossover (Heuristic, Scattered, Intermediate, Single point, Two point, Arithmetic)). The denoted numbers in the x-axis are related to the mentioned different options respectively. The result is taken from 100 runs for each option. The population size is 50 and the number of generations is 500.
Figure 5-8: Bar chart showing the number of all good systems, good CTs and Bad CTs related to 18 different crossover and mutation methods in the GA for 100 runs for each case. The green bars are related to the systems which have the CT shape with an objective function value less than 0.01. The blue bars are related to the systems which have the CT shape with an objective function value higher than 0.01. The red bars shows the number of all good systems (including CTs and other shapes) with objective function values less than 0.01 having any system shape.

Figure 5-8 illustrates the number of good (bad) CTs and good systems for the mentioned 18 options. It is obvious from the figure that the option Gaussian mutation and heuristic crossover is the only option which can find the good CTs.

In conclusion, Gaussian mutation combined with heuristic crossover gives the best results with respect to the least objective function values, the average value of the objective function and the number of CTs with desired objective function values.

The important point is that Gaussian mutation combined with heuristic crossover is not only the best option which can satisfy our targets, but it is the only option which could give these results.
5.2.4 Scale and Shrink in Mutation Gaussian

We found that ‘Gaussian’ is the best mutation method. Scale and shrink are two tuning options involved in this method which can affect the performance of the GA. We tried to fine-tune these options to improve the results found by the GA and to find the influence of these parameters on the performance of the GA.

The scale and shrink can take values in the range from 0 to 1. We assessed the results for 9 combinations of 3 scales and shrinks (i.e. 0.01, 0.5 and 1). Figure 5-9 shows the histogram for the number of occurrences per objective function values. The resulting data are for 100 runs for each case.

**Figure 5-9:** The histogram of 9 different combinations of scale and shrink. Each graph shows the number of occurrences per objective function values for 100 runs. The values of scale and shrink are set to be 0.01, 0.5 and 1. The defined options are set to: population = 50, generation = 500, mutation Gaussian and Heuristic crossover.
Figure 5-10: Bar chart showing the number of all good systems (i.e. CT and non-CT systems with objective function value less than 0.01, red bars), good CTs (objective function < 0.01, green) and Bad CTs (objective function > 0.01, blue) related to 9 different scale and shrink in 100 runs for each case. The values of scale and shrink are set to be 0.01, 0.5 and 1. The defined options are set to: population = 50, generation = 500, mutation Gaussian and heuristic crossover.

From the figure 5-9 it appears that the best designs are found for a scale value of 0.5, irrespective of the shrink value. The figure 5-10 illustrates the number of good (bad) CTs and good systems. This figure confirms that the GA could find the maximum number of good systems and good CTs for scale 0.5, irrespective of the shrink value.

We thus conclude that the rate at which the average amount of mutation decreases is not the deterministic factor for our case study. In contrast, the range of the allowed deviation of the mutation (which is determined by the scale factor) can highly affect the results. Giving a high deviation range (results from selecting scale = 1) or limiting this factor to small range (by setting the scale to 0.01) both degrade the improvement of the results. The optimum is somewhere in between (scale = 0.5), which provides a reasonable deviation range from the initial population range for the mutation operator.
5.2.5 Crossover Fraction

Crossover Fraction is another factor which can influence the GA performance. This factor determines the probability at which the population at the next generation is created by the crossover operator [11]. As discussed in section 3.3, the children in the next generations (except elites) are created by mutation and crossover. Therefore, the probability at which the children of the next generation are produced by the mutation is 1 –crossover fraction.

In other words, having a crossover fraction of 0 means the next generation children are all created by the mutation (excluding the elite children), while the crossover fraction of 1 means all children (other than elite ones) are crossover children.

The optimum crossover fraction is expected to be in neither of its extremes. Having a very high mutation probability causes the algorithm to run as a random procedure, thereby neglecting the essential procedure of the GA, namely cross-over. In contrast, decreasing the probability of mutation to a very low value does not generally give good results as it may prevent the algorithm to search all possible areas which cannot be reached by crossover function and therefore may prevent the GA from reaching a satisfactory solution [11].

The GA default of this option is 0.8 in MatLab. We checked 9 different crossover fractions ranges from 0.1 to 0.9 with the steps of 0.1 to find out the optimum value of this option for our problem.

The histogram of the objective function values for 100 runs for each case is represented in figure 5-11. The figure shows that the least objective function is achieved by the fraction 0.6.
Figure 5-11: Histogram for 9 different crossover fractions. Each graph shows the number of occurrences per objective function values for 100 runs. The crossover fractions changes from 0.1 to 0.9 with the steps of 0.1. The GA options are set to: population = 50, generation = 500, mutation Gaussian and Heuristic crossover.

Figure 5-12: Bar chart showing the number of all good systems, good CTs and Bad CTs for 9 different crossover fractions in 100 runs for each case. The crossover fractions changes between the ranges 0.1-0.9. The GA options are set to: population = 50, generation = 500, mutation Gaussian and heuristic crossover.
Figure 5-12 gives the information on the number of good systems achieved by the 9 mentioned crossover fractions. From the resulting data taken from both figures it is concluded that for our problem also neither of the extremes of crossover probabilities is good and the crossover fraction 0.6 gives the best results.

5.3 Combining GA with local optimization

In order to improve the optimization results, a local optimizer can be combined with a global optimizer. We thus added a local optimization operation at the end of our GA code in MatLab to assess the improvement achieved by this combination. The optimized curvatures found by GA are feed to the local optimization as its starting point.

We run the combined code for 18 different combinations of crossover and mutation methods (the same as options mentioned in section 5.2.3). We tested the results for two cases, i.e. population = 20, generations = 300 and population = 50, generations = 500. The average objective function values for 18 options are represented in figure 5.13 (a) and (b) respectively. The dark-blue (light-blue) bars indicate the values in the presence (absence) of local optimization.

From figure 5-13 (a) it is obvious that the local optimization causes some improvement in the results. This improvement is less for some options such as Gaussian mutation combined with Heuristic crossover or Intermediate crossover. In figure (b) both bar graphs are precisely overlapped for all 18 options.
Figure 5.13: The bar chart representing the average objective function values for 18 different combinations of crossover and mutation methods. The dark-blue (light-blue) bars indicate the values in the presence (absence) of local optimization. Part (a) represents the resulting data for generation = 300 and population = 20. In Part (b), generation = 500 and population = 50. The resulting data are taken from 100 run.
We extract the absolute objective function values for all 1800 runs related to 18 options for population=50 and generation = 500 before applying the local optimization and after that. The values for all options were exactly the same. As a sample, we illustrated the data of the twenty best systems in the figure 5-14. Part (a) (blue bar graphs) presents the objective function values before local optimization. Part (b) (green bar graphs) gives the same data after applying local optimization.

This shows that when GA is tuned to higher population size and is left to be executed for larger numbers of generations (p>=50 and g>=500), it can perfectly reach the local minima in any basin of attraction, including the best global optimum found.

However, regarding the execution time, and to avoid the risk of not converging to the local minimum for lower number of generation and population size, it is efficient to add a local optimizer to the GA.
5.4 Optical system shapes found using GA and clustering

In this section we show the best optimized optical systems found by the GA and characterize them using bar charts of their curvatures. This helps us to see the shapes of the best optimized systems found by GA at a glance.

Moreover, by having these charts we can use our visual recognition ability as a tool to group the systems having similar configurations. This grouping (called clustering) helps us to roughly find out the distance between the solutions and to discover the possible existing pools in the objective-function landscape.

We therefore extracted the optical system data related to 20 best systems for three situations. We first search among the resulting data taken from 100 runs of 18 different mutation and crossover methods with the generation 300 and a population size of 30. Then we study the 20 best systems among the resulting data taken from 100 runs of 18 different mutation and crossover methods with the generation 500 and the population size 50. The last case is for 900 runs, i.e. 100 runs for 9 different scales and shrink of Gaussian mutation and heuristic crossover for generation = 500 and population size = 50.
The resulting data for each case are illustrated in figure 5-15, 5-16 and 5-17 respectively.

**Figure 5-15:** 20 best optical systems found by GA among 1800 runs (100 runs for 18 different mutation and crossover methods. Number of generations = 300, Population size = 30). The numbers on the x-axis correspond to the spherical-surfaces. The y-axis gives the curvature values of the related spherical surfaces.

From figure 5-15 it appears that there are three system shapes found by the GA (shown by blue, yellow and purple, the CTs are the purples ones). The configurations of these systems are illustrated in figure 5-18.

As seen in this figure, there are some other system shapes which do not seem to be related to any of these classes.
Figure 5-16: 20 best optical systems found by GA for 1800 runs (100 runs for 18 different Mutation and Crossover methods. Generation = 500, Population size = 50). The numbers in the x-axis corresponds to the spherical-surfaces’ numbers. The y-axis gives the curvature values of the related spherical surfaces.

Figure 5-16 shows that by increasing the number of generations and the population sizes, the occurrence of systems not in one of the identified classes is decreasing and the systems are converging to the three main mentioned groups. A clustering of the system shapes can therefore now be better formed.

The smallest objective function values are related to the CTs, however their number of occurrence is still low compared with the two other system shapes.
Figure 5-17: 20 best optical systems found by GA for 900 runs (100 runs for 9 different scales and shrink for Mutation Gaussian and Crossover Heuristic. Generation = 500, Population size = 50). The numbers on the x-axis correspond to the spherical-surfaces. The y-axis gives the curvature values of the related spherical surfaces.

As seen from the figure 5-17, when we use better tuned options, GA reaches CTs with higher probability (here 75%). The results show that if GA is correctly tuned, it can escape the basins of attraction related to the local minima and reach to the GA best-found global minimum’s basin of attraction.

It also appears that by improving the GA options, the non-optimal groups would shrink. It seems that for the perfectly tuned GA the routine might only find one good system; the same happens in Zemax.
Figure 5-18: Three main different system shapes found by GA optimization code in MatLab. The optical system configurations are taken from Zemax [14].

In figure 5-19 we illustrate the spread of objective function values below 0.012 for the three system shapes mentioned.

![Spread of Objective Function values below 0.012 for Three Different System Groups](image)

Figure 5-19: The spread of objective function values for three different system groups. The y-axis indicates the ranges of the objective function values below a threshold of 0.012 related to the three system groups (shown by pink (CTs), blue and yellow).

As seen from the figure, the objective function values of the three system groups have overlap. This turns out the reason why GA finds three system shapes and not only the group of CTs.
5.5  The Artificial Bee Colony (ABC) versus the GA

MatLab have default settings for the GA optimization method. We have used them to study the performance of GA in optical systems design. However, for the ABC method, there are not yet such a standard packages in MatLab. To compare the performance of the ABC method in optical system optimization with the GA technique, we have used a code written by another researcher [17]. We took the common parameters of ABC and GA, such as population size and number of iterations, the same. Figure 5-20 presents the histogram of the objective function values resulting from the ABC code. The results are taken from 100 runs. The number of good (bad) CTs and the whole good systems found by the ABC and the GA is represented as bar charts in Figure 5-21.

![Histogram of objective function values](image)

**Figure 5-20:** The histograms of objective-function values resulting from the ABC code for 100 runs. The graph is related to the result of the ABC without implementing the local optimization.
Figure 5-21: The System shapes found by the Artificial Bee Colony (ABC) and the GA method (both without local optimization. good CTs (or bad CTs) are the systems which have a CT system shape with the objective function value less (or above) than 0.01. ‘good systems’ have objective function value less than 0.01 irrespective their shapes.

Comparing figure 5-20 with figure 5-13 and 5-14, shows that the mean and the minimum objective-function values resulting from the ABC are higher than those taken from the GA. This illustrates that in our case study, the ABC method could not perform better than GA.

Figure 5-21 shows that the ABC technique can reach to the CTs better than GA. GA has proved weak in finding the CT system shapes which indicates that it does not easily reach the global optimum points. This suggests that the Artificial Bee Colony is a better global optimizer than GA but is poorer in local optimization part. If this is the case, combining it with a local optimizer might get results of the ABC better than GA.

To check this probability, we polished the results taken from the ABC method by local optimization. This is done by feeding the optimized curvatures of optical system resulting from Artificial Bee Colony to the local optimizer operand as its initial point. The resulting data of the combined codes are represented in Figure 5-22 and 5-23.
Figure 5-22: histogram of objective function values resulted from the ABC method combined with local optimization. The graph presents the results for 100 runs.

Figure 5-23: The System shapes found by the Artificial Bee Colony (ABC) and the GA method (both combined with local optimization) in 100 runs. Good CTs (or bad CTs) are the systems which have a CT system shape with the objective function value less (or above) than 0.01. ‘good systems’ shows the number of all systems with the objective function value less than 0.01 disregarding their shapes.

Figure 5-22 and 5-23 show that, combining a local optimization with ABC does not make a significant improvement on the results.

There are many papers which claim that the ABC technique generally gives better results than GA. Dervis Karaboga et al. [18] claim the outperformance of ABC method than GA for multivariable
functions. He confirmed his conclusion again in another paper [19] representing the results of optimization for a large set of numerical test functions.

Our results did not confirm this claim for optical design and raised doubts on the efficiency of ABC method for this application.

However, to prove that in optical system design the GA is a more efficient method than the ABC, a more extensive study on the Artificial Bee colony method and involving more variation of its parameters are needed which due to time limitations, this could not be done within the present study.
6 Summary and Conclusion

In this project we have developed a MatLab code for optical system optimization using the MatLab GA module. This has been done to assess the performance of the GA in optimization of optical systems and to find out the influence of GA tuning parameters on the results. The optical imaging system which we worked on consisted of six spherical optical surfaces. Extension to more surfaces is straightforward. To validate our developed codes, the results of all steps have been checked against the commercial lens-design software-package Zemax. Our results were in good agreement with those of Zemax.

The main parameters of the GA such as population size, number of generation, crossover and mutation, as well as effective factors in Gaussian mutation (i.e. scale and shrink) and the crossover fraction were varied and assessed.

Regarding our case study, i.e. optimization of six-spherical-surfaces optical system, we expected to find CTs among the results. The resulting data shows that for the tuned GA the algorithm can reach to the CT group. It turned out that they are the best system found by our GA optimization codes in terms of their objective function values.

We found that by increasing the generations for the GA with default options (even to very high values of around 600), the algorithm cannot reach satisfactory objective function values. Incrementing the population size, the GA will generally search the space more extensively. This increases the probability for the GA to reach the global solution and avoids getting trapped in local minima. Incrementing the population size to large values (i.e. 200) significantly increased the execution time of the GA while it still did not result in either the CTs or other system shapes with the desired objective function values in a reasonable time.

We then changed the mutation and crossover methods and noticed considerable changes in the results. Our finding shows that these two options are the most effective parameters in GA (as noted by Lin et al. [12]). Among different combinations of mutation and crossover methods, the Gaussian mutation combined with heuristic crossover gives the best results. It also turns out that both in terms
of objective function value as in the number of good CT’s, Gaussian mutation combined with heuristic crossover is the only option which yielded good results.

Regarding the scale and shrink, it appears that the best designs are found for a scale value of 0.5, irrespective of the shrink value. In contrast, changing the strength of mutation by changing scale factor from its optimum value of 0.5 degrades the results significantly.

The optimum crossover fraction is expected to be in neither of its extremes. Our results showed that a crossover fraction of 0.6 gives the best results.

The GA outcomes are fed into a local optimization operation to improve the results. With this combination some improvements have been achieved. In view of the execution time, and to avoid the risk of not converging to the local minimum for lower number of generation and population size, we found it efficient to add a local optimizer to the GA.

We characterized the best optimized optical systems found by the GA in terms of their curvatures and used our visual recognition ability as a tool to group the systems having similar configurations. This grouping helped us to roughly find out the distance between the solutions and to discover the possible existing pools of good solutions in the objective-function landscape.

Starting from population 30 and 300 for the Gaussian mutation and crossover heuristic, three main classes were recognized. By increasing the number of generations and the population sizes, the occurrence of systems not in one of the three identified classes is decreasing and the systems are converging to the CT group. The resulting data showed that by improving the GA options, the non-optimal groups would shrink. It seems that for the perfectly tuned GA the routine might only find one good system (as happens in the GA in Zemax).

The Artificial Bee Colony (ABC), as an alternative to GA, is one of the newest methods of global optimization which is claimed by some authors ([18], [19]) to perform superior to GA. We have applied this method to our problem. The results however indicated that GA is superior in comparison with the ABC method for optical design.

The use of this work is that it provides some guidelines on the GAs in optical design. Our current study shows that the Genetic Algorithms, despite their robustness and versatility [15] can only be efficient if their parameters are rigorously tuned (at least for the case of our defined optical system). Although the parameters founded and recommended are efficient for the defined optical system,
however, these parameters were found after extensive trial and error work outlined in the thesis. This reveals that a GA can be used as an automatic optical system optimization tool with an acceptable aberration correction even on a small personal computer (mentioned also by Van Leijenhorst et al. [15]), but, under the condition that its parameters are first precisely tuned otherwise it may never get the desired target.

**Future Works:**

For the sake of simplicity we placed the aperture stop at the first surface while in the real optical design for optical systems with six spherical surfaces the aperture stop is at the third surface. Our assumption was sufficient for our goal (i.e. study the GA performance for optical systems). However, to improve the results of our MatLab optimization code we can change the place of the aperture stop to the third surface. Due to the limitation of the time this has not been done and can be investigated later.

The form of the objective function landscape near the optimum is an important feature which gives invaluable information for in optical design. It is possible to use the developed code to achieve some information on the objective function behavior near the optimum point in order to get some insights on the shape of objective function. This idea could not be investigated within this master project.
Appendix A

Real Ray Tracing, EFL, BFL and Spot Diagram Calculation Using MatLab and Comparison with Zemax

In chapter 4 we mentioned the high accuracy of our Matlab codes on real ray tracing, optical parameter and spot diagram calculations in compare with Zemax. In this appendix we give some examples of these comparisons.

We first present the results of real ray tracing together with the EFL-BFL calculation for a Cooke Triplet system found by Zemax (we called this system here CT1). For the spot diagram we gave also another example of a CT system which has been found by our optimization program in MatLab.

Real Ray Tracing:

Table A.1 represents the information of a Cooke Triplet optical system (i.e. CT1). The configuration of this system is shown in figure A.1.

Table A.1:

<table>
<thead>
<tr>
<th>Surf</th>
<th>Type</th>
<th>Radius</th>
<th>Thickness</th>
<th>Glass</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJ</td>
<td>STANDARD</td>
<td>Infinity</td>
<td>Infinity</td>
<td></td>
</tr>
<tr>
<td>STO</td>
<td>STANDARD</td>
<td>22.01359</td>
<td>3.258956</td>
<td>SK16</td>
</tr>
<tr>
<td>2</td>
<td>STANDARD</td>
<td>−435.7604</td>
<td>6.007551</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>STANDARD</td>
<td>−22.21328</td>
<td>0.9999746</td>
<td>F2</td>
</tr>
<tr>
<td>4</td>
<td>STANDARD</td>
<td>20.29192</td>
<td>4.750409</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>STANDARD</td>
<td>79.6836</td>
<td>2.952076</td>
<td>SK16</td>
</tr>
<tr>
<td>6</td>
<td>STANDARD</td>
<td>−18.39533</td>
<td>42.41507</td>
<td></td>
</tr>
<tr>
<td>IMA</td>
<td>STANDARD</td>
<td>Infinity</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure A.1: Configuration of a Cooke Triplet optical system found by Zemax.
Table A.2: The resulting Real-Ray-Tracing data of CT1 system extracted from Zemax.

<table>
<thead>
<tr>
<th>Surf</th>
<th>X-coordinate</th>
<th>Y-coordinate</th>
<th>Z-coordinate</th>
<th>X-cosine</th>
<th>Y-cosine</th>
<th>Z-cosine</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJ</td>
<td>Infinity</td>
<td>Infinity</td>
<td>Infinity</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>1</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
</tr>
<tr>
<td>2</td>
<td>1.3346691375E-01</td>
<td>3.2589356565E+00</td>
<td>1.0563926812E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
</tr>
<tr>
<td>3</td>
<td>5.3165592507E+00</td>
<td>9.2601437229E+00</td>
<td>1.0563926812E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
</tr>
<tr>
<td>4</td>
<td>5.0242904301E+00</td>
<td>1.0274941361E+00</td>
<td>1.0563926812E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
</tr>
<tr>
<td>5</td>
<td>1.0563926812E+00</td>
<td>1.0563926812E+00</td>
<td>1.0563926812E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
</tr>
<tr>
<td>6</td>
<td>1.2184323532E+00</td>
<td>1.7926570226E+00</td>
<td>6.0384036600E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
</tr>
<tr>
<td>7</td>
<td>3.3272246017E+00</td>
<td>6.0384036600E+00</td>
<td>6.0384036600E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
<td>0.00000000E+00</td>
</tr>
</tbody>
</table>

Real Ray Trace Data:

Table A.3: The resulting Real-Ray-Tracing data of CT1 system extracted from Matlab.

<table>
<thead>
<tr>
<th>Surf Number</th>
<th>x coordinates</th>
<th>y coordinates</th>
<th>z coordinates</th>
<th>X cosine-direction</th>
<th>Y cosine-direction</th>
<th>Z cosine-direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.0000000666</td>
<td>0.0000100000</td>
<td>0</td>
<td>0.0409951272</td>
<td>0.999159346</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.1337129477</td>
<td>3.2589364851</td>
<td>0</td>
<td>0.0663277044</td>
<td>0.997797893</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0.5326362260</td>
<td>9.2601212038</td>
<td>0</td>
<td>0.0500661499</td>
<td>0.998745903</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0.5835050803</td>
<td>10.274897864</td>
<td>0</td>
<td>0.0993147657</td>
<td>0.995056067</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1.0575008862</td>
<td>15.023946129</td>
<td>0</td>
<td>0.0560946114</td>
<td>0.998425457</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1.2206839938</td>
<td>17.9284348984</td>
<td>0</td>
<td>0.0497007757</td>
<td>0.998764152</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>3.3333726873</td>
<td>60.3840646335</td>
<td>0</td>
<td>0.0497007757</td>
<td>0.998764152</td>
</tr>
</tbody>
</table>
**Table A.4:** EFL and BFL of CT1 optical system taken from Zemax.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effective Focal Length</td>
<td>49.99999 (in air at system temperature and pressure)</td>
</tr>
<tr>
<td>Back Focal Length</td>
<td>42.41507</td>
</tr>
</tbody>
</table>

**Table A.5:** EFL and BFL of CT1 optical system taken from MatLab.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effective Focal Length (EFL)</td>
<td>49.999859</td>
</tr>
<tr>
<td>Back Focal Length (BFL)</td>
<td>42.415070</td>
</tr>
</tbody>
</table>

**Spot Diagram:**

We represent the resulting spot diagrams from both Zemax and MatLab in the following figures and tables. The first presented set are related to an optimized CT found by Zemax (CT1), the second one is another optimized CT system found by our MatLab optimization code.
Figure A.2 Spot Diagram of CT1 optical system. The resulting figures and data are extracted from Zemax.

Table A.6 Spot Diagram of CT1 optical system. The resulting figures and data are extracted from MatLab.

<table>
<thead>
<tr>
<th>Field</th>
<th>RMS Radius</th>
<th>Geo Radius</th>
<th>Scale Bar</th>
<th>Reference</th>
<th>Optical System</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.183</td>
<td>4.96</td>
<td>20</td>
<td>CENTROID</td>
<td>Neda_C3plet_1</td>
</tr>
<tr>
<td>2</td>
<td>2.679</td>
<td>5.514</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3.332</td>
<td>7.177</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Units are in meter.

RMS Spot Size: 2.111e-6  2.722e-6  3.426e-6
**Figure A.4** Spot Diagram of an optimized CT found by MatLab. The results are taken from Zemax.

**Table A.7:** Spot Diagram of an optimized CT found by MatLab. The results are taken from MatLab.

<table>
<thead>
<tr>
<th>Field</th>
<th>Units are in meter.</th>
<th>Optical system: NEDTU1_2154</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rms Spot Size: 0.0468 e-6</td>
<td>1.113 e-6</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2.1326 e-6</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix B

The Main Optimization MatLab Codes:

Real Ray Tracing Code

function [merit_function] = Raytracing_compare_zemax_drawing(c)
% This is the 3D_raytracing code
clear all; clc;clf;close all;
format long;
% Take the information of the object and the direction of incoming rays from the file "data_of_staring_point.m":
% Take the data related to the optical system from the file "data_of_optical_system.m":
[d ,n ,c , nsurf ] = data_of_optical_system ;
[x_minus1, y_minus1 , z_minus1, L , M , N , number_of_gridpoints] = grid_data_of_staring_point;
[paraxial_focal_distance ] = paraxial_calculation
d(nsurf+1) = 2.*paraxial_focal_distance
delta = 0 ;
sum_deltas = 0;
for kk=1:number_of_gridpoints
    for ii = 1: (nsurf+1)
        z0 = z_minus1(1,kk) + d(ii) - N(1,kk).*delta ;
        x0 = x_minus1(1,kk) + (L(1,kk)./N(1,kk)).*( z0 - z_minus1(1,kk)) ;
        y0 = y_minus1(1,kk) + (M(1,kk)./N(1,kk)).*( z0 - z_minus1(1,kk)) ;
        x = [x_minus1(1,kk) x0];
        y = [y_minus1(1,kk) y0];
        z = [z_minus1(1,kk) z0];
        figure(5);
        if (ii == 1); linespec = 'm'; end
        if (ii == 2); linespec = 'b'; end
        if (ii == 3); linespec = 'm'; end
        if (ii == 4); linespec = 'b'; end
        if (ii == 5); linespec = 'm'; end
        if (ii == 6); linespec = 'b'; end
        if (ii == 7); linespec = 'm'; end
        if (ii == 8); linespec = 'b'; end
        if (ii == 9); linespec = 'm'; end
        plot3(z,x,y,linespec);
        if ( ii< (nsurf+1) )
            hold on; grid on;
xlabel('z'); ylabel('x'); zlabel('y');
            % calculate the position of the intersection point of
            % the incoming ray( P_-1:P0) with surface c(ii)(which occurs at point p1(x1,y1,z1))
            F = c(ii).*x0.^2 + y0.^2 ;
            G = N(1,kk) - c(ii).*L(1,kk).*x0 + M(1,kk).*y0;
            %delta = the distance p0p1
            delta = F ./ ( G + (G.^2 - c(ii).*F).^1/2);
            x1 = x0 + L(1,kk).* delta ;
            y1 = y0 + M(1,kk).* delta ;
            z1 = z0 + N(1,kk).* delta ;
            %plot line between point 'p_-1'(x_minus1,y_minus1,z_minus1) and point 'p0'(x0,y0,z0)
            if (kk==1)
                fprintf(' ************for the starting point: \n x_minus1 = \%f \n , y_minus1 = \%f\n , z_minus1 = \%f\n
 z_minus1[1,1],y_minus1[1,1],z_minus1[1,1]);
                fprintf('with the starting directional cosines : L = \%f\n , M = \%f\n , N = \%f\n
 L(1,1),M(1,1),N(1,1));
                fprintf(' ************ for the surface number %ld \n X- coordinate = \6.10f\n Y-coordinate = \6.10f\n Z-coordinate = \6.10f\n\n', ii+1 , x1,y1,z1);
            end
        end
end

Optical System Optimization Using Genetic Algorithm

x = [x0 x1];
y = [y0 y1];
z = [z0 z1];
linespec = ':g';
plot3(z,x,y,linespec);

% calculate the ray coming from p1 after refraction
% which gets the direction L',M',N

cosI = (G.^2 - c(ii).*F).^(1/2);

k = c(ii).* (nprime_cosIprime - n(ii).* cosI);
Lprime = (n(ii).*L(1,kk) - k.*x1 )./ n(ii+1);
Mprime = ( n(ii).*M(1,kk) - k.*y1 )./ n(ii+1);
% \$Nprime = (n(ii).*N - k.*z1 + nprime_cosIprime - n(ii).* cosI )/ n(ii+1)
% but we also can use the conservation law for directional cosines and
% derive Nprime directly from Lprime and Mprime as below:
Nprime = (1 - (Lprime.^2 + Mprime.^2) ).^(1/2);
fprintf ('X-cosine = %6.10f\n Y-cosine = %6.10f\n Z-cosine = %6.10f\n',
Lprime,Mprime , Nprime);

end

end

end

hold off;
end

data_of_optical_system.m:
function [ d, n , c , nsurf ] = data_of_optical_system( )
nsurf = 6;
d(1) = 0    ; d(2) = 2            ; d(3) = 3    ; d(4) = 2            ;  d(5) = 5     ; d(6)= 2 ;           ; d(7) = 10  ;
c(1) = 1/23 ; c(2) = -1/45        ; c(3)= 1/63  ; c(4)= -1/43         ; c(5) = 1/80   ; c(6) = -1/35 ; c(7) = 0  ;
n(1) = 1    ; n(2) = 1.6226076881 ; n(3) = 1    ; n(4) = 1.6226076881 ; n(5)=1        ; n(6) = 1.6226076881 ; n(7)= 1    ; n(8) = 1 ;
end

grid_data_of_staring_point.m:
function [ x_minus1 , y_minus1 , z_minus1 , L , M , N , number_of_gridpoints ] =
grid_data_of_staring_point( )
clear all; clc;clf;close all;
nm = 0 ;
% nm is the number of points inside the circular grid for which we want to calculate cosine
directions L,M,N and trace its coming rays
for ii=1:3
% Assuming the object in physical infinity (100*focal lenght), we define 3
% points of the object as the starting points:
% one at the optical axis (drawn with m.. colour), second is in
% 2/3 of the maximum object`s height (drawn with green) and the third is at the
% top of the object(drawn with blue ).
if (ii==1);linespec = 'm'; x0_object=0;y0_object=0;z0_object=-10e5; end
if (ii==2);linespec = 'g'; x0_object=0;y0_object=2/3*1e5;z0_object=-10e5; end
if (ii==3);linespec = 'b';x0_object=0;y0_object=1e5;z0_object=-10e5; end
xgrid_cartesian = [-2.4 : 0.8 : 2.4 ];
ygrid_cartesian = [-2.4 : 0.8 : 2.4 ];
zgrid_cartesian = [0 0 0 0 0 0 0 0 ] ;
for mmm = 1:7 % numbr of points in the x axis
    for nnn = 1:7 % number of points in the y axis
        if ((xgrid_cartesian(nnn)).^2+ ygrid_cartesian(mmm).^2)<(2.5^2) % form a
circular grid with the radius of 2.5
            % count the number of points inside the circular grid
            xgrid_square(nnn) = xgrid_cartesian(nnn);
ygrid_square(nnn)=ygrid_cartesian(mmm);
zgrid_square(nnn)= zgrid_cartesian(mmm);
end
end
end end end

% end of calculation for one travelling + refraction from one surface
% start calculation for the next travelling + refraction step
L(1,kk) = Lprime ; M(1,kk) = Mprime ; N(1,kk) = Nprime ;
x_minus1(1,kk) = x1 ;
y_minus1(1,kk) = y1 ;
z_minus1(1,kk) = z1 ;
end
end
end

hold off;
x = [x0_object xgrid_square(nm)];
y = [y0_object ygrid_square(nm)];
z = [z0_object zgrid_square(nm)];
figure(5);plot3(x,y,z,'line spec'); % plot the circular grid
hold on; grid on;

% Calculate the directions of rays extracted from the circular grid
lenght_object_to_gridpoint = ((xgrid_square(nm) - x0_object).^2 + (ygrid_square(nm) - y0_object).^2 + (zgrid_square(nm) - z0_object).^2).^(1/2);
L_grid(nm) = (xgrid_square(nm) - x0_object)./lenght_object_to_gridpoint;
M_grid(nm) = (ygrid_square(nm) - y0_object)./lenght_object_to_gridpoint;
N_grid(nm) = (1-((L_grid(nm).^2)+(M_grid(nm).^2))).^(1/2);

figure(5); plot3(z,x,y,'line spec'); % plot the circular grid
hold on; grid on;
% Calculate the directions of rays extracted from the circular grid
lenght_object_to_gridpoint = ((xgrid_square(nm) - x0_object).^2 + (ygrid_square(nm) - y0_object).^2 + (zgrid_square(nm) - z0_object).^2).^(1/2);
L_grid(nm) = (xgrid_square(nm) - x0_object)./lenght_object_to_gridpoint;
M_grid(nm) = (ygrid_square(nm) - y0_object)./lenght_object_to_gridpoint;
N_grid(nm) = (1-((L_grid(nm).^2)+(M_grid(nm).^2))).^(1/2);

paraxial_calculation.m
function [paraxial_focal_distance] = paraxial_calculation
    c(7)=0;
    % Take the data related to an optical system.
    [nsurf, d, n, c] = data_of_optical_system;
    % calculate bfl with Matrix method
    Mtot = [ 1 0; 0 1 ];
    for ii = 1: (nsurf - 1)
        M_r = [ 1 -((n(ii+1) - n(ii)).*c(ii)); 0 1 ];
        M_t = [ 1 0; d(ii+1)/n(ii+1) 1 ];
        Mtot = M_t * M_r * Mtot;
    end
    ii = nsurf;
    Mtot = [ 1 -((n(ii+1) - n(ii)).*c(ii)); 0 1 ] * Mtot;
    B = Mtot(1,1); A = -Mtot(1,2);
    D = -Mtot(2,1); C = Mtot(2,2);
    beta = (B + A.*(-500000000./n(1))).^(-1);
    paraxial_focal_distance = (n(nsurf+1)/A).*C - beta;
    bfl = paraxial_focal_distance
end

function efl_bflCalculation
    c(7)=0;
    % Take the data related to an optical system.
    [nsurf, d, n, c] = data_of_optical_system;
    % calculate bfl with Matrix method
    Mtot = [ 1 0; 0 1 ];
    for ii = 1: (nsurf - 1)
        M_r = [ 1 -((n(ii+1) - n(ii)).*c(ii)); 0 1 ];
        M_t = [ 1 0; d(ii+1)/n(ii+1) 1 ];
        Mtot = M_t * M_r * Mtot;
    end
    ii = nsurf;
    Mtot = [ 1 -((n(ii+1) - n(ii)).*c(ii)); 0 1 ] * Mtot;
    B = Mtot(1,1); A = -Mtot(1,2);
    D = -Mtot(2,1); C = Mtot(2,2);
    beta = (B + A.*(-500000000./n(1))).^(-1);
    paraxial_focal_distance = (n(nsurf+1)/A).*C - beta;
    bfl = paraxial_focal_distance
% end of bfl calculation using matrix method
% calculate efl and bfl with ray-tracing method
[paraxial_focal_distance] = data_of_staring_point_efl;
x_minus1 = 0;
y_minus1 = 0.0002;
z_minus1 = 0 ;
L = 0.00 ;
M = 0 ;
N = 1 - ((L.^2+M.^2).^1/2);
delta = 0 ;

for ii = 1: (nsurf+1)
    z0 = z_minus1 + d(ii) - N.*delta ;
    x0 = x_minus1 + (L./N).* (z0 - z_minus1) ;
    y0 = y_minus1 + (M./N).* (z0 - z_minus1) ;
    if ( ii< (nsurf+1) )
        F = c(ii).*((x0.^2 + y0.^2) ;
        G = N - c(ii).*((L.*x0 + M.*y0) ;
        delta = the distance p0p1
        delta = F ./ ( G + (G.^2 - c(ii).*F).^1/2);
        x1 = x0 + L.* delta ;
        y1 = y0 + M.* delta ;
        z1 = z0 + N.* delta ;
        cosI = (G.^2 - c(ii).*F).^1/2 ;
        nprime_cosIprime = ((n(ii+1).^2 - (n(ii).^2.*(1 - cosI .^2)) ).^1/2);
        k = c(ii) .* (nprime_cosIprime - n(ii) .* cosI ) ;
        Lprime = [n(ii).*L - k.*x1 ] ./ n(ii+1) ;
        Mprime = [n(ii).*M - k.*y1 ] ./ n(ii+1) ;
        Nprime = ( 1 - (Lprime.^2 + Mprime.^2) ).^1/2 ;
    end
debug end of calculation for one travelling + refraction from one surface
% start calculation for the next travelling + refraction
L = Lprime ; M = Mprime ; N = Nprime ;
if (ii==nsurf);
    N_of_last_surf = Nprime;
    y_minus1_of_lastsurf = y1 ;
end
x_minus1 = x1 ;
y_minus1 = y1 ;
z_minus1 = z1 ;
end

nu_angle = acos(N_of_last_surf) ;
efl = (0.0002 ./ nu_angle)
bfl2 = y_minus1_of_lastsurf./nu_angle
% end of bfl and efl calculation using ray-tracing
method...................................................
disp('bfl using matrix method = '); disp(bfl1);
disp('bfl using raytracing method = '); disp(bfl2);
disp('efl using raytracing method = '); disp(efl);
end

Spot Diagram and Spot Size calculation Code

function [ bjective_fun ] = objective_funcalspot(c)
clf; close all;
% Take the data related to the optical system from the file "data_of_optical_system.m" :
nsurf , d ,n ] = data_of_optical_system ;c(7)=0;
Mtot = [ 1 0 ; 0 1 ] ;
for ii = 1: (nsurf-1)
    M_r = [ 1 -((n(ii+1)- n(ii)).*c(ii) ; 0 1 ] ;
    M_t = [ 1 0 ; d(ii+1)./n(ii+1) 1 ] ;
    Mtot = M_t * M_r * Mtot ;
end
ii = nsurf ;
Mtot = [ 1 -((n(ii+1)- n(ii) ).*c(ii) ) ; 0 1 ] * Mtot ;
B = Mtot(1,1) ; A = - Mtot(1,2) ;
D = -Mtot(2,1) ; C = Mtot(2,2) ;
beta = ( D + A .* (-5000000000./n(1) ) ).^-1 ;
paraxial_focal_distance = (n(nsurf+1)/A).* (C - beta ) ;
bfl = paraxial_focal_distance;
% end of BFL calculation
d(nsurf+1) = paraxial_focal_distance;
delta = 0;
sum_deltas(1,3) = 0;
sum_x(1,3) = 0; sum_y(1,3) = 0;
[x_minus1,y_minus1,z_minus1,Lbottom,Mbottom,Nbottom,Lmiddle,Mmiddle,Nmiddle,Ltop,Mtop,Ntop]
,num_gridpoints]= grid_data_of_staring_point_ga;
x_start = x_minus1;
y_start = y_minus1;
z_start = z_minus1;
number_of_gridpoints= num_gridpoints;
n_grid = num_gridpoints;
% iiobj is the number of different points of object (here it is 3 points, bottom of object,middle(2/3) and top)
num_of_objectpoints = 3;
for iiobj=1:num_of_objectpoints
sum_x(1,iiobj) = 0;
sum_y(1,iiobj) = 0;
sum_deltas(1,iiobj) = 0;
if (iiobj==1)
L = Lbottom;
M = Mbottom;
N = Nbottom;
elseif (iiobj==2)
L = Lmiddle;
M = Mmiddle;
N = Nmiddle;
elseif (iiobj==3)
L = Ltop;
M = Mtop;
N = Ntop;
end
for kk=1:number_of_gridpoints
x_minus1 = x_start;
y_minus1 = y_start;
z_minus1 = z_start;
for ii = 1: (nsurf+1)     z0 = z_minus1(1,kk) + d(ii) - N(1,kk).*delta;
x0 = x_minus1(1,kk) + (L(1,kk)./N(1,kk)).*( z0 - z_minus1(1,kk));
y0 = y_minus1(1,kk) + (M(1,kk)./N(1,kk)).*( z0 - z_minus1(1,kk));
x = [x_minus1(1,kk) x0];
y = [y_minus1(1,kk) y0];
z = [z_minus1(1,kk) z0];
% calculate the position of the intersection point of ..
% the incoming ray(P_-1:P0) with surface c(1)(which occurs at point   pl(x1,y1,z1))
P = c(ii).*(x0.^2 + y0.^2) ;
G = N(1,kk) - c(ii).*x0 + M(1,kk).*y0;
%delta = the distance p0p1
delta = F ./ ( F^2 - c(ii).*F ).*(1/2));
x1 = x0 + L(1,kk).* delta ;
y1 = y0 + M(1,kk).* delta ;
z1 = z0 + N(1,kk).* delta ;
% calculate the ray coming from p1 after refraction
% which gets the direction L',M',N
k = c(ii).*(nprime_cosIprime - n(ii).*cosI ) ;
Lprime = (n(ii).*L(1,kk) - k.*x1) ./ (n(ii)+1) ;
Mprime = (n(ii).*M(1,kk) - k.*y1) ./ (n(ii)+1) ;
Nprime = (1- (Lprime.^2 + Mprime.^2) ).*(1/2) ;
% start calculation for the next travelling + refraction step
L(1,kk) = Lprime;
M(1,kk) = Mprime;
N(1,kk) = Nprime;
x_minus1(1,kk) = x1;
y_minus1(1,kk) = y1;
z_minus1(1,kk) = z1;
end
x_spotdiagram(iiobj,kk) = x0
y_spotdiagram(iiobj,kk) = y0
sum_x (1,iiobj) = sum_x (1,iiobj) + x_spotdiagram(iiobj,kk) + sum_x (1,iiobj);
sum_y (1,iiobj) = sum_y (1,iiobj) + y_spotdiagram(iiobj,kk) + sum_y (1,iiobj);
end
avg_x(1,iiobj) = sum_x(1,iiobj) ./ number_of_gridpoints;
avg_y(1,iiobj) = sum_y(1,iiobj) ./ number_of_gridpoints;

% plot spot diagram
if (iiobj==1)
    figure (1) ;hold on;
    title ('the spot diagram at paraxial image plane ');
    plot(x_spotdiagram(1,1:n_grid),y_spotdiagram(1,1:n_grid),'.');
    hold off;
    figure (2) ;hold on;
    title ('the spot diagram at paraxial image plane ');
    plot(x_spotdiagram(1,1:n_grid),y_spotdiagram(1,1:n_grid),'.');
    axis([-5e-3 -5e-3 5e-3 5e-3]);
else if(iiobj==2)
    figure (3) ;hold on;
    title ('the spot diagram at paraxial image plane ');
    plot(x_spotdiagram(2,1:n_grid),y_spotdiagram(2,1:n_grid),'.');
    hold off;
    figure(4);hold on;
    title('the spot diagram at paraxial image plane ');
    plot(x_spotdiagram(2,1:n_grid),y_spotdiagram(2,1:n_grid),'.');
    axis([-5e-3 5e-3 3.33 3.34]);
else if(iiobj==3)
    figure (5) ;hold on;
    title ('the spot diagram at paraxial image plane ');
    plot(x_spotdiagram(3,1:n_grid),y_spotdiagram(3,1:n_grid),'.');
    hold off;
    figure(6);hold on;
    title('the spot diagram at paraxial image plane ');
    plot(x_spotdiagram(3,1:n_grid),y_spotdiagram(3,1:n_grid),'.');
    axis([-5e-3 5e-3 5.000 5.01]);
end

for iiobj = 1:3
    sum_deltas(1,iiobj) = 0;
    for kk = 1:number_of_gridpoints;
        delta_x(iiobj,kk) = x_spotdiagram(iiobj,kk) - avg_x(1,iiobj);
        delta_y(iiobj,kk) = y_spotdiagram(iiobj,kk) - avg_y(1,iiobj);
    end
    distance_real_paraxial(iiobj,kk) = (delta_x(iiobj,kk).^2 + delta_y(iiobj,kk).^2);
    sum_deltas(1,iiobj) = sum_deltas(1,iiobj) + distance_real_paraxial(iiobj,kk);
end

spot_size(1,iiobj) = sum_deltas(1,iiobj) ./ number_of_gridpoints;

if (iiobj == 1)
    spot_size1 = spot_size (1,iiobj)
elseif (iiobj == 2)
    spot_size2 = spot_size (1,iiobj)
elseif (iiobj== 3)
    spot_size3 = spot_size (1,iiobj)
end

% efl calculation
[x_minus1, y_minus1, z_minus1, L, M, N] = data_of_staring_point_efl;
% Take the data related to the optical system from the file "data_of_optical_system.m":
[n_surf, d, n ] = data_of_optical_system;
delta = 0;
for ii = 1: (n_surf+1)
    z0 = z_minus1 + d(ii) - N.*delta;
    x0 = x_minus1 + (L./N).*(z0 - z_minus1);
    y0 = y_minus1 + (M./N).*(z0 - z_minus1);
    if ( ii< (n_surf+1) );
        F = c(ii).*x0.^2 + y0.*2;
        G = N - c(ii).*x0 + (L.*x0 + M.*y0);
        delta = the distance p0pl
        G = G + (G.*2 - c(ii).*F).*1/(2);
        x1 = x0 + L.*delta;
        y1 = y0 + M.*delta;
        z1 = z0 + N.*delta;
        cosI = (G.*2 - c(ii).*F).*1/(2);
        nprime_cosIprime = ((n(ii+1).*2) - (n(ii).*2)) .* (1- cosI .^2) .^1/2;
        k = c(ii).*L.* (nprime_cosIprime - c(ii)) .^1/2;
        Lprime = n(ii).*L - k.*x1/. n(ii+1);
        Mprime = n(ii).*M - k.*y1/. n(ii+1);
        Nprime = (1- (Lprime.^2 + Mprime.^2)) ./ (Lprime);%end of calculation for one travelling + refraction from one surface
% start calculation for the next travelling + refraction from the ...
L = L_prime; M = M_prime; N = N_prime;
if (ii==nsurf);
    N_of_last_surf = N_prime;
end
end
x_minus1 = x1;
y_minus1 = y1;
z_minus1 = z1;
end
end
nu_angle = acos(N_of_last_surf);
efl = (0.0002 ./ nu_angle);
% w is the weight of the factors - efl_0 is our target fixed efl
efl_0 = 49.9999;
w1 = 1; w2 = 1; w3 = 1; w4 = 0.1;
objective_fun = w1.*spot_size1+w2.*spot_size2+ w3.*spot_size3+ w4.*((efl-efl_0).^2).^(1/2))
end

GA Optimization Code (for 9 different Crossover and Mutation options)

function [ optimum_R ] = optim_fun_9diffpar_part2(objective_fun)
% the function take the optical system data from other files and
% start to optimize(minimize) the objective function (calculated
% at "objective_funcalzimax.m" code) to find the optimum curvatures.
% the code runs for 9 different GA options (9 different combination of crossover and mutation
% methods
% for each option it runs 100 times.
% At the end it makes a histogram of all the options.
clc;close all;clear all;clf;
fileID = fopen('with_withoutlocal_part3new.txt','w');
ff = 0;
ffimag = 0;
num_of_correct_ans = 0;
num_of_failed_ans = 0;
error_number_fmin = 0;
opt_c_matrix = zeros (1,1:6)
rayfailed_R1 = zeros (1,1:6)
failed_initial_R1 = zeros (1,1:6)
opt_c_matrix0 = zeros (1,1:6)
rayfailed_R10 = zeros (1,1:6)
opt_obj_fun_matrix = zeros (9,100)
opt_obj_fun_matrix0 = zeros (9,100)
time1 = cputime;
um_of_surf = 6;
Num_of_run = 100;
rn = Num_of_run;
number_of_options = 9;
p = 50; % p is population
g = 500; % g is generation
for n_of_option = 1 : number_of_options
    if (n_of_option==1) ;
        options = gaoptimset ('Display','iter','TolFun',1e-30...,'CrossoverFcn',@crossoversinglepoint,'MutationFcn',@mutationadaptfeasible,'populationsize',p,'Generations', g,'TolCon', 1e-20);% options = gaoptimset ('Display','iter','TolFun',1e-30...,'CrossoverFcn',@crossovertwopoint,'MutationFcn',@mutationadaptfeasible,'populationsize',p,'Generations', g,'TolCon', 1e-20);% options = gaoptimset ('Display','iter','TolFun',1e-30...,'CrossoverFcn',@crossoverarithmetic,'MutationFcn',@mutationadaptfeasible,'populationsize',p,'Generations', g,'TolCon', 1e-20);% options = gaoptimset ('Display','iter','TolFun',1e-30...,'CrossoverFcn',@crossoverheuristic,'MutationFcn',@mutationgaussian,'populationsize',p,'Generations', g,'TolCon', 1e-20)
elseif (n_of_option==5) ;
    options = gaoptimset ('Display','iter','TolFun',1e-30...
    ,'CrossoverFcn',@crossoverscattered,'MutationFcn',@mutationgaussian,'populationsize',p,'Generations', g,'TolCon', 1e-20)
    elseif (n_of_option==6) ;
    options = gaoptimset ('Display','iter','TolFun',1e-30...
    ,'CrossoverFcn',@crossoverintermediate,'MutationFcn',@mutationgaussian,'populationsize',p,'Generations', g,'TolCon', 1e-20)
    elseif (n_of_option==7) ;
    options = gaoptimset ('Display','iter','TolFun',1e-30...
    ,'CrossoverFcn',@crossoversinglepoint,'MutationFcn',@mutationgaussian,'populationsize',p,'Generations', g,'TolCon', 1e-20)
    elseif (n_of_option==8) ;
    options = gaoptimset ('Display','iter','TolFun',1e-30...
    ,'CrossoverFcn',@crossovertwopoint,'MutationFcn',@mutationgaussian,'populationsize',p,'Generations', g,'TolCon', 1e-20)
    elseif (n_of_option==9) ;
    options = gaoptimset ('Display','iter','TolFun',1e-30...
    ,'CrossoverFcn',@crossoverarithmetic,'MutationFcn',@mutationgaussian,'populationsize',p,'Generations', g,'TolCon', 1e-20)
    end
    for ii = 1 : Num_of_run
        lb = [-1/10 -1/10 -1/10 -1/10 -1/10 -1/10 ];
        ub = [ 1/10 1/10 1/10 1/10 1/10 1/10 ];
        [optimum_c, opt_objective_fun,exitflag] = ga (@objective_funcalzimax, 6 ,
        [],[],[],[],lb,ub,[],[],options)
        x0 = optimum_c;
        optimum_c0 = optimum_c;
        disp(optimum_c0);
        % optimum_c0 is the optimum curvature without applying the local opt.
        optimum_R = 1./optimum_c
        optimum_objective_fun0 = opt_objective_fun;
        fprintf(fileID, '
the opt_objective_fun resulted from GA without Local opt. is: %f
',opt_objective_fun)
    try
        optionsfmin = optimset('Display','iter','TolFun',1e-30,'TolX',1e-20);
        lb = [-1 -1 -1 -1 -1 -1 ];
        ub = [ 1 1 1 1 1 1 ];
        [optimum_c, opt_objective_fun,exitflag] = fmincon2(@objective_funcalzimax ,x0
        ,[],[],[],[],lb,ub,[],optionsfmin)
        optimum_R = 1./optimum_c
    catch
        error_number_fmin = error_number_fmin + 1;
        fprintf(fileID,'failure in finding local optimizer due to starting point!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n')
    end
    if  ( imag(opt_objective_fun) == 0 )
        num_of_correct_ans = num_of_correct_ans + 1;
        nn = num_of_correct_ans;
        opt_objective_fun nn (nn) = opt_objective_fun;
        % opt_objective_fun nnGA is only with GA without Local opt.
        opt_objective_fun nn0 (nn) = opt_objective_fun0;
        opt_c_matrix (nn, 1:6) = [ optimum_c(1,1) optimum_c(1,2) optimum_c(1,3) optimum_c(1,4)
        optimum_c(1,5) optimum_c(1,6) ];
        opt_c_matrix0 (nn, 1:6) = [ optimum_c0(1,1) optimum_c0(1,2) optimum_c0(1,3)
        optimum_c0(1,4) optimum_c0(1,5) optimum_c0(1,6) ];
        n_matrix = nn - ((n_of_option - 1 )*2);
        opt_obj_fun_vector(n_matrix) = opt_objective_fun;
        opt_obj_fun_matrix(n_of_option,n_matrix) = opt_objective_fun0;
        else
            fprintf(fileID,'\nobj-fun is imaginary!!ray
failure!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n');
            disp( opt_objective_fun );
disp( opt_objective_fun0 );
ffimag = ffimag + 1 ;
rayfailed_R1 (ffimag,1:6) = [x0(1,1) x0(1,2) x0(1,3) x0(1,4) x0(1,5) x0(1,6)];
end
end

% draw the histogram for 9 second part of 27 different GA options :
% nr is the Number_of_Runs
xcenters = 0:0.001:0.1 ;
if (n_of_option==1) ;
    figure(2);hold on;
    subplot(3,3,1);hold on;
    hist(opt_obj_fun_vector(1 , 1:1*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.sing.p&M.adfeas');
    figure(1);hold on;
    subplot(3,3,1);hold on;
    hist(opt_obj_fun_vector0(1 , 1:1*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.sing.p&M.adfeas');
elseif (n_of_option==2);
    figure(2);hold on;subplot(3,3,2);
    hist(opt_obj_fun_vector(1, 1+1:2*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.2p&M.adfeas');
    figure(1);hold on;
    subplot(3,3,2);hold on;
    hist(opt_obj_fun_vector0(1, 1:1*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.2p&M.adfeas');
elseif (n_of_option==3);
    figure(2);hold on; subplot(3,3,3);
    hist(opt_obj_fun_vector(1, 2*nr+1:3*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.arith&M.adfeas');
    figure(1);hold on; subplot(3,3,3);
    hist(opt_obj_fun_vector0(1, 2*nr+1:3*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.arith&M.adfeas');
elseif (n_of_option==4);
    figure(2);hold on; subplot(3,3,4);
    hist(opt_obj_fun_vector(1, 3*nr+1:4*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.hur&M.ga');
    figure(1);hold on; subplot(3,3,4);
    hist(opt_obj_fun_vector0(1, 3*nr+1:4*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.hur&M.ga');
elseif(n_of_option==5);
    figure(2);hold on; subplot(3,3,5);
    hist(opt_obj_fun_vector(1, 4*nr+1:5*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.scat&M.ga');
    figure(1);hold on; subplot(3,3,5);
    hist(opt_obj_fun_vector0(1, 4*nr+1:5*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.scat&M.ga');
elseif(n_of_option==6);
    figure(2);hold on; subplot(3,3,6);
    hist(opt_obj_fun_vector(1, 5*nr+1:6*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.inter&M.ga');
    figure(1);hold on; subplot(3,3,6);
    hist(opt_obj_fun_vector0(1, 5*nr+1:6*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.inter&M.ga');
elseif(n_of_option==7);
    figure(2);hold on; subplot(3,3,7);
    hist(opt_obj_fun_vector(1, 6*nr+1:7*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.singsp&M.ga');
    figure(1);hold on; subplot(3,3,7);
    hist(opt_obj_fun_vector0(1, 6*nr+1:7*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.singsp&M.ga');
elseif(n_of_option==8);
    figure(2);hold on; subplot(3,3,8);
    hist(opt_obj_fun_vector(1, 7*nr+1:8*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.twops&M.ga');
    figure(1);hold on; subplot(3,3,8);
    hist(opt_obj_fun_vector0(1, 7*nr+1:8*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.twops&M.ga');
elseif(n_of_option==9);
    figure(2);hold on; subplot(3,3,9);
    hist(opt_obj_fun_vector(1, 8*nr+1:9*nr),xcenters) ; axis([0 0.1 0 Num_of_run/10]);
    title('Cr.arith&M.ga');
xlabel('obj-fun-val');
ylabel('occurrence no.');
hold off;
hold off;
figure(1); hold on; subplot(3,3,9);
hist(opt_obj_fun_vector0(1, 8*nr+1:9*nr), xcenters); axis([0 0.1 0 Num_of_run/10]);
title('Cr.arith&M.ga');
xlabel('obj-fun-val');
ylabel('occurrence no.');
hold off;
hold off;
end
xlabel('obj-fun-val');
ylabel('occurrence no.');
end

% for GA without local
% draw the histogram for 9 different GA options
% nr is the Number of Runs
%........................................................................
% calculate the moments for 27 different ga parameter options
for jj = 1: number_of_options
    sum_fit_power1(jj) = 0;
    sum_fit_power2(jj) = 0;
    sum_fit_power3(jj) = 0;
end
for jj = 1: number_of_options
    for ii = ((jj-1)*nr)+1 : jj*nr
        sum_fit_power1(jj) = sum_fit_power1(jj) + opt_obj_fun_vector(1,ii) ;
        sum_fit_power2(jj) = sum_fit_power2(jj) + (opt_obj_fun_vector(1,ii)).^2 ;
        sum_fit_power3(jj) = sum_fit_power3(jj) + (opt_obj_fun_vector(1,ii)).^3 ;
    end
    av_fit_power1 (jj) = sum_fit_power1(jj) ./ nr;
    av_fit_power2 (jj) = sum_fit_power2(jj) ./ nr;
    av_fit_power3 (jj) = sum_fit_power3(jj) ./ nr;
end
% for GA without local ...........................
% calculate the moments for 9 different ga parameter options
for jj = 1: number_of_options
    sum_fit_power10(jj) = 0;
    sum_fit_power20(jj) = 0;
    sum_fit_power30(jj) = 0;
end
for jj = 1: number_of_options
    for ii = ((jj-1)*nr)+1 : jj*nr
        sum_fit_power10(jj) = sum_fit_power10(jj) + opt_obj_fun_vector0(1,ii) ;
        sum_fit_power20(jj) = sum_fit_power20(jj) + (opt_obj_fun_vector0(1,ii)).^2 ;
        sum_fit_power30(jj) = sum_fit_power30(jj) + (opt_obj_fun_vector0(1,ii)).^3 ;
    end
    av_fit_power10 (jj) = sum_fit_power10(jj) ./ nr;
    av_fit_power20 (jj) = sum_fit_power20(jj) ./ nr;
    av_fit_power30 (jj) = sum_fit_power30(jj) ./ nr;
end
for jj = 1: number_of_options
    if (jj==1); fprintf(fileID, 'op10: crossoverheuristic , mutationuniform,' populationsize, ', %d, ' Generations', ', %d, ' TolCon', ', 1e-20:
\ n , ', g , p);
    elseif (jj==2); fprintf(fileID, 'op11: crossoverscattered , mutationuniform,' populationsize, ', %d, ' Generations', ', %d, ' TolCon', ', 1e-20:
\ n , ', g , p);
    elseif (jj==3); fprintf(fileID, 'op12: crossoverintermediate,mutationuniform,' populationsize, ', %d, ' Generations', ', %d, ' TolCon', ', 1e-20:
\ n , ', g , p);
    elseif (jj==4); fprintf(fileID, 'op13: crossoversinglepoint , mutationuniform,' populationsize, ', %d, ' Generations', ', %d, ' TolCon', ', 1e-20:
\ n , ', g , p);
    elseif (jj==5); fprintf(fileID, 'op14: crossovertwo point    ,   mutationuniform,' populationsize, ', %d, ' Generations', ', %d, ' TolCon', ', 1e-20:
\ n , ', g , p);
    elseif (jj==6); fprintf(fileID, 'op15: crossoverarithmetic  ,  mutationuniform,' populationsize, ', %d, ' Generations', ', %d, ' TolCon', ', 1e-20:
\ n , ', g , p);
    elseif (jj==7); fprintf(fileID, 'op16: crossoverheuristic   ,mutationadaptfeasible,' populationsize, ', %d, ' Generations', ', %d, ' TolCon', ', 1e-20:
\ n , ', g , p);
    elseif (jj==8); fprintf(fileID, 'op17: crossoverscattered   ,mutationadaptfeasible,' populationsize, ', %d, ' Generations', ', %d, ' TolCon', ', 1e-20:
\ n , ', g , p);
    elseif (jj==9); fprintf(fileID, 'op18: crossoverintermediate,mutationadaptfeasible,' populationsize, ', %d, ' Generations', ', %d, ' TolCon', ', 1e-20:
\ n , ', g , p);
else
        fprintf(fileID, 'op19: crossoverarithmetic  ,  mutationadaptfeasible,' populationsize, ', %d, ' Generations', ', %d, ' TolCon', ', 1e-20:
\ n , ', g , p);
    endif
end
fprintf(fileID,'av_fit_power1 = %.7f ',av_fit_power1 (jj) )
fprintf(fileID,'av_fit_power2 = %.7f ',av_fit_power2 (jj) )
fprintf(fileID,'av_fit_power3 = %.7fn
',av_fit_power3 (jj) )
end
% for GA without local
for jj = 1: number_of_options
  if     (jj==1); fprintf(fileID, 'op10: crossoverheuristic , mutationuniform
',
        'populationsize',%d,"Generations", %d,"TolCon", 1e-
        20:\n ', g ,p);
  elseif (jj==2); fprintf(fileID,'op11: crossoverscattered   , mutationuniform
',
        'populationsize',%d,"Generations", %d,"TolCon", 1e-
        20:\n ', g ,p);
  elseif (jj==3); fprintf(fileID,'op12: crossoverintermediate,mutationuniform
',
        'populationsize',%d,"Generations", %d,"TolCon", 1e-
        20:\n ', g ,p);
  elseif (jj==4); fprintf(fileID,'op13: crossoversinglepoint , mutationuniform
',
        'populationsize',%d,"Generations", %d,"TolCon", 1e-
        20:\n ', g ,p);
  elseif (jj==5); fprintf(fileID,'op14: crossovertwopoint    ,   mutationuniform
',
        'populationsize',%d,"Generations", %d,"TolCon", 1e-
        20:\n ', g ,p);
  elseif (jj==6); fprintf(fileID,'op15: crossoverarithmetic  ,  mutationuniform
',
        'populationsize',%d,"Generations", %d,"TolCon", 1e-
        20:\n ', g ,p);
  elseif (jj==7); fprintf(fileID,'op16: crossoverheuristic   ,
',
        'mutationadaptfeasible,"populationsize",%d
        "Generations", %d,"TolCon", 1e-
        20:\n ', g ,p);
  elseif (jj==8); fprintf(fileID,'op17: crossoverscattered   ,
',
        'mutationadaptfeasible,"populationsize",%d
        "Generations", %d,"TolCon", 1e-
        20:\n ', g ,p);
  elseif (jj==9); fprintf(fileID,'op18: crossoverintermediate,
',
        'mutationadaptfeasible,"populationsize",%d
        "Generations", %d,"TolCon", 1e-
        20:\n ', g ,p);
  end
fprintf(fileID,'av_fit_power1 = %.7f  ',av_fit_power10 (jj) )
fprintf(fileID,'av_fit_power2 = %.7f  ',av_fit_power20 (jj) )
fprintf(fileID,'av_fit_power3 = %.7fn
',av_fit_power30 (jj) )
end
% print the list of initial points (i.e. the 6 curvatures) which make the
% program be failed in finding local optima, also the correct
% answers and their related objective function values
printf(fileID,'************************************nList of optimized curvatures with
their objective function values\n');
printf(fileID, 'Opt Radius:  R1  R2\n');
for ii = 1:num_of_correct_ans
  format short;
  fprintf(fileID, 'runno.=%d,objfun=%5.8f
                     %5.8f %5.8f %5.8f %5.8f
',ii, opt_obj_fun_vector(ii), opt_c_matrix(ii,1),
                     opt_c_matrix(ii,2),opt_c_matrix(ii,3),opt_c_matrix(ii,4),opt_c_matrix(ii,5),opt_c_matrix(ii,6)
                     );
end
if (ff~=0)
  fprintf(fileID,'list of initial points (including 6 curvatures)\nwhich make the program be failed in finding a local optimizer\n');
  fprintf(fileID, '\n\n',failed_initial_R1);
end
if (ffimag ~=0)
  fprintf(fileID,'list of initial points (including 6 curvatures)\nwhich make the program face ray failure \n');
  fprintf(fileID, '\n\n',rayfailed_R1);
end
fprintf(fileID,'Among %d number of running %d times fmin could not find any local optimizer\nand %f times ray-failure has been occured \n\n...Num_of_run , error_number_fmin, ffimagn, runtime - optime - time1;
fclose(fileID);
save('opt_part3with_withoutL.mat');
function [objective_fun] = objective_funcalzimax(c)
% this function calculate the objective function used in the optimization code.
[nsurf , d , n ] = data_of_optical_system ;
c(7)=0;
% calculate the BFL based on the curvatures each time.
% Take the data related to an optical system.
[Mtot = [ 0 0 0 ];
for ii =1: (nsurf-1)
    % calculate the BFL based on the curvatures each time.
    M_r = [ 1 -((n(ii+1)- n(ii)).*c(ii)) ; 0 1 ];
    M_t = [ 1 0 ; d(ii+1)./n(ii+1) ] ;
    Mtot = M_t * M_r * Mtot ;
end
ii = nsurf ;
Mtot = [ 1 -(( n(ii+1)- n(ii )).*c(ii )) ; 0 1 ] * Mtot ;
B = Mtot(1,1) ; A = - Mtot(1,2) ;
D = -Mtot(2,1) ; C = Mtot(2,2) ;
beta = ( B + A .* ( -500000000./n(1) ) ) .*(-1) ;
paraxial_focal_distance = (n(nsurf+1)./A ) .* (C - beta ) ;
% add a condition to reject the cases which their BFL is out of
% desired BFL interval. The desired bfl range is between 20-100 mm.
if ( bfl>20 & bfl <100);
d(nsurf+1) = bfl ;
delta = 0 ;
sum_deltas(1,3) = 0;
end
% convert the start point and the end point to the grid coordinates
[x_start , y_start , z_start]= grid_data_of_staring_point_ga;
number_of_gridpoints= num gridpoints;
for iiobj=1:num_of_objectpoints
    sum_x (1,iiobj) = 0 ;
    sum_y (1,iiobj) = 0 ;
    sum_deltas(1,iiobj) = 0 ;
end
for kk=1:number_of_gridpoints
    x_minus1 = x_start ;
    y_minus1 = y_start ;
    z_minus1 = z_start ;
    for ii = 1: (nsurf+1)
        z0 = z_minus1(1,kk) + d(ii) - N(1,kk).*delta;
        x0 = x_minus1(1,kk) + (L(1,kk).*z0 - z_minus1(1,kk));
        y0 = y_minus1(1,kk) + (M(1,kk).*z0 - z_minus1(1,kk));
        F = c(ii).*x0.*y0 ;
        G = N(1,kk) - c(ii).*L(1,kk).*x0 + M(1,kk).*y0;
        delta = the distance from p0p1
        delta = F ./ ( G + (G.*z - c(ii).*F).^(.5));
\[
x_1 = x_0 + L(1,kk) \cdot \Delta_x; \\
y_1 = y_0 + M(1,kk) \cdot \Delta_y; \\
z_1 = z_0 + N(1,kk) \cdot \Delta_z; \\
\]
% which gets the direction \(L',M',N\)
\[
cos I = (G^2 - c(ii) \cdot F)^{1/2}; \\
cos I' = (n(ii+1)^2 - (n(ii)^2 \cdot (1 - \cos I^2)))^{1/2}; \\
k = c(ii) \cdot (n^{prime}\cos I' - n(ii) \cdot \cos I); \\
L' = \frac{n(ii) \cdot L(1,kk) - k \cdot x_1}{n(ii+1)}; \\
M' = \frac{n(ii) \cdot M(1,kk) - k \cdot y_1}{n(ii+1)}; \\
N' = \sqrt{1 - (L'^2 + M'^2)}; \\
L(1,kk) = L'; \\
M(1,kk) = M'; \\
N(1,kk) = N'; \\
x_{minus1}(1,kk) = x_1; \\
y_{minus1}(1,kk) = y_1; \\
z_{minus1}(1,kk) = z_1; \\
\]
end
\[
x_{spotdiagram}(iiobj,kk) = x_0; \\
y_{spotdiagram}(iiobj,kk) = y_0; \\
sum_x(1,iiobj) = x_{spotdiagram}(iiobj,kk) + sum_x(1,iiobj); \\
sum_y(1,iiobj) = y_{spotdiagram}(iiobj,kk) + sum_y(1,iiobj); \\
end
\[
avg_x(1,iiobj) = \frac{sum_x(1,iiobj)}{number_of_gridpoints}; \\
avg_y(1,iiobj) = \frac{sum_y(1,iiobj)}{number_of_gridpoints}; \\
\]
\end{verbatim}

\begin{verbatim}
for iiobj = 1:3
sum_deltas(1,iiobj) = 0;
for kk = 1: number_of_gridpoints;
    delta_x(iiobj,kk) = x_{spotdiagram}(iiobj,kk) - avg_x(1,iiobj); \\
    delta_y(iiobj,kk) = y_{spotdiagram}(iiobj,kk) - avg_y(1,iiobj); \\
    distance_real_paraxial(iiobj,kk) = \sqrt{delta_x(iiobj,kk)^2 + delta_y(iiobj,kk)^2}; \\
    sum_deltas(1,iiobj) = sum_deltas(1,iiobj) + distance_real_paraxial(iiobj,kk); \\
end
spot_size(1,iiobj) = \frac{sum_deltas(1,iiobj)}{number_of_gridpoints}; \\
\]
\end{verbatim}

\begin{verbatim}
if (iiobj == 1) \\
    spot_size1 = spot_size(1,iiobj); \\
elseif (iiobj == 2) \\
    spot_size2 = spot_size(1,iiobj); \\
elseif (iiobj == 3) \\
    spot_size3 = spot_size(1,iiobj); \\
end
\end{verbatim}

\begin{verbatim}
end
\end{verbatim}

\begin{verbatim}
% calculate efl
[x_minus1, y_minus1, z_minus1, L, M, N] = data_of_staring_point_efl; \\
x_minus1 = 0; \\
y_minus1 = 0.0002; \\
z_minus1 = 0; \\
L = 0.00; \\
M = 0; \\
N = 1 - ((L^2 + M^2)^1/2); \\
delta = 0; \\
\end{verbatim}

\begin{verbatim}
for ii = 1: (nsurf+1) \\
    z0 = z_minus1 + d(ii) - N \cdot \Delta_z; \\
x0 = x_minus1 + (L/N) \cdot \Delta_z; \\
y0 = y_minus1 + (M/N) \cdot \Delta_z; \\
    if (ii < (nsurf+1)) \\
        F = c(ii) \cdot \sqrt{x0^2 + y0^2}; \\
        G = N - c(ii) \cdot (L \cdot x0 + M \cdot y0); \\
        \text{\% delta is the distance p0P1} \\
        delta = F \cdot \sqrt{G + (G \cdot F^2 - c(ii) \cdot F)^{1/2}}; \\
x1 = x0 + L \cdot \Delta_z; \\
y1 = y0 + M \cdot \Delta_z; \\
z1 = z0 + N \cdot \Delta_z; \\
    \cos i = (G^2 - c(ii) \cdot F)^{1/2}; \\
\end{verbatim}
\[ n_{\prime \cos I_{\prime}} = \left( (n_{\text{ii}+1})^2 - (n_{\text{ii}})^2 \cdot (1 - \cos I \cdot 2) \right) ^{\frac{1}{2}}; \]
\[ k = c_{\text{ii}} \cdot (n_{\prime \cos I_{\prime}} - n_{\text{ii}} \cdot \cos I); \]
\[ L_{\prime} = \left( n_{\text{ii}} \cdot M - k \cdot x_1 \right) / n_{\text{ii}+1}; \]
\[ M_{\prime} = (n_{\text{ii}} \cdot M - k \cdot y_1) / n_{\text{ii}+1}; \]
\[ N_{\prime} = \left( 1 - (L_{\prime}^2 + M_{\prime}^2) \right) ^{\frac{1}{2}}; \]

% end of calculation for one travelling + refraction from one surface
% start calculation for the next travelling + refraction
% L = L_{\prime}; M = M_{\prime}; N = N_{\prime};
if (ii==nsurf);
\[ N_{\text{of_last_surf}} = N_{\prime}; \]
\[ y_{\text{minus1_of_lastsurf}} = y_1; \]
\[ x_{\text{minus1}} = x_1; \]
\[ y_{\text{minus1}} = y_1; \]
\[ z_{\text{minus1}} = z_1; \]
end
end

nu_angle = \arccos(N_{\text{of_last_surf}});
enuf = (0.0002 ./ nu_angle);
bfl2 = y_{\text{minus1_of_lastsurf}} / nu_angle

% end of efl calculation
% w(i) are the weights for 3 spot sizes and for the efl constraint.
% the constraint on the efl is to keep the efl of system fixed around 49.99
% efl_0 = 49.99;
w1 = 1; w2 = 1; w3 = 1; w4 = 0.1;
objective_fun = w1 \cdot \text{spot_size1} + w2 \cdot \text{spot_size2} + w3 \cdot \text{spot_size3} + w4 \cdot ((efl - efl_0)^2);
else
objective_fun = 5;
end
end
References

[1] Florian Bociort, 120009685_E-EOE_R1_111102_spi, Delft University of Technology, the Netherlands.


[7] Optical design program, Zemax, user`s guide version 9.0, Tucson, AZ 85731-8228 USA.

[8] Composite optical elements. MIT 2.71/2.710. MIT Open Course Ware.


[17] Mohsen Davarynejad, ABC optimization programming in MatLab.


