PRL Summer Internship SIP-2019

Reg. No. : 19STS0310

Optical properties of different glass types and wave aberrations

Author
GOTI RIDHESH
University : Pandit Deendayal Petroleum University
Course : B.Sc. Physics (Hons.)

Supervisor
Mr. VAIBHAV DIXIT

July 7, 2019

Contents

Optical properties of glasses

Ridhesh Goti

June 2019

1 Introduction

In this project, I have done work in three parts because it has three different objectives. First part was basics of starting the project. I had created one gitlab account in which I can upload my latest files and work into the remote server for backups. I had learned the gitlabs basics and git bash commands. After that, My **first objective** of the project was to find the dispersion relation. Dispersion relation is expression between refractive index of medium and wavelength. To achieve goal, I had set of catalogues where I can find the all constants values and necessary data of glass types which I used in the dispersion relation to find the refractive index vs wavelength plot. I had extracted 10 to 11 different glass types from different glass catalogues and find the trends.

Second objective of the project was to analyze the temperature effects on the refractive index of the different glass types. Temperature changes the medium of the glass and that gives the variations in the refractive index. I had plotted for three relation, (1) change of rate of absolute refractive index vs temperature (2)change of rate of relative refractive index vs temperature and (3) refractive index vs temperature. Last part of the project was to verify all data which I had calculated through Python programming. From reference data, I got the refractive index of selected 50 glass types from different catalogues and calculate the difference between the values. I also verified for temperature with other reference data with 5 different glass types. **Third objective** of the project was to make model of the different optical system to find the different aberrations. Aberrations are occurred due to different rays travel different paths. Because of this variation, it creates the wave-front error which is phase variations with respect to the spherical wave. In the astronomy, the wave aberration in the telescope is main concern of the research. To reduce that we can create the arbitrary shape lens and different combinations of the optical elements. I created the model to find OPD(optical path difference) in the given system with Python programming. OPD is further using in to find phase change. I also verified these data with reference data. And There is another way to find wave front error is Zernike polynomials which I discussed in the further sections. All three works are interrelated because each parts need data of refractive index with specific wavelength to find temperature variations or optical path different.

2 Gitlab and Git commands

Gitlab is best platform for development of project when multiple users are involved. Gitlab provides separate space for project. It also provides privacy levels from private to public. There is one SSH key which provides users to use whenever they want to log in into server. SSH key is generated through Git bash command line. RSA,ED25519 are types of SSH keys which are supported by Gitlab. To generate new RSA key,

$$ssh-keygen-o-trsa-b4096-C"email@example.com"$$

is command. For ED25529,

$$ssh-keygen-ted25519-C"email@example.com"$$

is command. Once command is entered in Git bash, key is generated automatically. Key should be copied to Gitlab account by command,

$$cat\ /.ssh/id\_ed25519.pub|clip$$

After making account in gitlab, we can make first new project and then add directory and files from computer and edit on the platform. If there are so many users in project then we can make group in gitlab and add them with selected opacity of project. There two types of repository (1) local repository which means that data from any computer client or PC. We can extract data from local repository to gitlab server with commands. This gives the second terminology called as (2) Remote repository which means that data on server or gitlab account. If someone from remotly change the data of gitlab in group project then we can update our local repository easily with one line command in git bash called as **pull**,

git pull

Before doing this, configuration on gitlab is very important. First to check git has installed or not by command,

git –version

In output it'll reply about version of git bash command. Now set the email id and username for account for connection between remote and local repository,

git config −−global user.name "USER_NAME"
git config −−global user.email "email_id@gmail.com"

For verification for email and user name

git config −−global user.name
git config −−global user.email

First initializing the git through,

git init

To start working in locally, we have to clone whole repository in remote to local repository. This is done through, **Clone** is done through two ways,
(1) via link

git clone https://gitlab.com/gitlab-org/gitlab-ce.git

(2) via SSH key link

git clone git@gitlab.com:gitlab-org/gitlab-ce.git

This information is given in the gitlab remote repository data. There are branches in gitlab which is useful feature in the gitlab. With different branches, different users could work on different **branches** and after editing in the files and directory, they can **merge** the files and make unified file. Also merge undo request is also exist in the gitlab. To go in different branches,

git checkout -b ”branch name”

Whenever someone changes the details and path, we can check easily by,

git status

We can also edit the files from local to remote server by two lines. First we have to specify filename which has to be modified and gitlab would put that file into one imaginary server space because if we want to go back and undo to update that file then gitlab gives two options. After selecting files, we can **commit** the command by,

git add ”file name” git commit -m ”comments”

We can also write comments about changes for every date and time to check progress. Comments are written in commit part. We can add all changes at instant through . sign.

git add .

Now we have to just **push** the data from local repository to remote repository (here gitlab) by,

git push -u origin master

Here **master** is main branch when we had made the account and it is central part of works. We can also change data from different branches by change the path. There is another command which gives the all history of our commits inside local to remote or remote to local.

git log

There is also undo option for returning to unmodified files which have committed to remote server. We can undo this by,

git revert ”number”

number is code which is given to that commit.

Sometimes there is slightly changes in code or files data which very subtle work to find that. We can find that difference or changes by,

```
git diff
```

But the output of **diff** is not understandable so we can download the editor like **KDiff** in the PC and compare at most three files simultaneously in that editor. So this is basics about the gitlab and git bash command line.

3 Aberrations in glass

There are so many aberrations in glass. From seidel aberration theory there are total five aberrations. There is another aberration called as chromatic aberration.

3.1 Chromatic aberration

Different wavelengths have different refractive index in glass and different refractive index means it has different focusing power for same glass type. Therefore, it will not give white image of the object. This aberration contains all wavelengths and make image at different focuses. Here in 1 is without chromatic

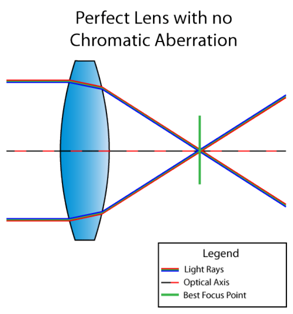

Figure 1: Without chromatic aberration

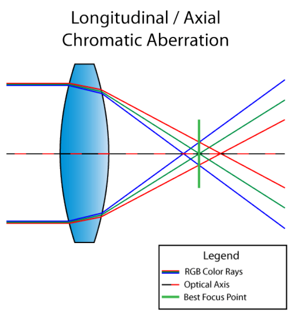

Figure 2: With chromatic aberration

aberration and all wavelengths concentrate on one focus point which means that glass has low dispersion. Therefore, refractive index does not change very much with different wavelengths. And 2 is for chromatic aberration where all wavelengths have different behaviour in glass therefore aberration occur in the glass.

3.2 Monochromatic aberration

There are other aberrations with monochromatic light.

3.2.1 Spherical aberration

In this aberration, one monochromatic light is sent to the lens with parallel to optical axis of the lens. The rays near to the optical axis get bend less than rays far from the optical axis. In the 5, to reduce spherical aberration, lens is made

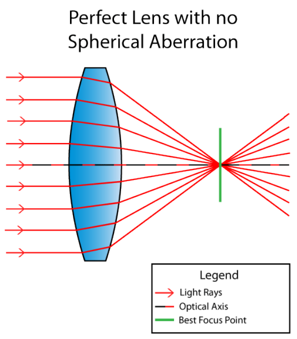

Figure 3: Without chromatic aberration

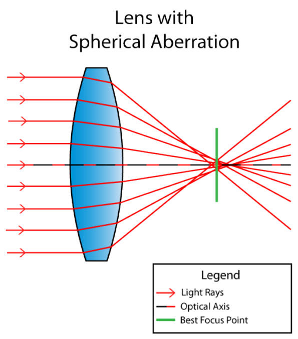

Figure 4: With chromatic aberration

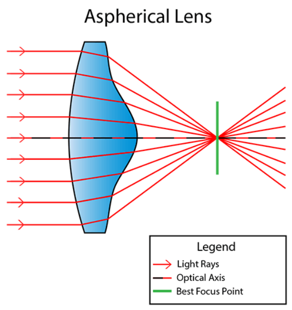

Figure 5: Without chromatic aberration

with extra optical path which gives the central rays more time to pass through glass and bend more.

3.2.2 Coma aberration

An off-axis effect which appears when a bundle of incident rays all make the same angle with respect to the optical axis (source at infinite). Rays are brought to a focus at different points on the focal plane found in lenses with large spherical aberrations an off-axis object produces a comet-shaped image. In the 6 the shape of the image is like comet because of focusing on focal plane at different locations. Comatic aberration is similar to the spherical aberration but this is off axis rather than on optical axis.the peripheral rays produce the smallest image (least magnification) and the coma aberration sign is said to be negative.

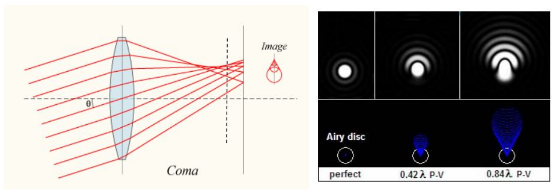

Figure 6: Coma aberration

In contrast, when the peripheral rays are focused further down the axis and produce a much larger image (greater magnification), the aberration is termed positive. To reduce the coma aberration, peripheral rays should be removed.

3.2.3 Astigmatism

Astigmatism occurs when there are not spherical curvature. That means curvature in one plane is different than perpendicular plane of it. In the 7, aberration is astigmatism. The black-line in the figure is optical axis. And lens is viewed from some angle. aberration of astigmatism by supposing that the lens has a different focal length in one plane than in the other. Due to the different focal plane in vertical and horizontal, sphere looks like circle or ellipse shape. Although different radii of curvature in different planes is not the usual cause

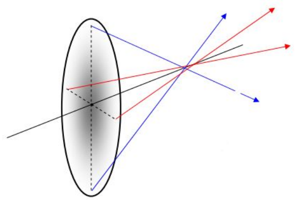

Figure 7: Astigmatism

of astigmatism, there is an exception - namely, the human eye. If the radii of curvature of the cornea, or of the lens, is different in different planes, then the image on the retina will be astigmatic even on-axis.

3.2.4 Curvature of field

Suppose the spherical, coma, astigmatic aberration is resolved but there will be another aberration called as curvature of field. This occur due to focusing of light at different point on one curvature focal curve. In the 8, all focused points are on one curvature plane instead of plane. Focal surface also called as **petzval** surface. One effective way of dealing with this problem, particularly if detector is a flexible film, is to shape the film-holder so that the film fits along the Petzval surface.

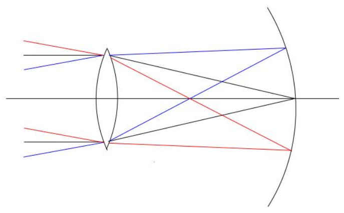

Figure 8: Curvature of field

3.2.5 Distortion

The magnification of an image is image distance divided by object distance, and image distance is different off-axis than on-axis, so the image magnification varies with distance from the axis. If there sqaure object then its distortion look

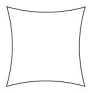

Figure 9: Pincushion shape

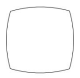

Figure 10: Barrel shape

like in 9 and 10. Here two figures are different depends on certain conditions. Off-axis, the image distance is less than the object distance,so the magnification is less off-axis than on-axis. Barrel distortion results. Off-axis, the image distance is greater than the object distance, so the magnification is greater off-axis than on-axis. Pincushion distortion results.

4 Dispersion of light in glasses

Refraction is bend of the light through out the medium and it depends on the material or glass type. Refractive index is calculated by ratio of speed of light in medium and speed of light in vacuum space. In water, Refractive index is 1.33 means that light speed slows 33 % of original value. Refraction of the light is expressed through snell's law of refraction which is given in the figure 11. In the

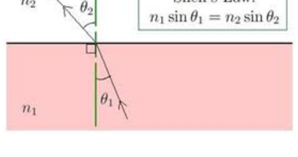

Figure 11: Snell's refraction law

figure the angle is changing after the refraction but refractive index is assumed to be constant. For constant value of refractive index, the angle would also be constant with different wavelengths. But here it is not case becuase refractive index of medium depends on the wavelength of light.

Refractive index depends on the wavelength of the light. Different wavelengths have different refractive index and bend with different angles. For chromatic rays, all colors are refracted with different directions because of the phenomena. This called as **dispersion of light** in medium. First approximation for relation is made by cauchy approximation. Cauchy expression is given by following equation,

$$n(\lambda) = B + \frac{C}{\lambda^2} \tag{1}$$

In this (1), B and C are constants and depend on the material type. Formation of rainbow and other phenomena are due to the dispersion.
Light being electromagnetic oscillations, the different colours have different wavelengths and different frequencies in vacuum for all of them move with the same speed c. Whenever light enters a dielectric medium this separation happens and the phenomenon is known as dispersion. There are more **cauchy expression** for dipersion,

$$n(\lambda) = B + \frac{C}{\lambda^2} + \frac{D}{\lambda^4} \tag{2}$$

These all constants are determined by experimentally and give the proper relation between refractive index and wavelength. The first two terms would suffice to give an accurate value of n. The derivative of (2) is given by,

$$\frac{dn}{d\lambda} = -\frac{B}{\lambda^3} \tag{3}$$

This is **normal dispersion** behaviour for light. But this behaviour is valid only for certain wavelength range. Outside the range there is absorption region where refractive has not valid value . refractive index suddenly decreases very fast and does not obey the Cauchy's law, if we approach the absorption region. Further increasing the wavelength once again refractive index becomes large. Again the behaviour is quite similar to the visible region for the increase in wavelength. This type of behaviour known as **anomalous dispersion**. There are many absorption region and with same repetition of pattern. The refractive index couldn't be found in absorption region because substance absorbs that particular wavelength completely because of the resonance effects in electronic spectra and vibration spectra of material. +

There are important facts about the normal dispersion is,
(1) refractive index of substance increases with decreasing of wavelength (2) The rate of increase becomes greater at shorter wavelength. In the 12, there is band between visible and infrared region with following cachy expression. To derive the expression for dispersion, sellmeier assumed that all elastically bound

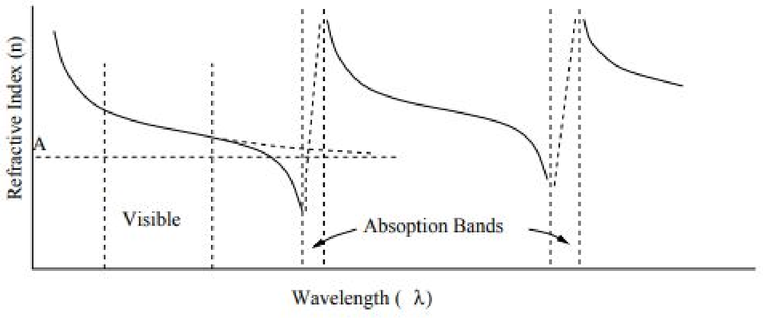

Figure 12: Anomalous dispersion

particles in the medium oscillate with a natural frequency ω_0 which correspond to a wavelength λ_0 in the vacuum. The final equation is looks like,

$$n^2 = 1 + \frac{A\lambda^2}{\lambda^2 - \lambda_0^2} \tag{4}$$

In (4), when $\lambda = \lambda_0$, the trend would diverge and give the same result as in figure 12. To explain many absorption bands, we have to assume different species of electrons with different natural frequencies ω_j corresponding to wavelengths λ_j in the substance and then

$$n^2 = 1 + \Sigma_j \frac{A_j\lambda^2}{\lambda^2 - \lambda_j^2} \tag{5}$$

5 Elementary theory of dispersion

5.1 Theory and derivation of sellemier equation

Effect of electromagnetic field on dielectric medium will be given by electromagnetic theory,

$$D = \varepsilon E \tag{6}$$

. When there is external electric field in the dielectric medium the total displacement is,

$$D = \varepsilon_0 E + P = \epsilon_0(1 + \chi)E \tag{7}$$

. The factor χ is called the electric susceptibility of the substance and the factor $(1 + \chi) = \epsilon_r$ is nothing but the relative permittivity. And vector P is polarization vector. When an electric field is applied to a substance electrons of the molecules of that substance are displaced from there mean position as shown in the fig 13. Electron would not leave molecule but oscillate with that frequency and also generate dipole moment inside medium. Polarization per unit volume is given by,

$$P = -Nex \tag{8}$$

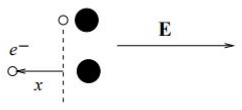

Figure 13: Oscillation of electron from mean position

In equation (8), N is total number of electrons and x is displacement of electron. Also electric field is in x- direction and have negative sign because of electron negative charge. Here we are ignoring oscillation of positive ions and electron will oscillate with certain frequency ω_0. But light has oscillate electric field then displacement would also oscillate with that time period. Here dampening force comes in equation because of the shielding effects, other atoms effects on electron which try to bound more in atom. For forced oscillation differential equation is given by ,

$$\ddot{x} + r\dot{x} + \omega_0^2 x = -\frac{eE}{m} \tag{9}$$

similarly we can write equation (9) as,

$$\ddot{P} + r\dot{P} + \omega_0^2 P = -\frac{Ne^2 E}{m} \tag{10}$$

From maxwell's equations, $\nabla \times E = -\dot{B}$ and $\nabla \times H = \dot{D}$, $\nabla \cdot E = 0$

$$\begin{aligned} \nabla \times \dot{B} &= \mu_0 \ddot{D} \\ \nabla \times (\nabla \times E) &= -\mu_0\epsilon_0 \ddot{E} - \mu_0 \ddot{P} \\ \nabla^2 \cdot E &= \mu_0\epsilon_0 \ddot{E} + \mu_0 \ddot{P} \end{aligned}$$

From all these equations (10) we can get,

$$\ddot{P} = \frac{1}{\mu_0}(\nabla^2 \cdot E - \frac{1}{c^2}\ddot{E}) \tag{11}$$

Differentiating equation (10) w.r.t time,

$$(\frac{\partial^2}{\partial t^2} + r\frac{\partial}{\partial t} + \omega_0^2)\ddot{P} = \frac{Ne^2}{m}\ddot{E} \tag{12}$$

Using equation (11) eliminating P in equation (12),

$$(\frac{\partial^2}{\partial t^2} + r\frac{\partial}{\partial t} + \omega_0^2)(\nabla^2 \cdot E - \frac{1}{c^2}\ddot{E}) = \mu_0\frac{Ne^2}{m}\ddot{E} \tag{13}$$

when a plane electromagnetic wave polarised along the x direction and E will become $E_0 = e^{i(kz-\omega t)}$ and put in equation (13),

$$(\omega^2 - ir\omega + \omega_0^2)(-k^2 + \frac{\omega}{c^2}) = -\frac{\mu_0 Ne^2\omega^2}{m} \tag{14}$$

Further arrangements in equation (14),

$$k^2 = \frac{\omega^2}{c^2}(1 + \frac{\mu_0 c^2 N e^2}{m(\omega_0^2 - ir\omega - \omega^2)}) \tag{15}$$

In the equation (15) $v = \frac{\omega}{k}$ is phase velocity and $n = \frac{c}{v}$ is refractive index of the medium. So equation will become,

$$n^2 = (1 + \frac{\mu_0 c^2 N e^2}{m(\omega_0^2 - ir\omega - \omega^2)}) \tag{16}$$

For refractive index being as real,$\omega_0^2 - \omega^2 >> r\omega$ which means that dampening is very slight,

$$n^2 = (1 + \frac{\mu_0 c^2 N e^2}{m(\omega_0^2 - \omega^2)}) \tag{17}$$

relation between wavelength and angular frequency is $\lambda = \frac{2\pi c}{\omega}$ then

$$n^2 = (1 + \frac{\mu_0 N e^2 \lambda_0^2 \lambda^2}{4\pi^2 m(\lambda^2 - \lambda_0^2)}) \tag{18}$$

and finally by assuming constants equation will look like sellmeier formula for dispersion.

5.2 Abbe number

Abbe number is measurement of the strength of dispersion by transparent material. Low abbe number means that dispersion is high vice versa. Abbe number of the material is defined by equation,

$$V_D = \frac{n_D - 1}{n_F - n_C} \tag{19}$$

In equation (19) $n_D, n_F, and n_c$ are refractive index of material at wavelengths of Fraunhofer D,F and C lines(589.3 nm, 486.1 nm, 656.3 nm respectively). Abbe number are useful number for classifying the different glasses. For example, **flint glass** has abbe number V < 50 and **crown glass** has V > 50. Typical values of V range from around 20 for very dense flint glass, around 30 for **polycarbonate plastics**, and up to 65 for very light crown glass, and up to 85 for **fluor-crown glass**.
Abbe numbers are only a useful measure of dispersion for visible light, and for other wavelengths, or for higher precision work, the group velocity dispersion is used. An Abbe diagram is produced by plotting the Abbe number of a material versus its refractive index. Glasses can then be categorised by their composition and position on the diagram.

6 ZEMAX - glass catalogs

6.1 Use of catalogs

In dispersion theory, different wavelengths have different refractive index. There are experimentally verified constants values for all dispersion formula and give the relation between refractive index and wavelengths. There are some catalogs made by manufacturer of glass which provides the details of the glass. From data we can extract the data of glass and its behaviour.

6.2 Example of catalogs

These catalogs provides the details of each glass type with its properties like Abbe number, Thermal coefficients, Transmission and Thermal expansion etc. There are more catalogs for different glasses depend on the manufacturing companies of that glass type. For example, **schott** is catalog name and each glasses in that catalog made by schott company. This catalog contains every details for analysis of glass type. There are many catalogs, **ohara, hoya, birefringent, hikari, sumita, misc, infrared** etc. Each catalogs contain the details of specific glass type. Application of this different glass types require for different fields like in telescope, glasses in spectacle, windows etc. For that different catalogs are required.

6.3 Extension of catalogs

Extension of the catalog is **AGF** which is basically ANSII file. It could open with notepad with texts. ZEMAX software uses this catalogs as AGF and converts into BGF (Binary) file for more efficiency of processing. Here We are using AGF file as texts and extract data from it through **python** platform. We can edit the catalogs as per latest data.

6.4 Availability

Catalogs are easily available on online. Each manufacturer provide the catalog of its glass types. We can download the catalogs by typing **catalogs-name.agf**. Here is link for ohara catalog,

https://www.oharacorp.com/catalog.html

6.5 How to read catalogs

In the catalog, first line has format like this,

$$CC < catalogname >$$

This tells about the name of catalog. Second line of the AGF file would be,

$$NM < glassname >< dispersionformula - number >< MIL >< N_d >< V_d >< Excludesub >< Status >< Meltfrequency >$$

In this line MIL represents the reference number for each glass type.N_d is refractive index of glass at wavelength 0.587 micrometers and it is just reference number. V_d is also just reference number and it is abbe number of glass. **Exclude Sub** represents that if it is checked then that glass type is not selected for conversion from model to real glass in application. Exclude sub has only two values, 1 for checked and 0 for unchecked. Status indicates the availability of that glass type in market. There four types of status, **Standard, Preferred, Obsolete, Special**. Standard glass generally available in purchase. Preferred glasses are frequently melted and more likely available on demand. Obsolete glasses are no longer available but may be available on demand. Melt frequency indicates that how frequently glass is melted during manufacturing of that glass. 1 represents the high frequency, 2 represents very often and 5 represents infrequently. Glass name part give the glass type for example N-BK7, YGH52 etc. Dispersion formula number is number given for each dispersion formulas. There are total 13 formulas and each formulas has its corresponding number. The dispersion formula number is 1 for Schott, 2 for Sellmeier 1, 3 for Herzberger, 4 for Sellmeier 2, 5 for Conrady, 6 for Sellmeier 3, 7 for Handbook of Optics 1, 8 for Handbook of Optics 2, 9 for Sellmeier 4, 10 for Extended, 11 for Sellmeier 5, and 12 for Extended 2 and 13 for Extended 3. Third line of the file is

$$GC < individualglasscomment >$$

This line gives the specifications or any extra comments of that glass type. Fourth line is

$$ED < TCE(-30to70) >< TCE(100to300) >< density >< dPgf >< ignorethermalexpansion >$$

TCE is thermal coefficient expansion and it is used for modelling of linear expansion of glass with temperature. For different temperature ranges, TCE have different values and therefore there are two columns of TCE one for -30 to 70 and second for 100 to 300. Density is density of glass type. dPgf gives the deviation of relative partial dispersion from normal D line. Ignore thermal expansion is useful for non solids materials. For example, gases and liquids have not thermal expansion but its mounted material would solid. So thermal modelling is different for solids and gases. For gases and liquids, edge effects of mounted material is consider. Fifth line is

$$CD < dispersioncoefficients1-10 >$$

This gives the all required coefficients of given dispersion formula. Sixth line is

$$TD < D0 >< D1 >< D2 >< E0 >< E1 >< Ltk >< Temp >$$

TD is useful for thermal analysis of glass for example how refractive index of particular wavelength is changed with temperature. Seventh line is

$$OD < relcost >< CR >< FR >< SR >< AR >< PR >$$

Relative cost is relatively to N-BK7 glass type. **CR** stands for climate resistance, **FR** is stain resistance, **SR** is acid resistance, **AR** is alkali resistance, **PR** is phosphate resistance. Eigth line is

$$LD < minlamda >< maxlamda >$$

This gives the validity range of wavelengths which could give proper answer to dispersion formula. Last line is

$$IT < lamda >< transmission >< thickness >$$

This represents the transmissivity of particular wavelength with given thickness of glass type. In catalog, if there is -1 in data that means that data is not available.

6.6 Dispersion formulas in glass catalogs

There are total 13 dispersion formulas which are used by different different catalogs for each glass types. With different dispersion formula, refractive index of each wavelength in the light will disperse differently and rise the aberrations. Here is first formula,
(1)schott

$$n^2 = a_0 + a_1\lambda^2 + a_2\lambda^{-2} + a_3\lambda^{-4} + a_4\lambda^{-6} + a_5\lambda^{-8}$$

(2)sellmeier1

$$n^2 - 1 = \tfrac{K_1\lambda^2}{\lambda^2 - L1} + \tfrac{K_2\lambda^2}{\lambda^2 - L2} + \tfrac{K_3\lambda^2}{\lambda^2 - L3}$$

(3)sellmeier2

$$n^2 - 1 = A + \tfrac{B_1\lambda^2}{\lambda^2 - \lambda_1^2} + \tfrac{B_2}{\lambda^2 - \lambda_2^2}$$

(4)sellmeier3

$$n^2 - 1 = \tfrac{K_1\lambda^2}{\lambda^2 - L1} + \tfrac{K_2\lambda^2}{\lambda^2 - L2} + \tfrac{K_3\lambda^2}{\lambda^2 - L3} + \tfrac{K_4\lambda^2}{\lambda^2 - L4}$$

(5)sellmeier4

$$n^2 - 1 = A + \tfrac{B\lambda^2}{\lambda^2 - C} + \tfrac{D\lambda^2}{\lambda^2 - E}$$

(6)sellmeier5

$$n^2 - 1 = \tfrac{K_1\lambda^2}{\lambda^2 - L1} + \tfrac{K_2\lambda^2}{\lambda^2 - L2} + \tfrac{K_3\lambda^2}{\lambda^2 - L3} + \tfrac{K_4\lambda^2}{\lambda^2 - L4} + \tfrac{K_5\lambda^2}{\lambda^2 - L5}$$

(7)herzberger

$$n = A + BL + CL^2 + D\lambda^2 + E\lambda^4 + F\lambda^6$$

$$L = \tfrac{1}{\lambda^2 - 0.0028}$$

(8)conrady

$$n = n_0 + \frac{A}{\lambda} + \frac{B}{\lambda^{3.5}}$$

(9)Handbook of optics 1

$$n^2 = A + \frac{B}{\lambda^2 - C} - D\lambda^2$$

(10)Handbook of optics 2

$$n^2 = A + \frac{B\lambda^2}{\lambda^2 - C} - D\lambda^2$$

(11)Extended 1

$$n^2 = a_0 + a_1\lambda^2 + a_2\lambda^{-2} + a_3\lambda^{-4} + a_4\lambda^{-6} + a_5\lambda^{-8} + a_6\lambda^{-10} + a_7\lambda^{-12}$$

(12)Extended 2

$$n^2 = a_0 + a_1\lambda^2 + a_2\lambda^{-2} + a_3\lambda^{-4} + a_4\lambda^{-6} + a_5\lambda^{-8} + a_6\lambda^{4} + a_7\lambda^{6}$$

(13)Extended 3

$$n^2 = a_0 + a_1\lambda^2 + a_2\lambda^4 + a_3\lambda^{-2} + a_4\lambda^{-4} + a_5\lambda^{-6} + a_6\lambda^{-8} + a_7\lambda^{-10} + a_8\lambda^{-12}$$

These are the total 13 formulas for dispersion relation of the glass type. All the corresponding constants are given in the respective catalogs. From these formulas we can find the relation between the refractive index and different wavelengths.

7 Configure optics as new virtual environment and packages

To extract data from catalogs with python, **spyder** is good platform for scientific developments. We have to download anaconda navigator. From anaconda command prompt, environment is good thing to start new project. New environment provides new separate working space which does not affect the other environment or base space. We can download different packages for it. For this project **optical properties of glasses**, I made the optics environment in anaconda and start new programming in spyder editor which is widely useful in scientific tools. To make new environment, we have to just type

```
conda create -n optics python=3.7 anaconda
```

in anaconda command prompt. Once the environment is created, activation of it is done by,

```
activate optics
```

New packages could be downloaded with command,

```
conda install -n optics package-name
```

In my optics virtual environment, I have downloaded **numpy, matplotlib, os, scipy etc** in the optics virtual environment with same command. We can import the packages by its requirement and name as short form for further using of packages. This packages contain well defined functions. To import the packages from environment by typing,

import numpy as np

to plot graphs, we have to call matplotlib package differently,

import matplotlib.pyplot as plt

8 Extract data from catalogs by python

Extracting data from catalogs is done by looping process. We have to check each and every line in catalogs to find specific glass type and its details. First we have to get input data from user for glass name with exact code. After using matplotlib package we can find the plot of the refractive corresponding to different wavelengths. Same goes for all analysis in the catalogs. There are thermal coefficients given which are useful in the thermal analysis of the glass type.

9 Thermal coefficients formulas and theory

The refractive index of optical glasses change with temperature, the extend of changing RI is depended on the glass type and on the wavelength. It also depends on pressure of surroundings so that would give the two part of refractive index **(1)Absolute refractive index** and **(2) Relative refractive index**. We can find the rate of change of refractive index from temperature range -100 C^o to 140 C^o. We need thermal coefficients for calculation of changes of refractive index with temperature. These coefficients are given in aforementioned catalogs. In the section 6.5, TD line gives the details of thermal coefficients. there are special glasses with negative temperature coefficients in an optical system can help to keep wave front deformations caused by temperature changes on a minimum level. Such glasses are called **athermal glasses**.
From fundamental equation of refractive index relative to vacuum is given by generally,

$$n_{abs}^2(\lambda) = 1 + a \cdot \frac{N}{V} \cdot f \cdot \frac{\lambda^2 \cdot \lambda_0^2}{\lambda^2 - \lambda_0^2} \tag{20}$$

In the equation (20), $n_{abs}^2(\lambda)$ is refractive index in vacuum at certain temperature. Rest of are constants depending on glass type. Differentiating this equation with respect to temperature will give simplify form of the derivative equation is,

$$\frac{dn_{abs}(\lambda, T)}{dT} = \frac{n^2(\lambda, T_0) - 1}{2 \cdot n(\lambda, T_0)} \cdot (D_0 + 2 \cdot D_1 \cdot \Delta T + 3 \cdot D_2 \cdot \Delta T^2 + \frac{E_0 + 2 \cdot E_1 \cdot \Delta T}{\lambda^2 - \lambda_{TK}^2}) \tag{21}$$

In the equation (21), T_0 is reference temperature which is also given in the catalogs. T is variable of temperature in celcius. ΔT is difference between variable and reference temperature. And $D_0, D_1, D_2, E_0, E_1, \lambda_{Tk}$ are constants depend on glass type. The refractive index values given in catalogs are in 101330 Pa pressure and they are called as relative refractive index. So $n(\lambda, T_0)$ is refractive index in catalog at reference temperature T_0 and pressure is 101330 Pa.The change is refractive index with temperature is given by,

$$\Delta n_{abs}(\lambda, T) = \frac{n^2(\lambda, T_0) - 1}{2 \cdot n(\lambda, T_0)} \cdot (D_0 \cdot \Delta T + \cdot D_1 \cdot \Delta T^2 + \cdot D_2 \cdot \Delta T^3 + \frac{E_0 \dot{\Delta} T + \cdot E_1 \cdot \Delta T^2}{\lambda^2 - \lambda_{TK}^2}) \tag{22}$$

$$n_{abs}(\lambda, T) = n_{abs}(\lambda, T_0) + \Delta n_{abs}(\lambda, T) \tag{23}$$

$$n_{rel}(\lambda, T) = \frac{n_{abs}(\lambda, T)}{n_{air}(\lambda, T, p)} \tag{24}$$

From all equations, (22),(23), and (24) we can get relation between the refractive index and temperature with specific pressure. We can get equation for derivative of relative refractive index with temperature,

$$\frac{dn_{rel}(\lambda, T)}{dT} = \frac{\frac{dn_{abs}(\lambda, T)}{dT} - n_{rel}(\lambda, T) \cdot \frac{dn_{air}(\lambda, T, p)}{dT}}{n_{air}(\lambda, T, p)} \tag{25}$$

In equation (25), we can find $fracdn_{rel}(\lambda, T)dT$ by putting all appropriate values that are given. Where $n_{air}(\lambda, T, p)$ given by,

$$n_{air}(\lambda, T, p) = 1 + \frac{n_{air}(\lambda, 15, p_0) - 1}{1 + 3.4785 \cdot 10^{-3} \cdot (T - 15)} \cdot \frac{p}{p_0} \tag{26}$$

Again In equation (26), $n_{air}(\lambda, 15, p_0)$ is given by,

$$n_{air}(\lambda, 15, p_0) = 1 + 10^{-8} \cdot (6432.8 + \frac{2949810 \cdot \lambda^2}{146 \cdot \lambda^2 - 1} + \frac{25540 \cdot \lambda^2}{(41 \cdot \lambda^2 - 1))} \tag{27}$$

Here p_0 is 101330 Pa and p is variable. For equation (25), we need

$$\frac{dn_{air}(\lambda, T, p)}{dT} = -0.00367 \cdot \frac{n_{air}(\lambda, T, p) - 1}{1 + 0.00367 \cdot T} \tag{28}$$

With the help of the all coefficients and these equations we can plot between n vs T, $\frac{dn_{rel}(\lambda,T)}{dT}$ vs T and $\frac{dn_{abs}(\lambda,T)}{dT}$ vs T and see how refractive index is changed with temperature and pressure effects.
From the equation (24) we can write n_{rel} as $n_{abs}(\lambda, T) = n_{rel}(\lambda, T) \cdot n_{air}(\lambda, T, P)$, So we can write,

$$n_{rel}(\lambda, T) \cdot n_{air}(\lambda, T, P) = n_{rel}(\lambda, T_0) \cdot n_{air}(\lambda, T_0, P_0) + \Delta n_{abs}(\lambda, T) \tag{29}$$

In equation (29), P_0 is one atmosphere pressure which is given in catalog. Further simplification of equation gives,

$$n_{rel}(\lambda, T) = (n_{rel}(\lambda, T_0) + \frac{\Delta n_{abs}(\lambda, T)}{n_{air}(\lambda, T_0, P_0)}) \cdot \frac{n_{air}(\lambda, T_0, P_0)}{n_{air}(\lambda, T, P)} \tag{30}$$

From equation (30), we can get refractive index of glass type at any temperature and pressure with different wavelengths.

10 Data extraction through python

Now we have all catalogs with details of every glass type. Once we get the coefficients of glass type for dispersion formula, we can get refractive index with different wavelengths. Plot between refractive index and wavelength gives the proper idea of its trend. Similarly, Thermal analysis is done with python commands with loops and logic process. Once we get all thermal coefficients and reference temperature, we can obtain the change of rate of derivative of absolute refractive index with temperature, the change of rate of derivative of relative refractive index with temperature and trend between refractive index and temperature. These all plots are drawn through python. The code is given in the appendix. In the next section there are some examples of glass type with its details of thermal and dispersion trend.

10.1 Glass type with details from python

Here are total four glass types with its information get from different catalogs.
(1) N-BK7 from SCHOTT catalog
(2) S-LAH99 from OHARA catalog
(3) K-SFLD14 from SUMITA catalog
(4) J-PSKH1 from HIKARI catalog

10.1.1 N-BK7 from SCHOTT catalog

First of all, we will see how data from catalogs looks like, Here is data from schott catalog for N-BK7,

```
NM N-BK7 2 517642.251 1.5168 64.17 0 1
GC step 0.5 available
ED 7.100000 8.300000 2.510000 -0.000900 0
CD 1.039612120E+00 6.000698670E-03 2.317923440E-01 2.001791440E-02
1.010469450E+00 1.035606530E+02 0.000000000E+00 0.000000000E+00
TD 1.860000E-06 1.310000E-08 -1.370000E-11 4.340000E-07 6.270000E-10
1.700000E-01 2.000000E+01
OD 1.0000 1.0000 0.0000 1.0000 2.3000 2.3000
LD 3.00000E-01 2.50000E+00
IT 3.00000E-01 5.00000E-02 2.50000E+01
IT 3.10000E-01 2.50000E-01 2.50000E+01
.....................................
```

To understand what is TD, OD, LD, NM etc, you can refer section 6.5. From this data of glass, we can get information about glass through python. Here is plot between refractive index vs wavelength by using sellmeier 1 formula.

In the figure 14, the curve has negative slope and refractive index is decreasing with increasing of wavelength. Also formula is valid only for certain range of wavelengths. Here range is 300 nm to 2500 nm. Refractive index of N-BK7

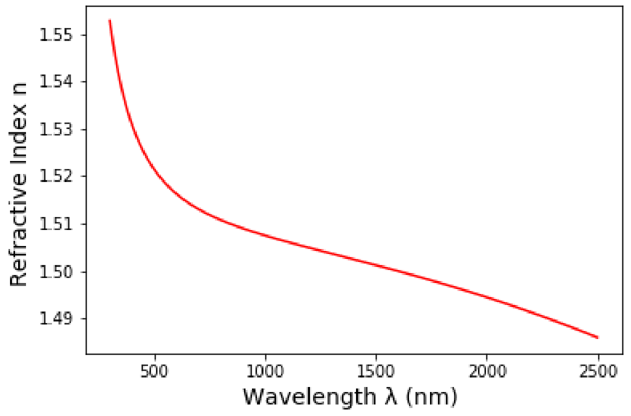

Figure 14: Refractive index vs wavelength for N-BK7 glass

at 1060 nm wavelength is 1.5066875568966998 and we can get every value by different inputs. From the equation of abbe number in section 5.1, the value for N-BK7 is 64.13319920721142. For thermal analysis, first I had found the change of rate of derivative of absolute refractive index with temperature. In the figure 15, there two curve, one for 560 nm and another for 1060 nm. The rate for 560 nm is greater than 1060 nm wavelength. Here the temperature range is -100 to 140 celcius.

Now refractive index at any given temperature and pressure is calculated from equation (30). In catalog, N-BK7 has reference temperature 20 celcius and I found the refractive index at temperature 40 celcius at pressure 103250 Pa. It gave the answer at wavelength 1060 nm around 1.5067275810747285. And at reference temperature with one atmospheric pressure is 1.5066875568966998. But if we keep temperature constant and decrease the pressure at 90000 Pa of surrounding then value is 1.5067772329259013. As we can see from data for refractive index at constant temperature but different pressure gives the different values. Reason for this trend is that pressure applies on lens surface inward while temperature of glass applies the force outward because of thermal expansion. So if we keep the temperature constant and decrease the pressure then temperature effect will dominate and refractive would be greater than the high pressure condition. Now the plot between refractive index for particular wavelength with temperature is in figure 16. For N-BK7, the refractive is increasing with increasing of temperature. Now fourth plot is between change of rate of derivative of relative refractive index with temperature at pressure 103250 Pa. This gives the rate at which relative refractive index changes with temperature. In the figure 17, the rate is decreasing till 0 celcius and increasing after that. So these are all analysis of N-BK7 from given catalog.

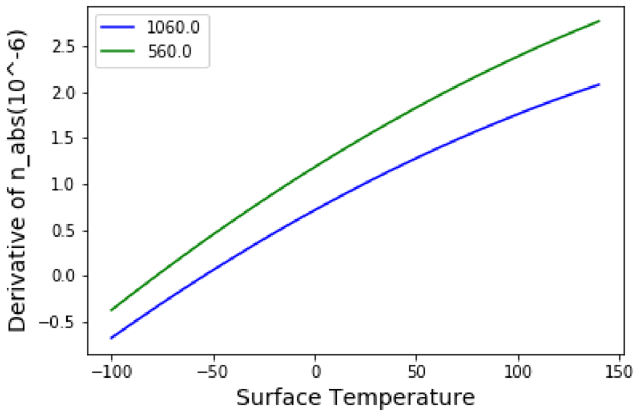

Figure 15: change of rate of derivative of absolute refractive index with temperature at 560 and 1060 nm

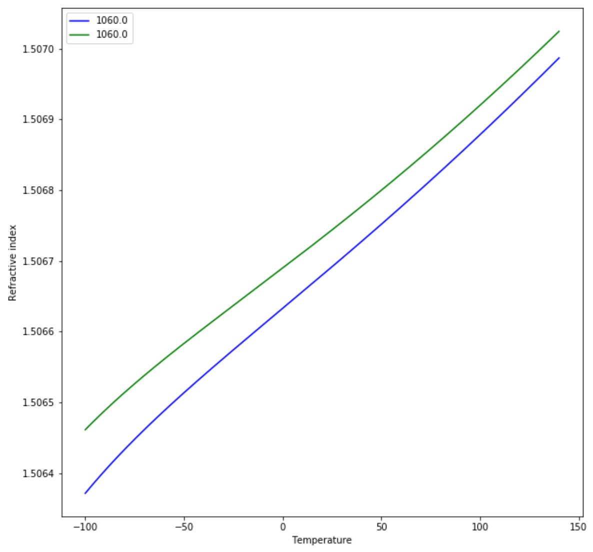

Figure 16: Green color for pressure at P=90000 Pa and Blue color for pressure P = 103250 Pa

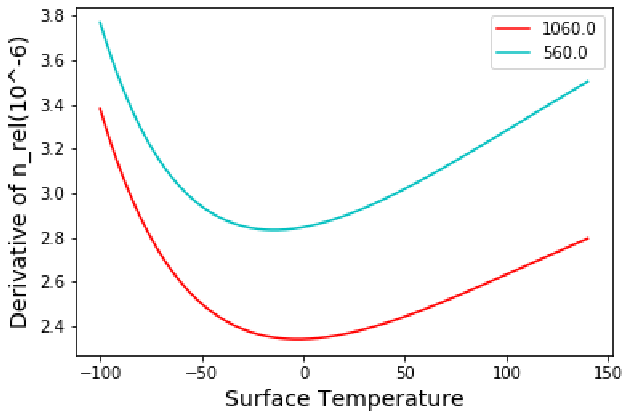

Figure 17: rate is changed drastically at temperature around 0 celcius

10.1.2 S-LAH99 from OHARA catalog

Data from catalog for S-LAH99 is,

NM S-LAH99 2 1 2.001000 29.139473 0 0 -1
GC
ED 7.500000000E+000 0.000000000E+000 5.020000000E+000
5.400000000E-003 0 0
CD 2.391406620E+000 1.314675000E-002 4.392192280E-001 5.532260420E-002
2.383584670E+000 1.612599000E+002 0.000000000E+000 0.000000000E+000
0.000000000E+000 0.000000000E+000
TD -5.060000000E-007 9.590000000E-009 -2.100000000E-011
1.090000000E-006 1.320000000E-009 2.670000000E-001 2.500000000E+001
OD 2.10000 1.00000 -1.00000 2.00000 -1.00000 1.00000
LD 4.04656000E-001 2.32542000E+000
IT 2.80000E-001 0.00000E+000 1.00000E+001
IT 2.90000E-001 0.00000E+000 1.00000E+001
....

From this data of glass, we can get information about glass through python. Here is plot between refractive index vs wavelength by using sellmeier 1 formula.

In the figure 14, the curve has negative slope and refractive index is decreasing with increasing of wavelength. Also formula is valid only for certain range of wavelengths. The range is 404 nm to 2325 nm. Refractive index of S-LAH99 at 1060 nm wavelength is 1.9659499406949175 and we can get every value by different inputs. From the equation of abbe number in section 5.1, the value for S-LAH99 is 29.11920404841508. For thermal analysis, the change of rate of

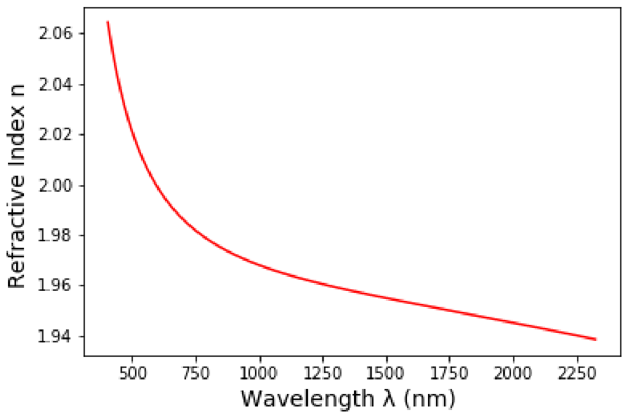

Figure 18: Refractive index vs wavelength for S-LAH99 with sellmeier 1 formula

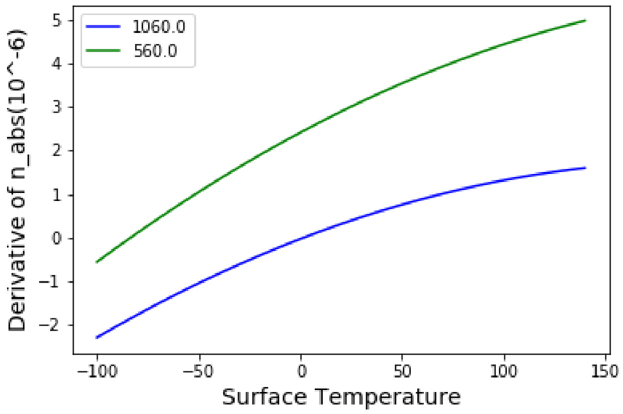

Figure 19: change of rate of derivative of absolute refractive index with temperature at 560 and 1060 nm

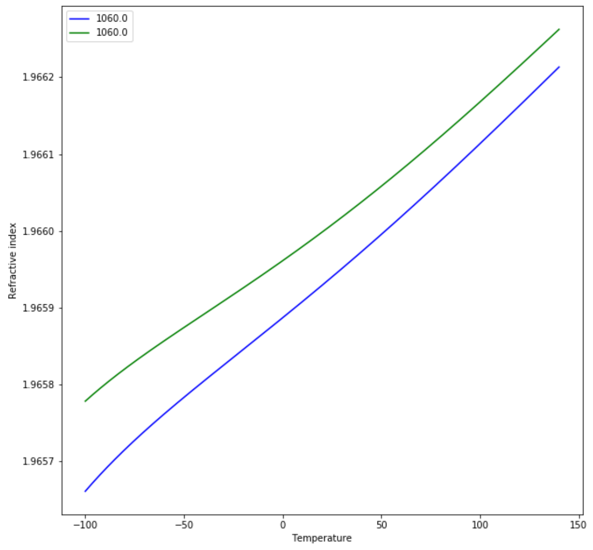

Figure 20: Green color for pressure at P=90000 Pa and Blue color for pressure P = 103250 Pa

derivative of absolute refractive index with temperature is given in the figure 19. There two curve, one for 560 nm and another for 1060 nm. The rate for 560 nm is greater than 1060 nm wavelength. Here the temperature range is -100 to 140 celcius. This is same as for N-BK7.

Now refractive index at any given temperature and pressure is calculated from equation (30). In catalog, S-LAH99 has reference temperature 25 celcius and I found the refractive index at temperature 40 celcius at pressure 103250 Pa. It gave the answer at wavelength 1060 nm around 1.965973023911095. And at reference temperature with one atmospheric pressure is 1.9659499406949175. But if we keep temperature constant and decrease the pressure at 90000 Pa of surrounding then value is 1.9660378094776565. As we can see from data for refractive index at constant temperature but different pressure gives the different values. Reason for this trend is same as for N-BK7. Now the plot between refractive index for particular wavelength with temperature is in figure 20. For S-LAH99, the refractive is increasing with increasing of temperature. Now fourth plot is between change of rate of derivative of relative refractive index with temperature at pressure 103250 Pa. This gives the rate at which relative refractive index changes with temperature. In the figure 21, the rate is decreasing till -50 celcius and increasing after that. So these are all analysis of

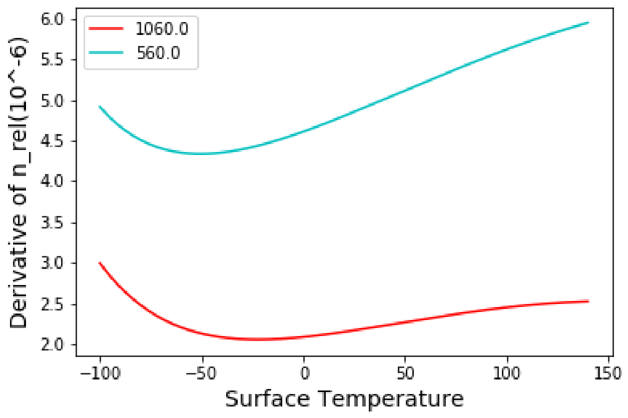

Figure 21: rate is changed drastically at temperature around -50 celcius

S-LAH99 from given catalog.

10.1.3 K-SFLD14 from SUMITA catalog

Data from catalog for K-SFLD14 is,

NM K-SFLD14 1 762265 1.76182 26.5 0
ED 8.5 10.2 3.15 0.0144 0
CD 2.9793916E+00 -1.1878549E-02 3.9643280E-02 1.4928665E-03
1.9358213E-05 1.2712921E-05 0.0000000E+00 0.0000000E+00
TD -4.13E-06 1.55E-08 -2.99E-10 9.65E-07 1.53E-09 0.290 2.00E+01
OD -1 1 -1 1 -1 -1
LD 0.4 1.55
IT 0.27 0.000 10
IT 0.28 0.000 10
....

From this data of glass, we can get information about glass through python. Here is plot between refractive index vs wavelength by using schott formula.

In the figure 22, the curve has negative slope and refractive index is decreasing with increasing of wavelength. Also formula is valid only for certain range of wavelengths. The range is 400 nm to 1550 nm. Refractive index of K-SFLD14 at 1060 nm wavelength is 1.7327813923080453 and we can get every value by different inputs. From the equation of abbe number in section 5.1, the value for K-SFLD14 is 26.48690066758293. For thermal analysis, the change of rate of derivative of absolute refractive index with temperature is given in the figure 23. There two curve, one for 560 nm and another for 1060 nm. The rate for 560 nm is greater than 1060 nm wavelength. Here the temperature range is -100

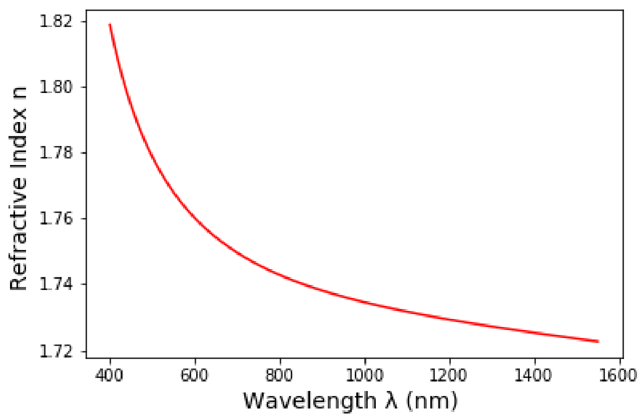

Figure 22: Refractive index vs wavelength for K-SFLD14 with schott formula

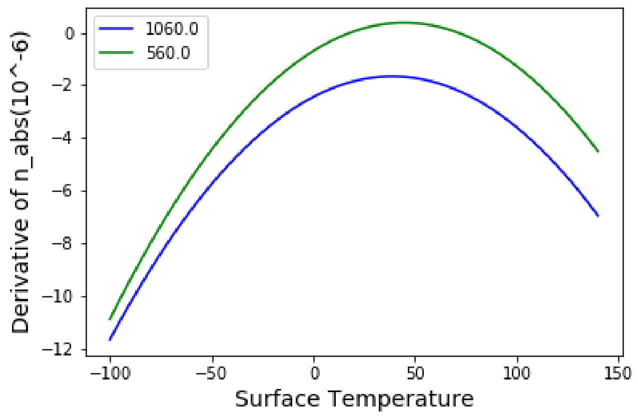

Figure 23: change of rate of derivative of absolute refractive index with temperature at 560 and 1060 nm

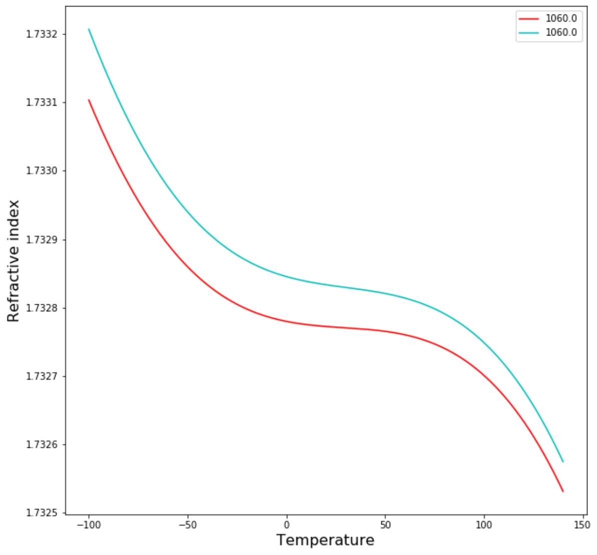

Figure 24: light blue color for pressure at P=90000 Pa and red color for pressure P = 103250 Pa

to 140 celcius. This is same as for N-BK7 and K-LAH99. Here there is slope change from positive to negative in this glass type.

Now refractive index at any given temperature and pressure is calculated from equation (30). In catalog, K-SFLD14 has reference temperature 20 celcius and I found the refractive index at temperature 40 celcius at pressure 103250 Pa. It gave the answer at wavelength 1060 nm around 1.7327685035899263. And at reference temperature with one atmospheric pressure is 1.7327813923080453. But if we keep temperature constant and decrease the pressure at 90000 Pa of surrounding then value is 1.7328256042662116. As we can see from data for refractive index at constant temperature but different pressure gives the different values. Reason for this trend is same as for N-BK7 and K-LAH99. Now the plot between refractive index for particular wavelength with temperature is in figure 24. For K-SFLD14, the refractive is decreasing with increasing of temperature. Now fourth plot is between change of rate of derivative of relative refractive index with temperature at pressure 103250 Pa. This gives the rate at which relative refractive index changes with temperature. In the figure 25, the rate is decreasing till 30 to 40 celcius and increasing after that. Here the rate is in negative and positive portion. That means that at some point refractive index does not change very much at that temperature. So these are all analysis

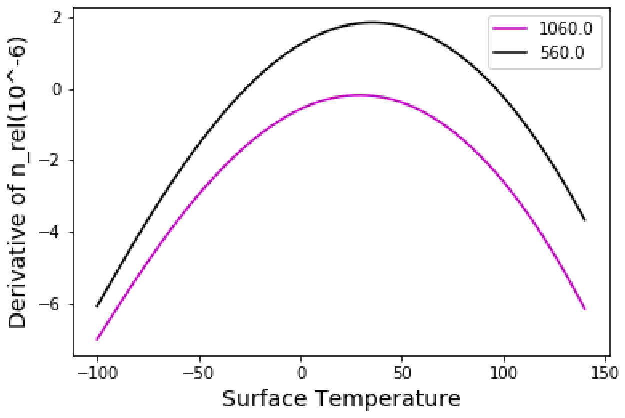

Figure 25: rate is changed drastically at temperature around 30 to 40 celcius

of K-SFLD14 from given catalog.

10.1.4 J-PSKH1 from HIKARI catalog

Data from catalog for J-PSKH1 is,

NM J-PSKH1 13 1 1.593190 67.900555 0 0 1
GC
ED 1.140000000E+001 0.000000000E+000 4.100000000E+000
1.350000000E-002 0 0
CD 2.502080830E+000 -6.721439070E-003 -5.343137510E-005
1.282644000E-002 1.562053880E-004 1.215935490E-006 9.595508690E-008
0.000000000E+000 0.000000000E+000 0.000000000E+000
TD 0.000000000E+000 0.000000000E+000 0.000000000E+000
0.000000000E+000 0.000000000E+000 0.000000000E+000 2.300000000E+001
OD -1.00000 -1.00000 -1.00000 -1.00000 -1.00000 3.00000
LD 3.88865000E-001 2.05809000E+000
IT 2.80000E-001 0.00000E+000 1.00000E+001
IT 2.90000E-001 0.00000E+000 1.00000E+001
....

From this data of glass, we can get information about glass through python. Here is plot between refractive index vs wavelength by using extended 3 formula.

In the figure 26, the curve has negative slope and refractive index is decreasing with increasing of wavelength. Also formula is valid only for certain range of wavelengths. The range is 388 nm to 2058 nm. Refractive index of J-PSKH1 at 1060 nm wavelength is 1.5830354695353313 and we can get every value by

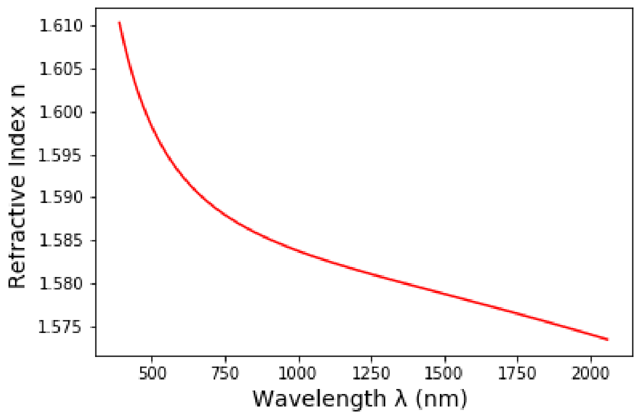

Figure 26: Refractive index vs wavelength for J-PSKH1 with extended 3 formula

different inputs. From the equation of abbe number in section 5.1, the value for J-PSKH1 is 67.86486866978349. For thermal analysis, the change of rate of derivative of absolute refractive index with temperature is given in the figure 27. There two curve, one for 560 nm and another for 1060 nm. The rate for 560 nm is same as 1060 nm wavelength. Here the temperature range is -100 to 140 celcius. The plot is constant with value 0. That means this glass type has no change in absolute refractive index

Now refractive index at any given temperature and pressure is calculated from equation (30). In catalog, J-PSKH1 has reference temperature 23 celcius and I found the refractive index at temperature 40 celcius at pressure 103250 Pa. It gave the answer at wavelength 1060 nm around 1.5830508417073328. And at reference temperature with one atmospheric pressure is 1.5830354695353313. But if we keep temperature constant and decrease the pressure at 90000 Pa of surrounding then value is 1.5831030086722035. As we can see from data for refractive index at constant temperature but different pressure gives the different values. Reason for this trend is same as for N-BK7 and K-LAH99. Now the plot between refractive index for particular wavelength with temperature is in figure 28. For J-PSKH1, the refractive is increasing with increasing of temperature. Now fourth plot is between change of rate of derivative of relative refractive index with temperature at pressure 103250 Pa. This gives the rate at which relative refractive index changes with temperature. In the figure 29, the rate is decreasing continuously with temperature. Here the rate is in only positive portion. So these are all analysis of J-PSKH1 from given catalog.

So These four glass type N-BK7, J-PSKH1,K-SFLD14, and S-LAH99 have different behaviour with temperature, their rate of change of refractive index, and change of refractive index with wavelengths. These all data got through

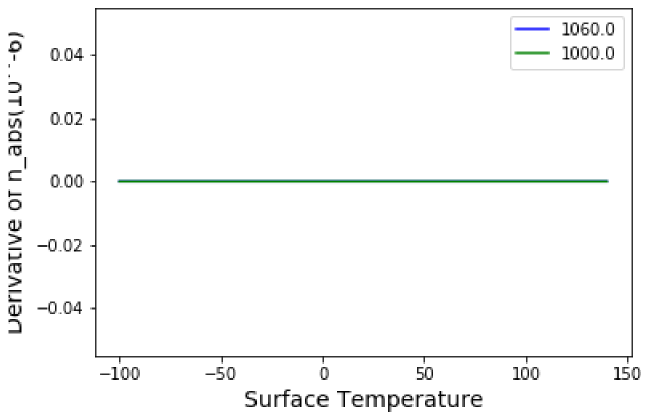

Figure 27: change of rate of derivative of absolute refractive index with temperature at 560 and 1060 nm

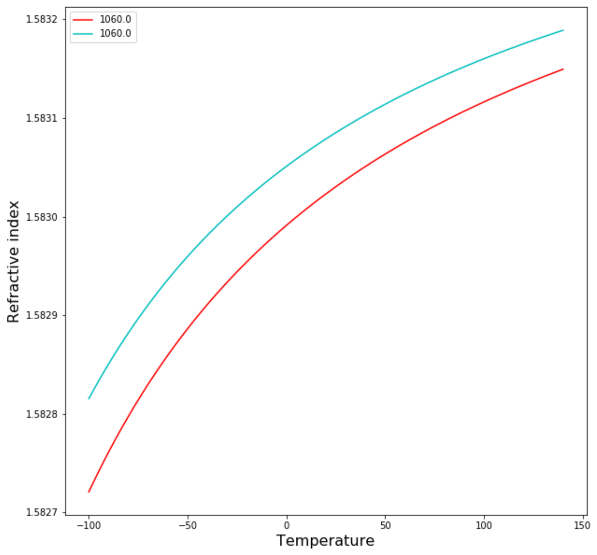

Figure 28: light blue color for pressure at P=90000 Pa and red color for pressure P = 103250 Pa

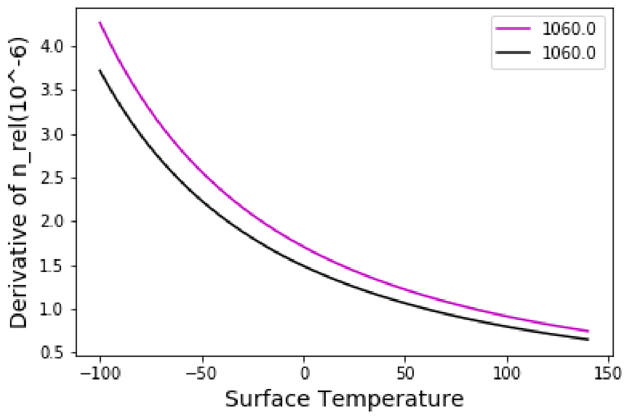

Figure 29: rate is decreasing with temperature

python programming. Now we need to verify our data with one reliable resources.

10.2 Verification of refractive index and thermal analysis data

From aforementioned data, I had plotted refractive index vs wavelength. After I got the reference data from one site which is mentioned in the references, I compared with my data with same glass type. The data verification were only for particular wavelength.For verification, I used the python programming to extract web/reference data directly. I include the code in the appendix. The wavelength is vary from glass types. Here is table which gives the verification of our data, From table 1, it is shown that the error is very low. That means data is valid. In the figure 30, the plot is between difference of data with reference and glass name. This is verification plot which has very small error.
Now come to data verification of thermal variations. From reference data, the verification is done through plotting of difference of rate of change of derivative of absolute refractive index vs temperature, difference of rate of change of derivative of relative refractive index vs temperature, and refractive index vs temperature at particular wavelength. I included only first two parts because there are any resources for refractive index vs temperature plot.

I included total five glasses for verification. (1) N-BK7, (2) F2 (3) N-LAF2 (4) N-PK51 (5) SF57 From the figures 31, 32, 33, 34, 35, 36, 37, 38, 39, and 40 the error or difference between given and calculated data is very small around 0.000001 which could be negligible. So these are all optical analysis of the different glass types in the given catalog.

I included table data for N-BK7 and F2 in table 2,3,4 and 5,6,7 respectively.

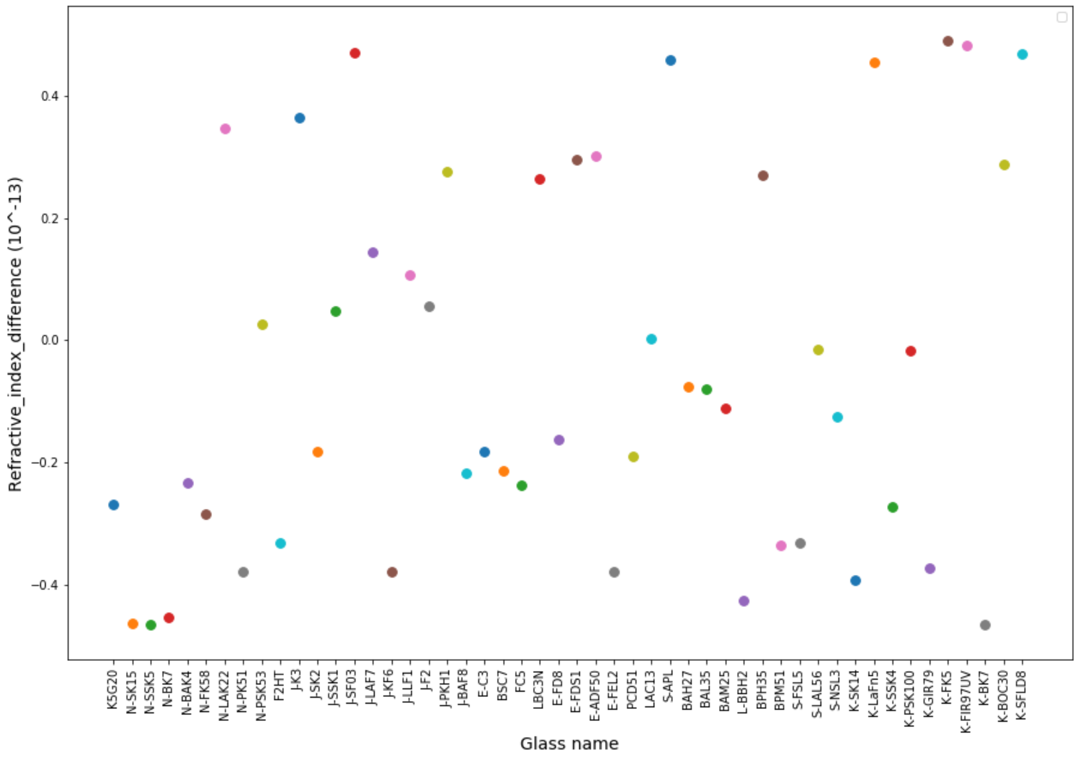

Figure 30: Verification of refractive index

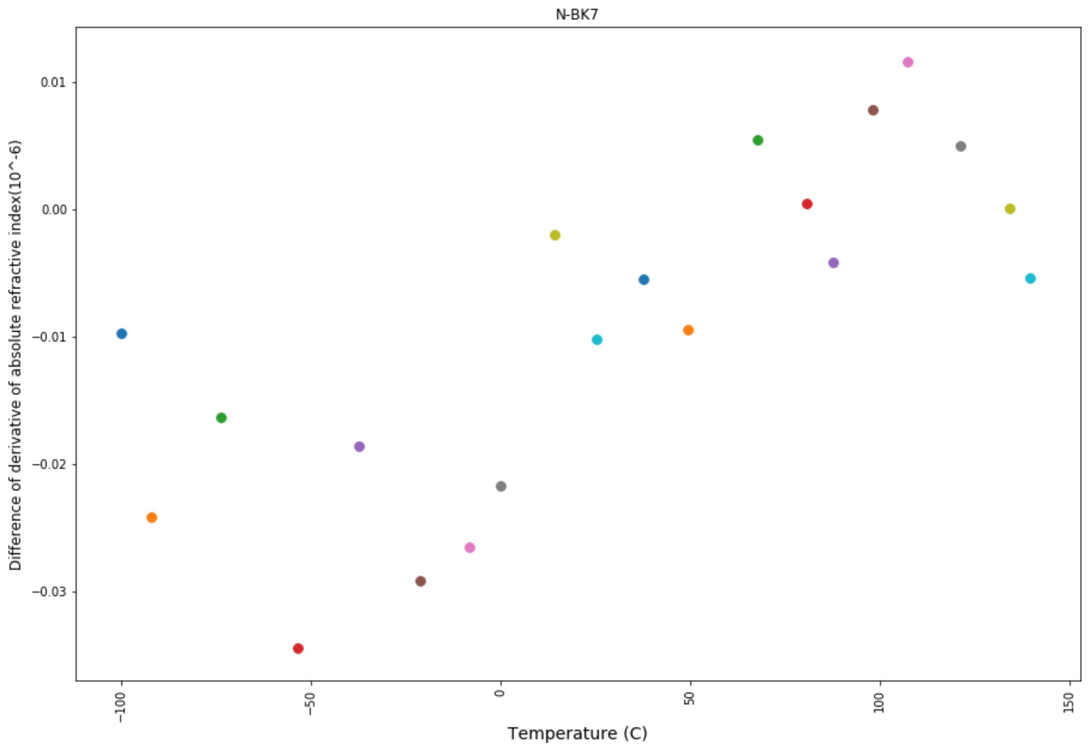

Figure 31: Verification of thermal data for N-BK7

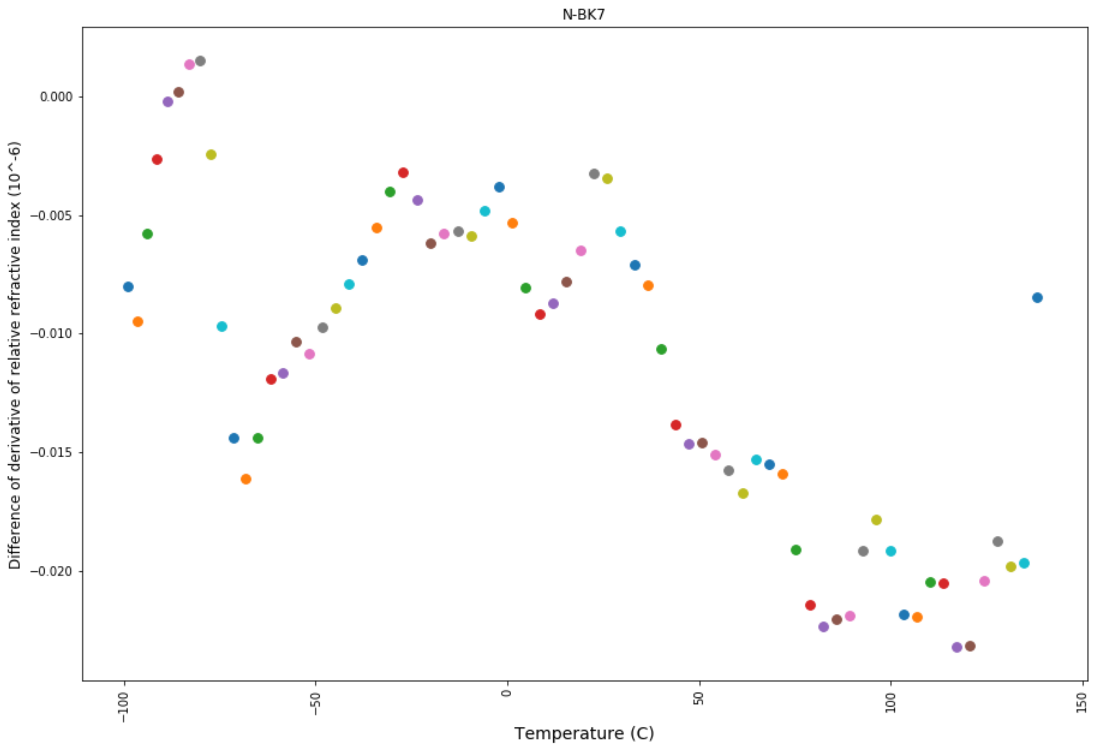

Figure 32: Verification of thermal data for N-BK7

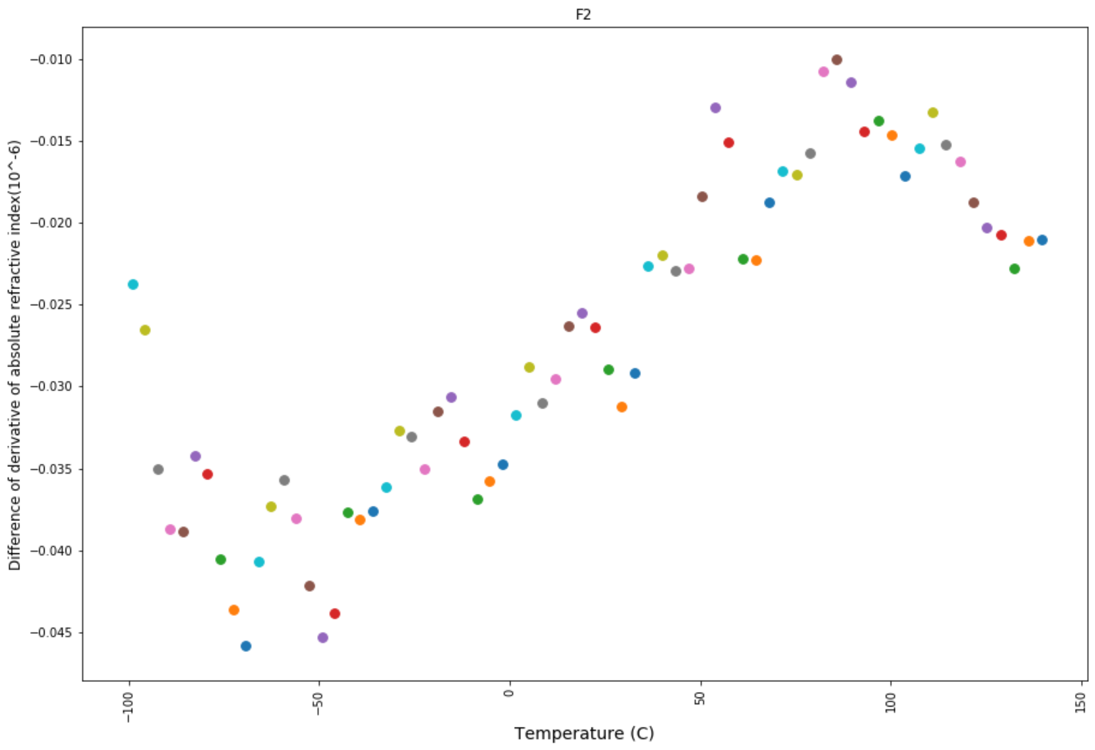

Figure 33: Verification of thermal data for F2

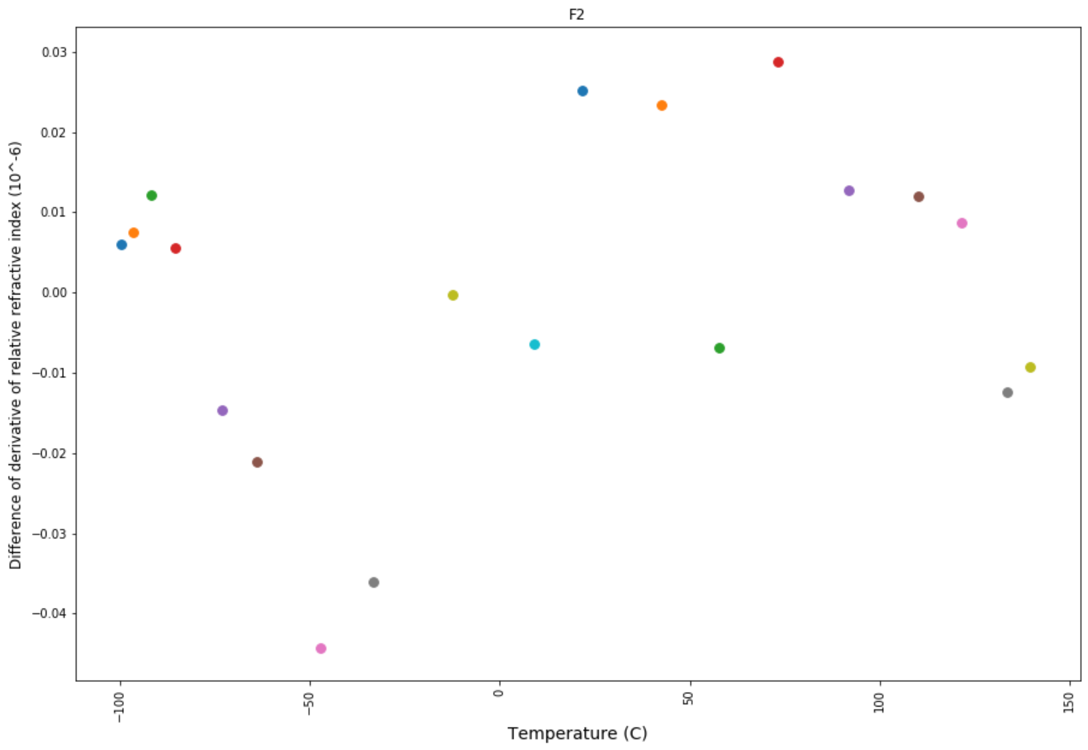

Figure 34: Verification of thermal data for F2

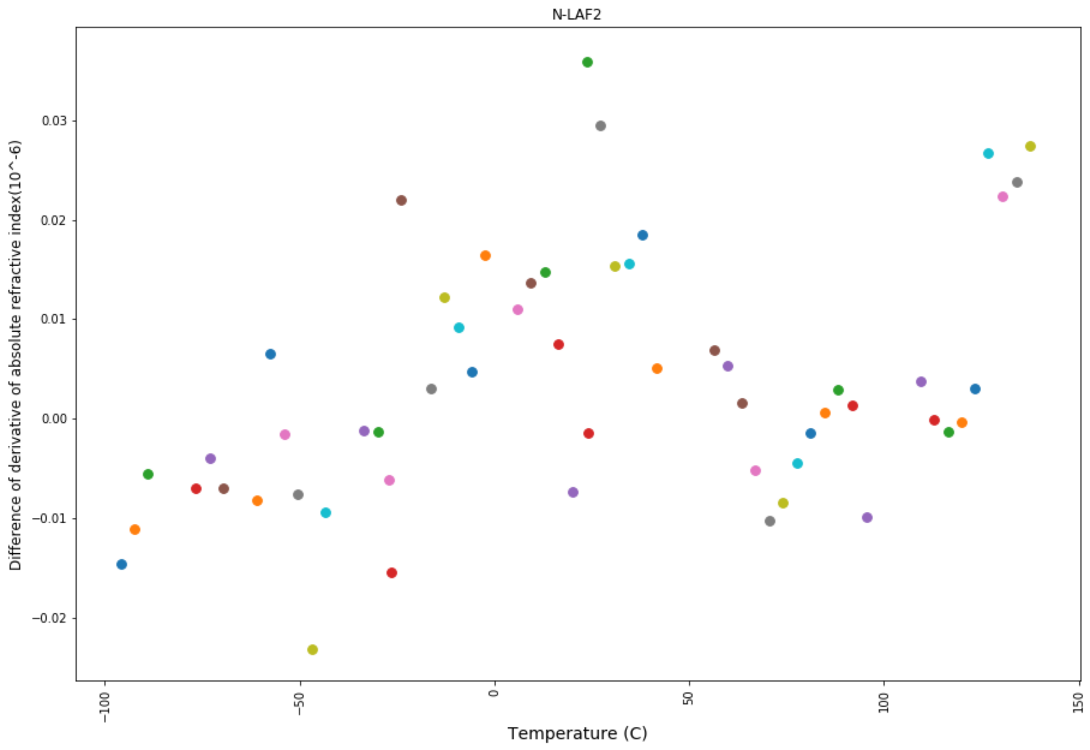

Figure 35: Verification of thermal data for N-LAF2

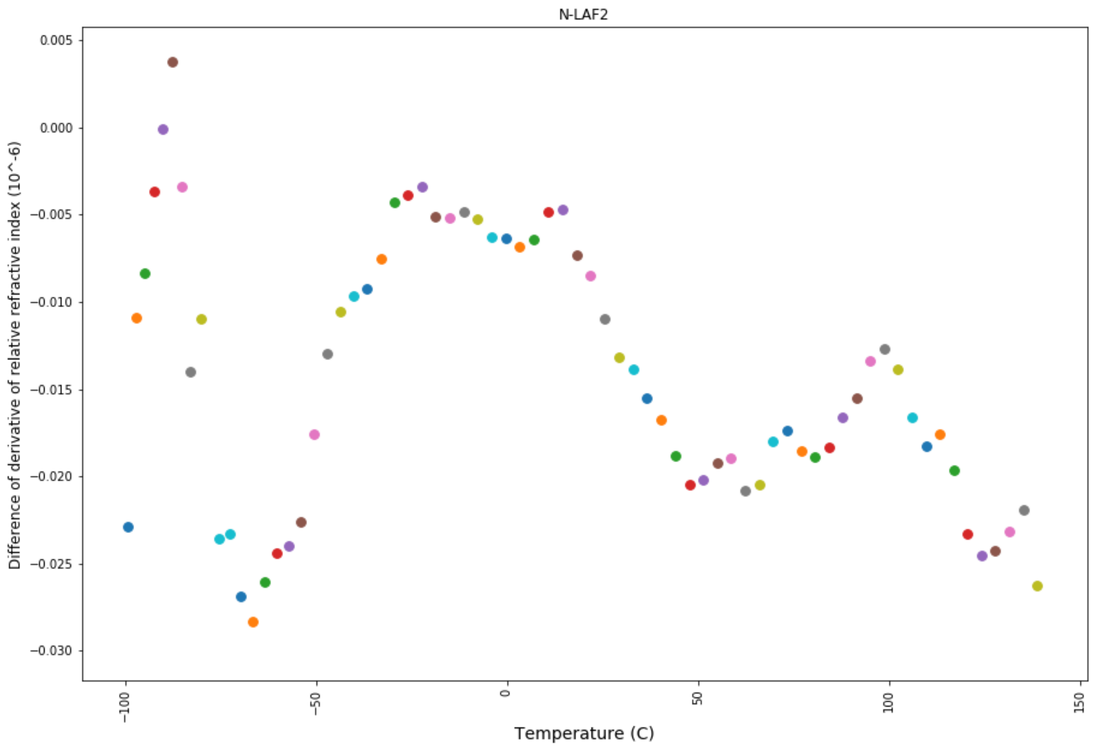

Figure 36: Verification of thermal data for N-LAF2

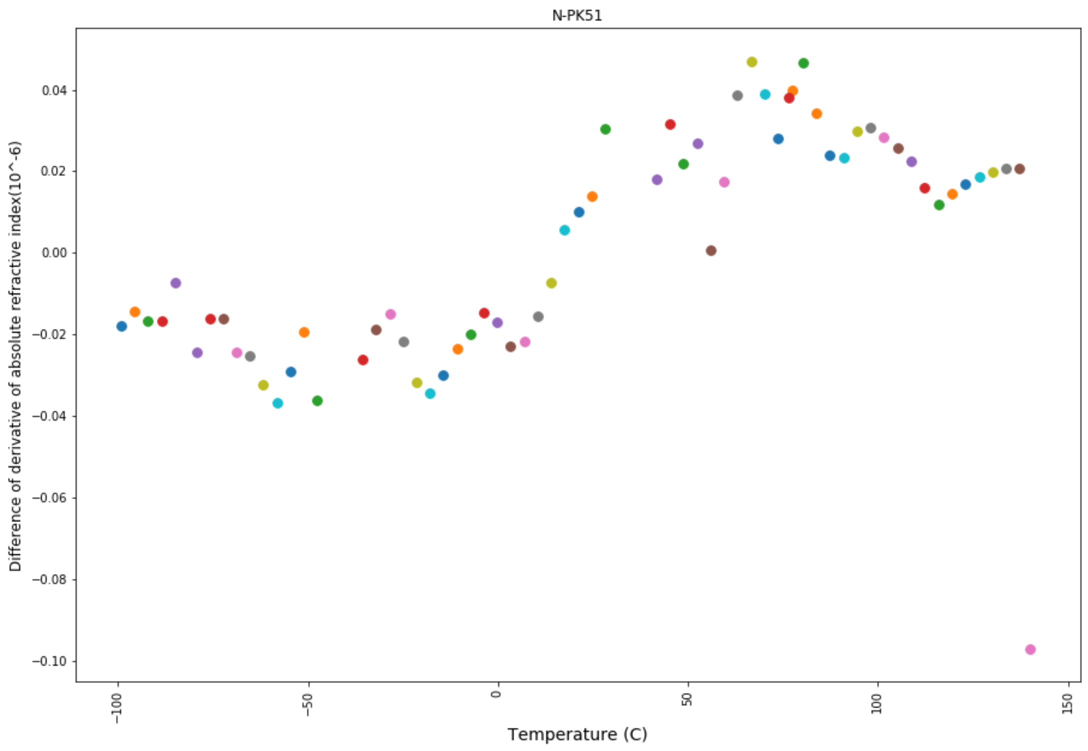

Figure 37: Verification of thermal data for N-PK51

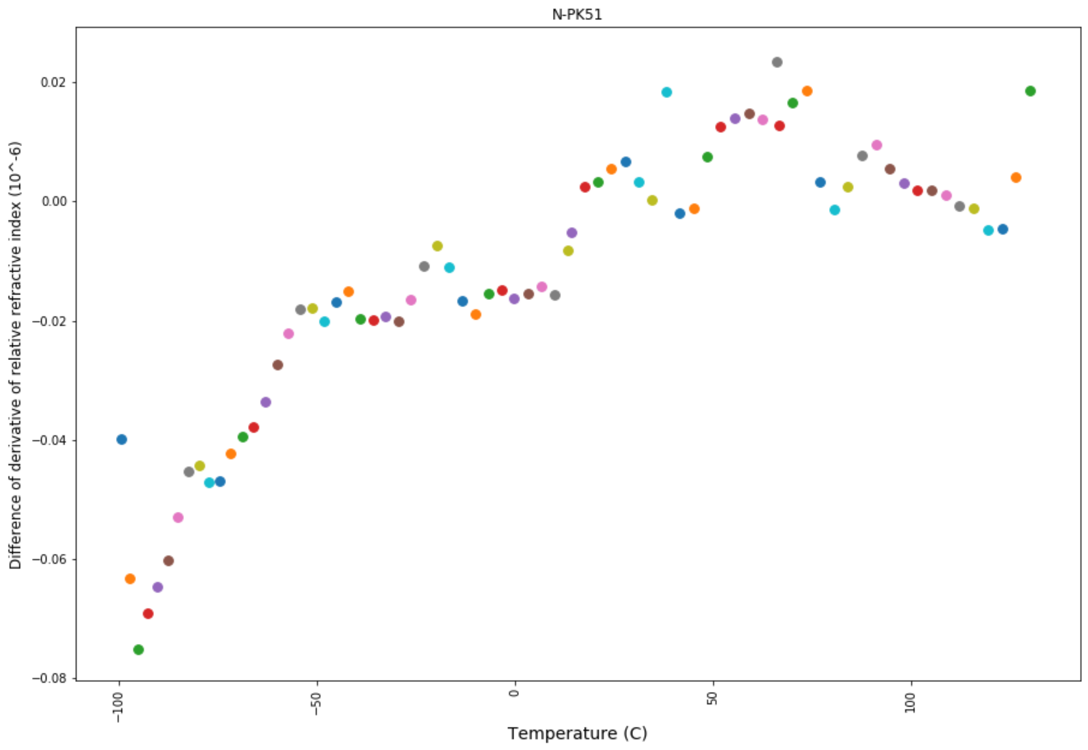

Figure 38: Verification of thermal data for N-PK51

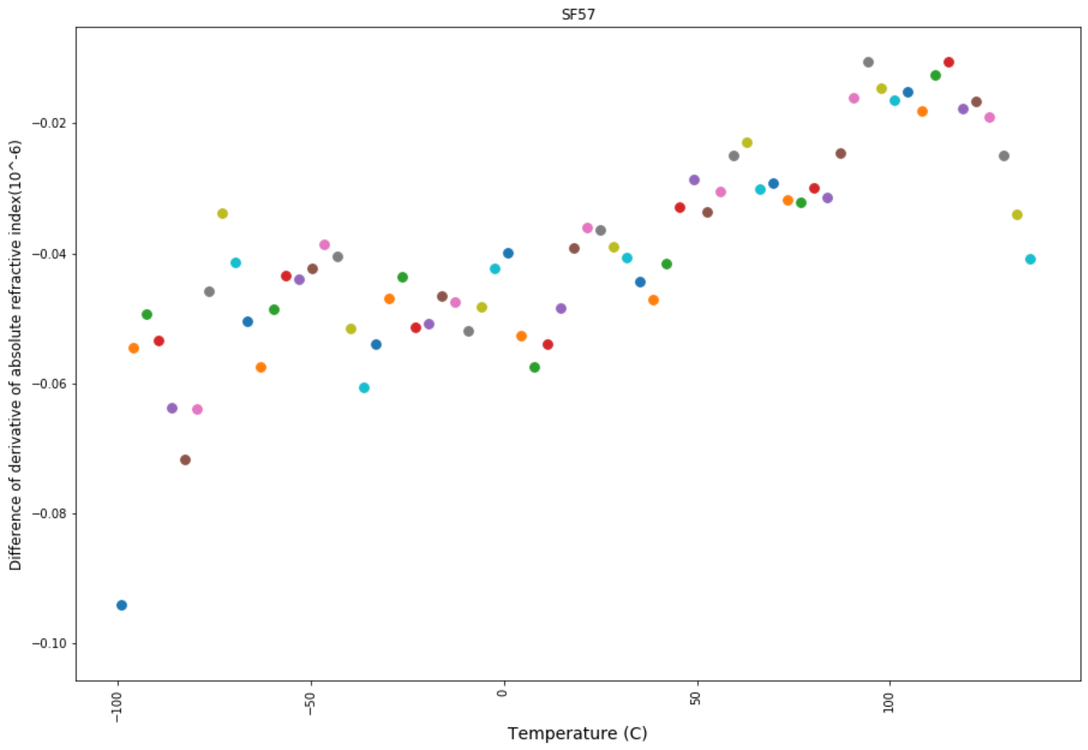

Figure 39: Verification of thermal data for SF57

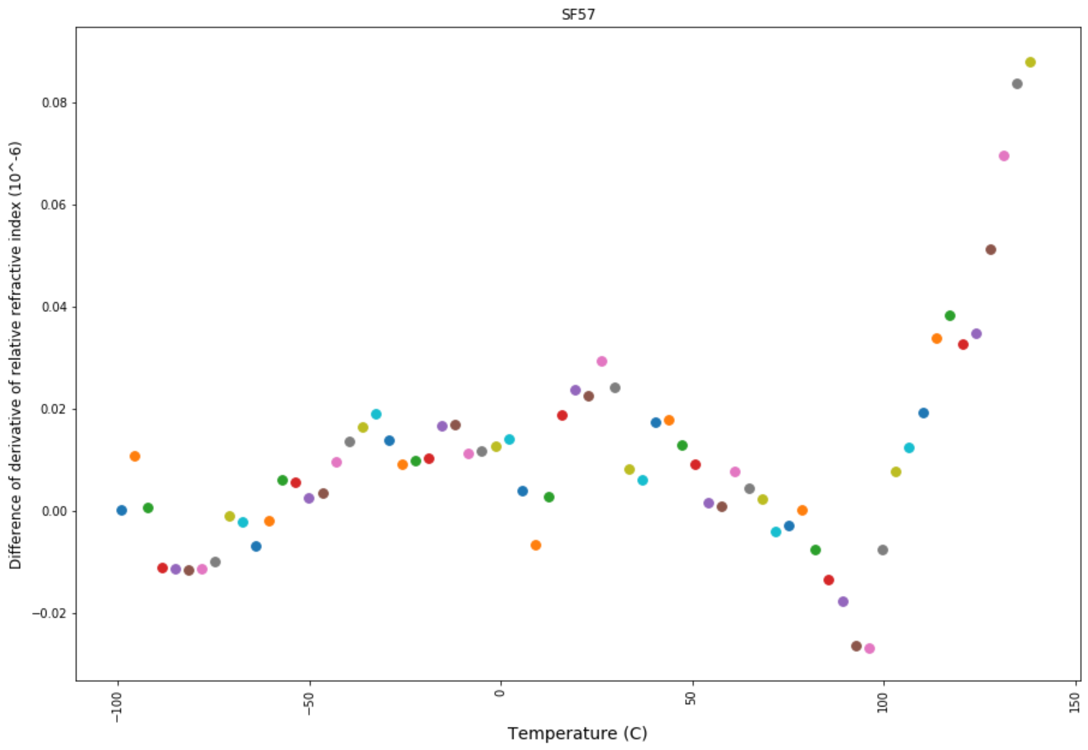

Figure 40: Verification of thermal data for SF57

Table 1: Verification of refractive index data

No.	Glass Name	Glass catalog	Wavelength ($\mu_{ab}m$	Refractive index in web	Refractive index calculated	Difference
1	K5G20	SCHOTT-K	1.435	1.508314166185	1.50831416618502	-2.68673971959287E-14
2	N-SK15	SCHOTT-SK	1.417	1.6055263722658	1.60552637226584	-4.64073224293315E-14
3	N-SSK5	SCHOTT-SSK	1.425	1.6385485784244	1.63854857842444	-4.66293670342565E-14
4	N-BK7	SCHOTT-BK	1.4	1.5024964846769	1.50249648467694	-4.52970994047063E-14
5	N-BAK4	SCHOTT-BaK	1.417	1.5524334945124	1.55243349451242	-2.33146835171282E-14
6	N-FK58	SCHOTT-FK	1.38	1.447776337143	1.44777633714302	-2.8421709430404E-14
7	N-LAK22	SCHOTT-LaK	1.405	1.632632492	1.632632492	3.46E-14
8	N-PK51	SCHOTT-PK	1.4	1.517637108	1.517637108	-3.80E-14
9	N-PSK53	SCHOTT-PSK	1.405	1.604256216	1.604256216	2.66E-15
10	F2HT	SCHOTT-F	1.41	1.596653591	1.596653591	-3.31E-14
11	J-K3	HIKARI-K	1.212	1.505919914	1.505919914	3.64E-14
12	J-SK2	HIKARI-SK	1.212	1.592865567	1.592865567	-1.82E-14
13	J-SSK1	HIKARI-SSK	1.212	1.601477046	1.601477046	4.88E-15
14	J-SF03	HIKARI-SF	1.223	1.806844021	1.806844021	4.71E-14
15	J-LAF7	HIKARI-LaF	1.223	1.723520287	1.723520287	1.44E-14
16	J-KF6	HIKARI-KF	1.212	1.503609009	1.503609009	-3.80E-14
17	J-LLF1	HIKARI-LLF	1.212	1.532465474	1.532465474	1.07E-14
18	J-F2	HIKARI-F	1.212	1.598971601	1.598971601	5.55E-15
19	J-PKH1	HIKARI-PK	1.212	1.507442564	1.507442564	2.75E-14
20	J-BAF8	HIKARI-BaF	1.212	1.606388671	1.606388671	-2.18E-14
21	E-C3	HOYA-C	0.6895	1.514523909	1.514523909	-1.82E-14
22	BSC7	HOYA-BSC	0.6895	1.513343373	1.513343373	-2.13E-14
23	FC5	HOYA-FC	0.6895	1.484504444	1.484504444	-2.38E-14
24	LBC3N	HOYA-LBC	0.6895	1.602243525	1.602243525	2.64E-14
25	E-FD8	HOYA-FD	0.6895	1.680091414	1.680091414	-1.62E-14
26	E-FDS1	HOYA-FDS	0.6895	1.905762088	1.905762088	2.95E-14
27	E-ADF50	HOYA-ADF	0.6895	1.647318779	1.647318779	3.02E-14
28	E-FEL2	HOYA-FEL	0.6895	1.535990459	1.535990459	-3.80E-14
29	PCD51	HOYA-PCD	0.6895	1.589715154	1.589715154	-1.91E-14
30	LAC13	HOYA-LaC	0.6895	1.688020147	1.688020147	2.22E-16
31	S-APL	OHARA-APL	0.6325	1.515705754	1.515705754	4.60E-14
32	BAH27	OHARA-BAH	0.6325	1.698063259	1.698063259	-7.55E-15
33	BAL35	OHARA-BAL	0.6325	1.587108397	1.587108397	-7.99E-15
34	BAM25	OHARA-BAM	0.6325	1.600315139	1.600315139	-1.11E-14
35	L-BBH2	OHARA-BBH	1.35	2.035367854	2.035367854	-4.26E-14
36	BPH35	OHARA-BPH	1.35	1.621582861	1.621582861	2.71E-14
37	BPM51	OHARA-BPM	1.35	1.592398805	1.592398805	-3.35E-14
38	S-FSL5	OHARA-FSL	1.34	1.475402158	1.475402158	-3.31E-14
39	S-LAL56	OHARA-LAL	1.365	1.658079449	1.658079449	-1.55E-15
40	S-NSL3	OHARA-NSL	1.36	1.504340537	1.504340537	-1.24E-14
41	K-SK14	SUMITA-SK	0.955	1.592759399	1.592759399	-3.93E-14
42	K-LaFn5	SUMITA-LaF	0.955	1.727775747	1.727775747	4.55E-14
43	K-SSK4	SUMITA-SSK	0.955	1.606249565	1.606249565	-2.73E-14
44	K-PSK100	SUMITA-PSK	0.955	1.581720436	1.581720436	-1.78E-15
45	K-GIR79	SUMITA-GIR	0.955	1.832273778	1.832273778	-3.73E-14
46	K-FK5	SUMITA-FK	0.955	1.479893339	1.479893339	4.91E-14
47	K-FIR97UV	SUMITA-FIR	0.955	1.421243807	1.421243807	4.82E-14
48	K-BK7	SUMITA-BK	0.955	1.507694575	1.507694575	-4.66E-14
49	K-BOC30	SUMITA-BOC	0.975	1.971646186	1.971646186	2.89E-14
50	K-SFLD8	SUMITA-SFLD	0.975	1.667971361	1.667971361	4.69E-14

11 Aspherical surface

Aspherical surface has wide application for correction in spherical aberration, coma aberration etc. To define the aspheric surfaces there is sag factor which varies with height from optical axis of the lens. The sag eqaution is given by,

$$Z(s) = \frac{Cs^2}{1 + (1 - (1+k)C^2s^2))^{1/2}} + A_4s^4 + A_6s^6 + ... \tag{31}$$

In equation (31), There are only even terms because of the symmetric lenses. If we want to design asymmetric lens then we have to add odd terms. Sag for any lens is measured from its line passing through vertex perpendicular to optical axis which is given in figure 41. Basically, sag defines the curvature of surface at particular point on lens. Here C is curvature of radius at vertex of the lens, k is depended on the reference shape of the lens like spherical, parabolic, hyperbolic

Table 2: Verification of absolute change for N-BK7

Given change in reference	Calculated by formula	Difference(10^-6)
-6.91E-01	-0.68148	-0.00973
-5.84E-01	-0.55973	-0.02421
-2.98E-01	-0.28162	-0.01636
-2.38E-02	0.010703	-0.03453
2.15E-01	0.233072	-0.01857
4.17E-01	0.446435	-0.02924
5.84E-01	0.610641	-0.02653
6.91E-01	0.713158	-0.02175
8.82E-01	0.884131	-0.00199
1.00E+00	1.011646	-0.0102
1.14E+00	1.150037	-0.00545
1.26E+00	1.273399	-0.00948
1.47E+00	1.46122	0.00551
1.59E+00	1.585696	0.000437
1.65E+00	1.650008	-0.00415
1.75E+00	1.745407	0.007863
1.84E+00	1.825202	0.011659
1.94E+00	1.93938	0.005054
2.04E+00	2.039976	0.000108
2.08E+00	2.081324	-0.00534

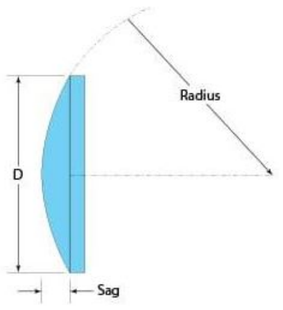

Figure 41: Sag for convex lens

Table 3: Verification of relative change for N-BK7

Given change in reference	Calculated by formula	Difference(10ˆ6)
3.27E+00	3.274056	-0.00802
3.20E+00	3.205571	-0.00948
3.13E+00	3.135894	-0.00578
3.07E+00	3.069411	-0.00264
3.00E+00	3.004968	-0.00022
2.94E+00	2.943821	0.000228
2.89E+00	2.88593	0.001375
2.83E+00	2.830337	0.001545
2.78E+00	2.778867	-0.00241
2.72E+00	2.732037	-0.00968
2.67E+00	2.686575	-0.01437
2.63E+00	2.64344	-0.0161
2.59E+00	2.602141	-0.01439
2.55E+00	2.564013	-0.01189
2.52E+00	2.529449	-0.01163
2.49E+00	2.497772	-0.01031
2.46E+00	2.46929	-0.01086
2.43E+00	2.443086	-0.00973
2.41E+00	2.419828	-0.0089
2.39E+00	2.39904	-0.00791
2.37E+00	2.380885	-0.00691
2.36E+00	2.364979	-0.00552
2.35E+00	2.351595	-0.00401
2.34E+00	2.340197	-0.00317
2.33E+00	2.330842	-0.00437
2.32E+00	2.32339	-0.00616
2.31E+00	2.317713	-0.00576
2.31E+00	2.313689	-0.00569
2.31E+00	2.311208	-0.00585
2.31E+00	2.310164	-0.00481
2.31E+00	2.310459	-0.00378
2.31E+00	2.312001	-0.00532
2.31E+00	2.314706	-0.00803
2.31E+00	2.318492	-0.00918
2.31E+00	2.323283	-0.00869
2.32E+00	2.32901	-0.00782
2.33E+00	2.335606	-0.0065
2.34E+00	2.3429	-0.00323
2.35E+00	2.35104	-0.00346
2.35E+00	2.359874	-0.00569

Table 4: Verification of relative change for N-BK7

Given change in reference	Calculated by formula	Difference(10^-6)
2.36E+00	2.369213	-0.00711
2.37E+00	2.379271	-0.00793
2.38E+00	2.389876	-0.01062
2.39E+00	2.400983	-0.01381
2.40E+00	2.412387	-0.01466
2.41E+00	2.424201	-0.01459
2.42E+00	2.436564	-0.01508
2.43E+00	2.449099	-0.01574
2.45E+00	2.461943	-0.01671
2.46E+00	2.475064	-0.01531
2.47E+00	2.488436	-0.01549
2.49E+00	2.502031	-0.01589
2.50E+00	2.515823	-0.01912
2.51E+00	2.529985	-0.02141
2.52E+00	2.544099	-0.02233
2.54E+00	2.55834	-0.02205
2.55E+00	2.572685	-0.02188
2.57E+00	2.587114	-0.01915
2.58E+00	2.601609	-0.01781
2.60E+00	2.616153	-0.01916
2.61E+00	2.630726	-0.02186
2.62E+00	2.645312	-0.02193
2.64E+00	2.65969	-0.02047
2.65E+00	2.674256	-0.02052
2.67E+00	2.68879	-0.02318
2.68E+00	2.703278	-0.02315
2.70E+00	2.717705	-0.02042
2.71E+00	2.731859	-0.01874
2.73E+00	2.746132	-0.01982
2.74E+00	2.760509	-0.01968
2.77E+00	2.774376	-0.00848

Table 5: Verification of absolute change for F2

Given change in reference	Calculated by formula	Difference(10^6)
2.50E+00	2.523539	-0.02103
2.48E+00	2.499842	-0.02109
2.45E+00	2.475096	-0.02273
2.43E+00	2.449303	-0.02069
2.40E+00	2.422461	-0.02024
2.38E+00	2.39457	-0.01875
2.35E+00	2.365631	-0.0162
2.32E+00	2.335644	-0.01524
2.29E+00	2.304608	-0.01324
2.26E+00	2.272524	-0.01546
2.22E+00	2.239865	-0.01711
2.19E+00	2.205699	-0.01462
2.16E+00	2.170484	-0.01371
2.12E+00	2.134221	-0.0144
2.09E+00	2.096909	-0.0114
2.05E+00	2.058549	-0.00999
2.01E+00	2.019703	-0.01073
1.96E+00	1.979838	-0.01573
1.92E+00	1.938954	-0.01708
1.88E+00	1.896445	-0.01679
1.83E+00	1.853508	-0.01872
1.79E+00	1.809552	-0.02227
1.74E+00	1.764577	-0.02216
1.70E+00	1.717918	-0.01509
1.66E+00	1.67089	-0.01293
1.61E+00	1.623537	-0.01836
1.55E+00	1.575193	-0.0228
1.50E+00	1.525137	-0.02289
1.45E+00	1.474062	-0.02196
1.40E+00	1.421968	-0.02265
1.34E+00	1.370387	-0.02914
1.29E+00	1.317063	-0.03124
1.23E+00	1.261956	-0.02891
1.18E+00	1.206638	-0.02638
1.12E+00	1.15033	-0.02549
1.07E+00	1.093031	-0.02626
1.01E+00	1.035595	-0.02952

Table 6: Verification of absolute change for F2

Given change in reference	Calculated by formula	Difference(10^-6)
9.45E-01	0.976331	-0.03096
8.87E-01	0.916076	-0.02877
8.24E-01	0.855726	-0.03176
7.61E-01	0.795323	-0.0347
6.97E-01	0.733064	-0.03578
6.34E-01	0.67078	-0.03684
5.73E-01	0.606612	-0.03337
5.10E-01	0.540517	-0.03062
4.44E-01	0.47539	-0.03147
3.75E-01	0.410322	-0.03503
3.09E-01	0.342335	-0.03302
2.41E-01	0.273387	-0.03269
1.69E-01	0.205546	-0.03611
9.82E-02	0.135752	-0.03757
2.69E-02	0.065023	-0.0381
-4.43E-02	-0.00664	-0.0377
-1.21E-01	-0.07705	-0.04382
-1.95E-01	-0.14945	-0.04532
-2.66E-01	-0.22387	-0.04216
-3.37E-01	-0.29923	-0.03806
-4.11E-01	-0.37552	-0.03567
-4.88E-01	-0.45042	-0.03731
-5.67E-01	-0.5262	-0.0407
-6.49E-01	-0.60286	-0.04586
-7.25E-01	-0.6816	-0.04366
-8.02E-01	-0.76125	-0.04055
-8.78E-01	-0.84302	-0.03531
-9.58E-01	-0.92325	-0.03425
-1.04E+00	-1.00311	-0.03886
-1.12E+00	-1.08508	-0.0387
-1.20E+00	-1.16793	-0.03502
-1.28E+00	-1.25296	-0.02653
-1.36E+00	-1.33758	-0.02373

Table 7: Verification of relative change for F2

Given change in reference	Calculated by formula	Difference(10ˆ6)
2.87E+00	2.865462	0.005967
2.81E+00	2.79969	0.007453
2.72E+00	2.709301	0.012127
2.61E+00	2.608725	0.005561
2.46E+00	2.478892	-0.01461
2.40E+00	2.421079	-0.02108
2.34E+00	2.380092	-0.04438
2.36E+00	2.393282	-0.03614
2.46E+00	2.464525	-0.00024
2.57E+00	2.577903	-0.00647
2.68E+00	2.653308	0.025264
2.81E+00	2.783769	0.023374
2.87E+00	2.878251	-0.00682
3.00E+00	2.971181	0.028819
3.09E+00	3.072942	0.012773
3.17E+00	3.159488	0.01194
3.21E+00	3.205558	0.008727
3.24E+00	3.248061	-0.01235
3.26E+00	3.266383	-0.00924

etc. For example, $k = 0$ it is sphere. Also s is height of the source from optical axis. And last part is constants which are $A_4, A_6, A_8, ...$ etc.
Disadvantage of the aspheircal surfaces are that they are very costly and hard to manufacture particular type of glass shape with good precision.

Example (1) For equation given below,

$$z = \frac{c_x x^2 + c_y y^2}{1 + ((1 - (1 + k_x)c_x^2 x^2 - (1 + k_y)c_y^2 y^2)^{1/2}} \tag{32}$$

In the equation (32), the surface is biconic which is graphically shown in the figure 42.

11.1 Radius of curvature at each point

In aspherical surfaces, the radius of curvature at each point would be different and it is founded by taking tangent curve at each point and some geometry. The final equation of the radius of curvature at each point is given by,

$$R = \frac{(1 + (z^{'}(s))^2)^{3/2}}{z^{''}(s)} \tag{33}$$

In the equation (33), the radius of curvature is founded by taking derivatives of sag factor.

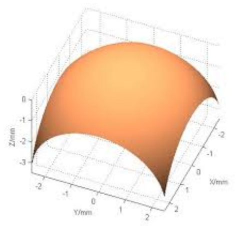

Figure 42: Biconic surface

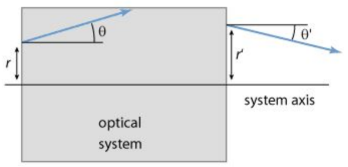

Figure 43: Rays tracing method

12 ABCD matrix for paraxial rays

This matrix is useful tool for describing the effects of optical elements in the system for monochromatic light. The matrix form for any lens with paraxial rays is given by equation,

$$\begin{bmatrix} r_1 \\ \theta_0 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} r_0 \\ \theta_0 \end{bmatrix}$$

Where r_0,θ_0 are initial position of the object and image would have coordinates r_1, θ_0 after passing through the optical elements. Where A,B,C and D are dependent on the lens types and material. In the figure 43, you can see the coordinate system for rays. Here are some standard examples, (1) Thin lens with focal length f,

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{-1}{f} & 1 \end{bmatrix}$$

(2)Translation matrix for the medium

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} 1 & d \\ 0 & 1 \end{bmatrix}$$

Where d id distance travelled by the ray. (3) Curved mirror with radius R,

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{-2}{R} & 1 \end{bmatrix}$$

(4)Refraction

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{n-n'}{n'R} & \frac{n'}{n} \end{bmatrix}$$

Where $n^{'}$ is refractive index of the medium and n is refractive index of outside the medium. If we have one system with convex lens then we can use this ABCD matrices for each and every element. For example, for convex lens, first ray will encounter with refraction at the first surface. After translation with some distance in lens medium it will again refract to the outside the lens. So we need to multiply each matrices to get final matrix for whole system. Order for multiplication should be maintained. For this example, The first matrix is refraction on coordinates, the second is translation on first and third is refraction on second matrix. But this matrices only use for the paraxial rays approximation.

To get precise image distance and its height, we need to consider oblique rays. So we can't use the ABCD matrix for the aspherical surfaces because it needs to modify equations. ABCD matrix is linear relation but when system becomes more complicated the matrix could not be represented as linear or matrix form. ABCD matrix is very useful for simple systems. To find precise answer we need to find optical path difference and height of image which also gives the idea of aberrations of system. I already mentioned the different types of the aberrations in the section (3).

13 Optical path difference and ray tracing method to find aberration

When paraxial rays(parallel to optical axis) or oblique rays or peripheral rays(at edge of the lens) are incident on the convex lens, not all rays travel the same distance to the image plane and it create the path difference between rays. In telescope, generally rays are incident on the lens with very small and equal angle. Because of the optical path difference there will be phase difference between rays in the wave-front. There are so many aberrations due to lens material, surface shape and surrounding conditions. There are first, second, third and higher order aberration in the lens. The aberration is already discussed in the section 3.0 .

From the aspheric surface, we can define any shape of convex lens with angular symmetric. From basic geometry of the convex lens we can get the final image of the plane parallel beam at infinite position. In the figure 44 the rays travel from initial position to the convex lens with refractive index $n = 1.5$ and transmit through air media. Here thickness is d. semi diameter is distance between vertex to maximum point or edge of the lens. Here two rays concentrate over different regions. This creates the coma and spherical aberration. Coma aberration occurs when the parallel beam incident with small angle. Optical axis is passing through middle of the lens. Aberration occurs because these two rays travel different path and path difference creates the phase difference. Now next part is to find optical path difference with simple geometry in the system and find through python programming.

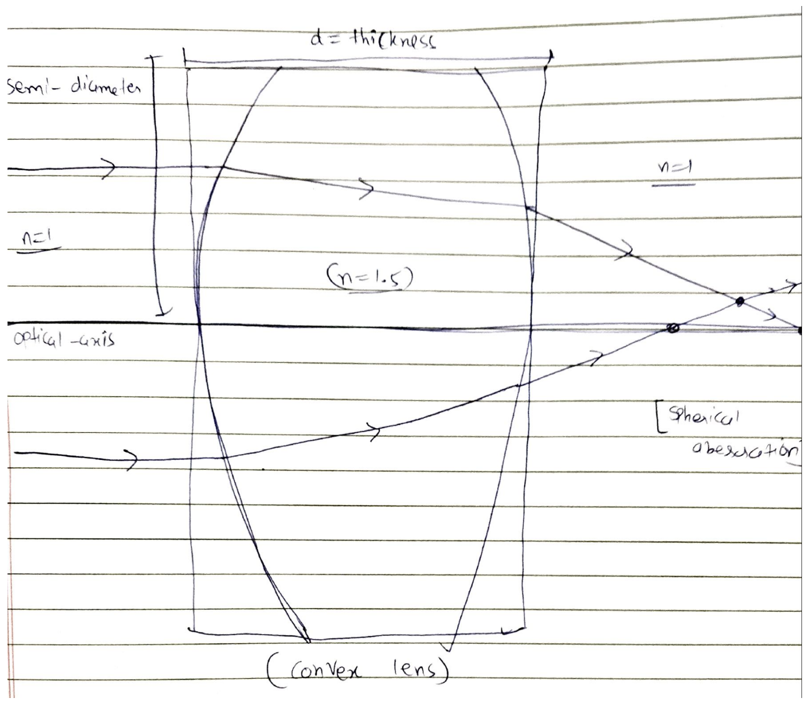

Figure 44: Convex lens with small portion

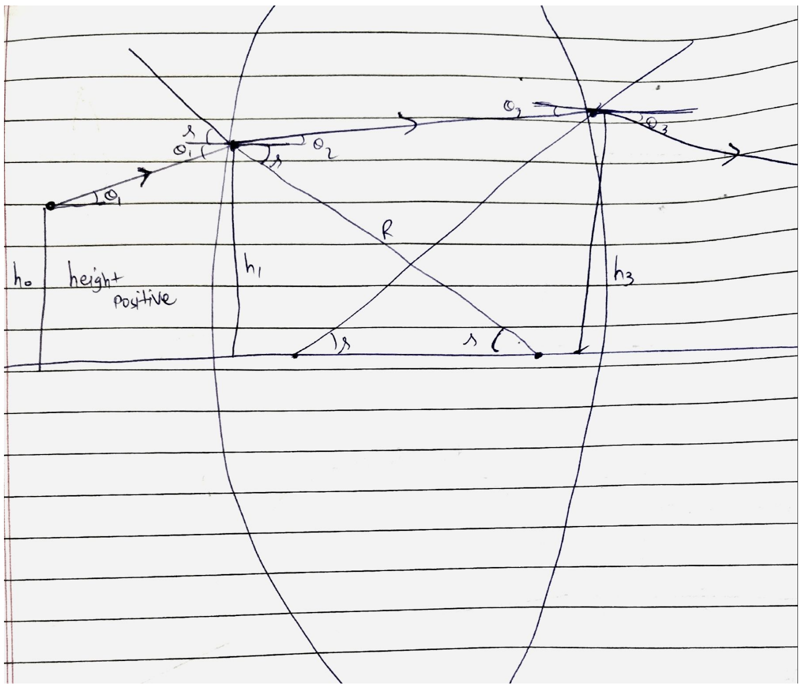

Figure 45: Lens with geometry

Geometry of ray is shown into figure 45. Here initial height is h_0 and incident on the lens with angle θ_1. The angle and height above the optical axis are taken as positive and below optical axis are taken as negative. In this example for instance I have taken both part as positive. Initial ray emits from height h_0 with positive angle θ_1. After bending of ray due to medium of lens, the resultant angle becomes θ_2 which could be negative or positive which depends on the snell's law for refraction at the surface of lens. Snell's law stated that $n_1 \cdot sin(\theta_1) = n_2 \cdot sin(\theta_2)$. I have taken the θ_2 as positive. Second thing is that due to angle θ_1 the height at surface would be different. Suppose that height is h_1. After travelling through medium, the final resultant angle becomes θ_3 and height becomes h_3. As aforementioned sag factor uses in to defining the surface of the lens, different rays travel different path in the air and glass media.

Also the medium could be any of the glass type which I included in the catalogs. For example if the media is N-BK7 and surrounding temperature is 30 C then refractive index will be 1.506678. So we can use the different glass types data with their thermal analysis and defining the surface structure, we can find the final height of the image and its aberration/optical path difference at different positions.

Geometry of the lens I included simple geometry for convex lens with middle part of it. From the figure 46, the ray starts from point O and refracts from front surface of the convex lens. Suppose ray makes the angle θ with horizontal line. From trigonometry of the figure, distance travelled by a ray between O and A would become, $D_1 = L/cos(\theta_1) + Z_1/cos(\theta_1)$ Where L and Z_1 are distance given in figure 47. Z_1 is sag at that point on surface. Here the surface is spherical so sag of the surface would become, $Z = \frac{h_1^2}{R+(R^2+h_1^2)^{1/2}}$. h is height of the point A on the surface. Both curved surface are assumed to be spherical therefore sag function is same for both surface.

After refraction at first surface, the ray will bend towards horizontal line. In the figure 46, AM_1 is the line passing through center of the curves surface. The angle made by this AM_1 is r and it is given by, $sin(r) = h_1/R$. R is radius of the curved surface. Snell's law applies for only angle made with perpendicular to the surface. Here AM_1 is perpendicular to the surface. So incident angle would become $i = \theta_1 + r$ and angle after refraction become $e = \theta_2 + r$. From snell's law $n_1 \cdot sin(i) = n_2 \cdot sin(e)$, We can find the angle e and subtraction with angle r gives the angle with horizontal line which is basically θ_2. n_1 is refractive index of the air and n_2 is refractive index of the medium which could be found by aforementioned methods to find refractive index for different wavelengths. To find the height at the surface, $h_1 = h_0 + tan(\theta_1)(L + Z_1)$. Here Z_1 is function of h_1 which makes the equation with function of h_1. After solving the equation, it gives the value of h_1.

After getting θ_2 and h_1, ray travels through medium to the point B in the

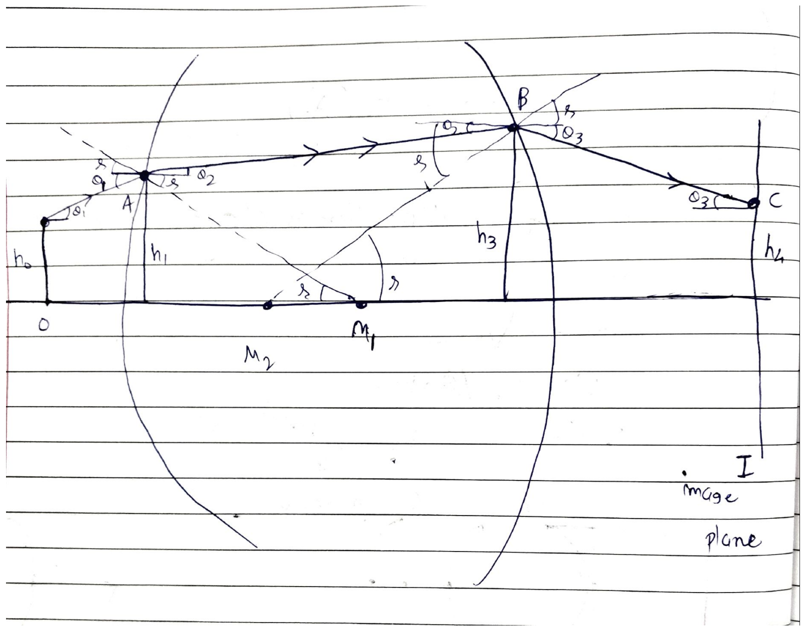

Figure 46: Convex lens

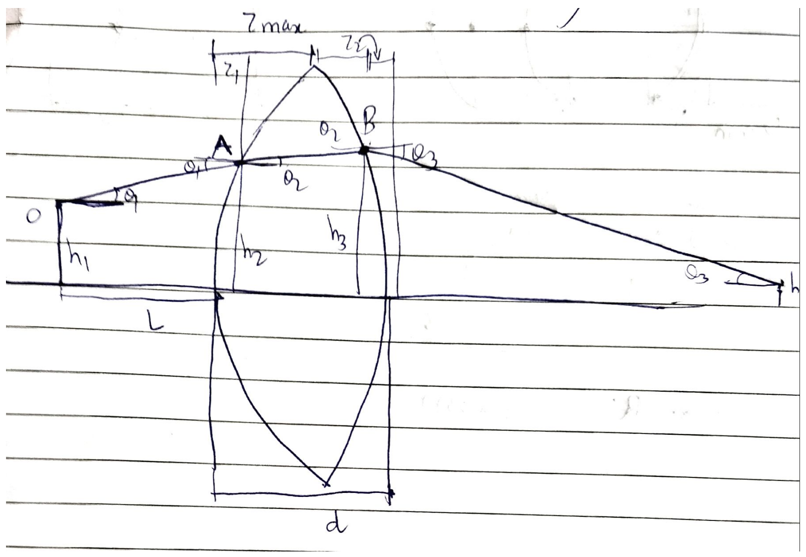

Figure 47: Notations of lens

figure 46. On the opposite side of the lens, the surface has negative radius and line passing through center is BM_1 in the figure 46. But here the surface has negative radius and its sag is defined by many ways, but here I defined from the vertex of the surface. Due to that, the function of sag would be same as Z_1. Here new quantity is Z_1max which is at the edge of the surface. Semi-diameter of the surface is distance between optical axis and the edge of the surface. Z_1max is sag at the semi-diameter height.

The distance travelled by ray into medium is given by, $D_2 = (d - 2 \cdot Z_1)/cos(\theta_2)$. d is thickness of the lens. The minimum value of the d is $d_{min} = 2 \cdot Z_1$. If the d is greater than d_{min} then lens would have three parts as shown in the figure 48. Again applying same method for surface two, we will get the θ_3 and h_3 at surface with negative radius.

The distance travelled by ray after refraction at surface two is given by $D_3 = (I + Z_1)/cos(\theta_3)$. I is the image plane distance. Z_1 would be different if the surface two is not spherical. h_3 is the height at point B in figure 46. Final height of the object at image plane is h_4. So this is ray tracing method to find optical path difference and image formation. From optical path difference, we can find the phase difference of the wave.

In the end total distance travelled by ray is $D = D_1 + D_2 + D_3$ where $D_1 = D_1 = L/cos(\theta_1) + Z_1/cos(\theta_1)$, $D_2 = D_2 = (d - 2 \cdot Z_1)/cos(\theta_2)$ and $D_3 = D_3 = (I + Z_1)/cos(\theta_3)$. To find optical distance, the D_1, D_2, D_3 are multiply by the corresponding refractive index of the medium. OPL (Optical path length) = $OPL = n_1 \cdot D_1 + n_2 \cdot D_2 + n_1 \cdot D_3$.
I took the reference ray as passing through the center of the lens so distance travelled by reference ray is, OPLR (Optical path length reference) = $OPLR = n_1 \cdot L + n_2 \cdot d + n_1 \cdot I$.
OPD (Optical path difference given by $OPD = OPL - OPLR$

From these equation we can find the phase difference from $\delta\phi = \frac{2 \cdot \pi OPD}{\lambda}$. We can use the different separate parts and its effects on the ray with python programming.Using python, I defined the total eight parts (1) convex surface with air to medium (2) convex surface with medium to air (3) concave surface with air to medium (4) concave surface with medium to air (5) air block for translation of ray (6) medium block for translation in the medium (7) refraction from plane surface (medium to air) (8) refraction from plane surface (air to medium). By combinations of all six parts, we can make any optical system in python and it gives the OPD and final height, angle with input of initial angle and height. For example, we have one optical system given in the figure 49. I divided the whole system with one block of air medium which defines the initial distance of the object. Second part is convex surface with air medium. Third one is block of medium with given thickness. Fourth part is convex surface with medium to air. Last part is block of air to define the image distance. I applied the necessary equations to find height, angle, and OPD for whole system. The code is given

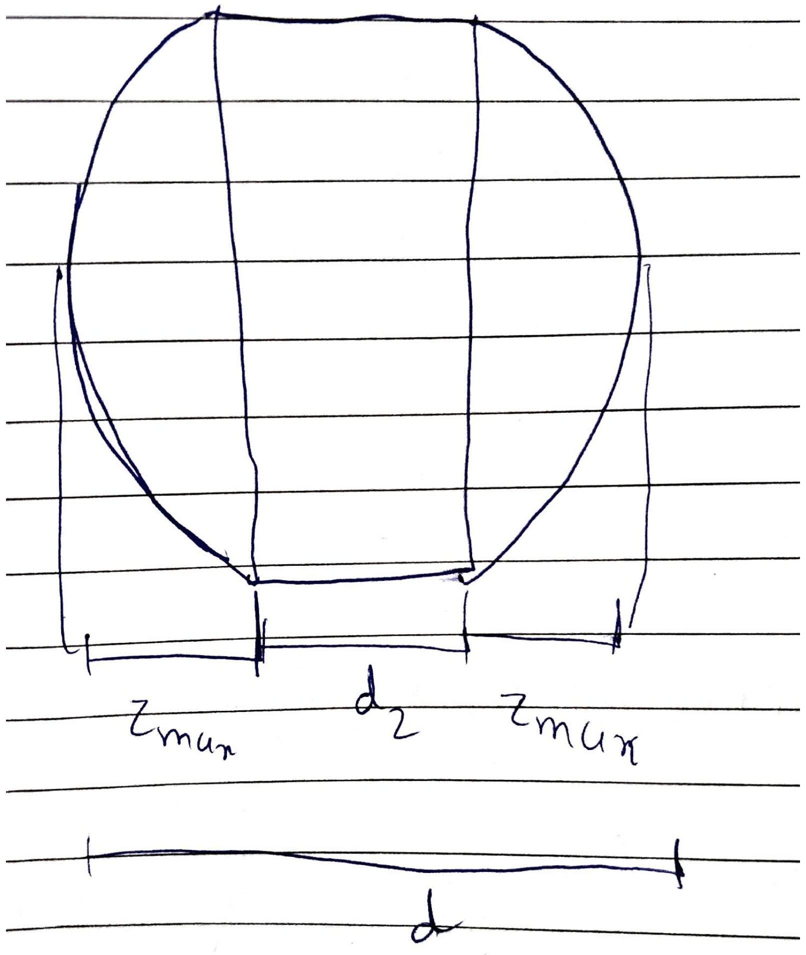

Figure 48: Minimum thickness

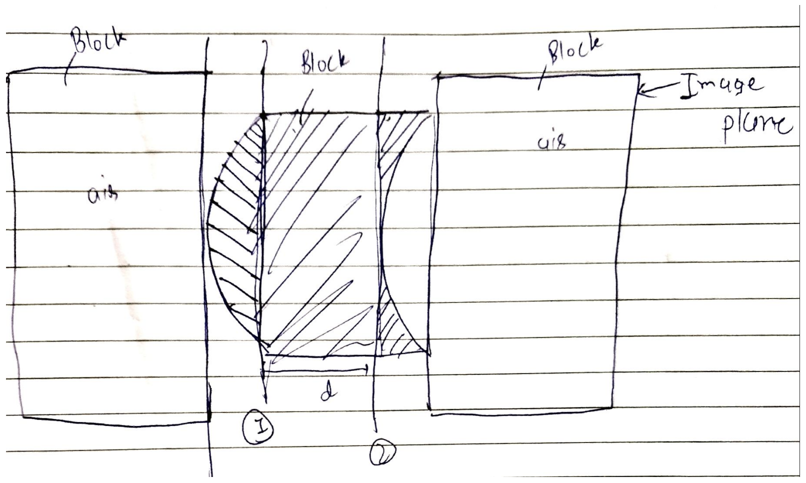

Figure 49: Optical system with parts

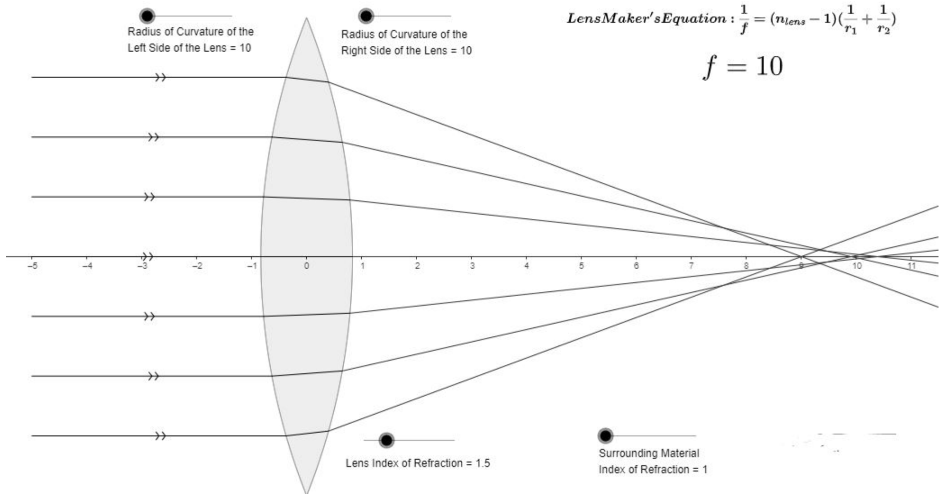
Radius of Curvature of the Left Side of the Lens = 10
Radius of Curvature of the Right Side of the Lens = 10
$LensMaker'sEquation : \frac{1}{f} = (n_{lens} - 1)(\frac{1}{r_1} + \frac{1}{r_2})$
$f = 10$
Lens Index of Refraction = 1.5
Surrounding Material Index of Refraction = 1

Figure 50: Model-1

in the appendix. I tried with simple model which is given in the figure 50. In the figure, the most of the rays are meeting at 9 to 10 mm from the lens. That means, not all rays are meeting at the one particular point which is called as spherical aberration. The radius of both curved surface has 10 mm. So image plane would be at 10 mm. The least area of confusion occurring at around 9.4 mm where the image would be clearer than other distance. The refractive index of the medium is 1.5. I applied these same optical system with same parameters with N-BK7 medium. N-BK7 has refractive index around 1.506687 at 25 celsius and pressure 101330 Pa. From that I got the plot between final image height vs initial height for angle 0 for all rays from the pupil.

Because of the refractive index for N-BK7 has slightly greater than 1.5, I took image plane distance at 9 mm. The plot is given in figure 51. The height of the image becomes very small compared to the initial. This is also agrees with the reference figure given in 50. Another plot is OPD vs initial height. Here initial height is taken between +1 mm to -1mm. So OPD would be in order of 0.01 mm. In the figure 52, The OPD is in mm and has symmetric values because of the parallel rays. This is for spherical aberration. For coma, the incident angle should be around 0.1 degree. I took incident angle as 0.1 degree and I got plots between Final height vs Initial height in the figure 53.In the figure the final height for all rays between are in positive direction above the optical axis which is also occurred in the coma aberration. In the figure 54, the plot is same as spherical aberration. Here the reference ray is taken as one which is passing through the center of the lens parallel to the optical axis. So these are all calculated data for given model which is consenting with reference figure. Now we can make arbitrary optical system to find the final image height and OPD from this model. The code is given in appendix. So these are the

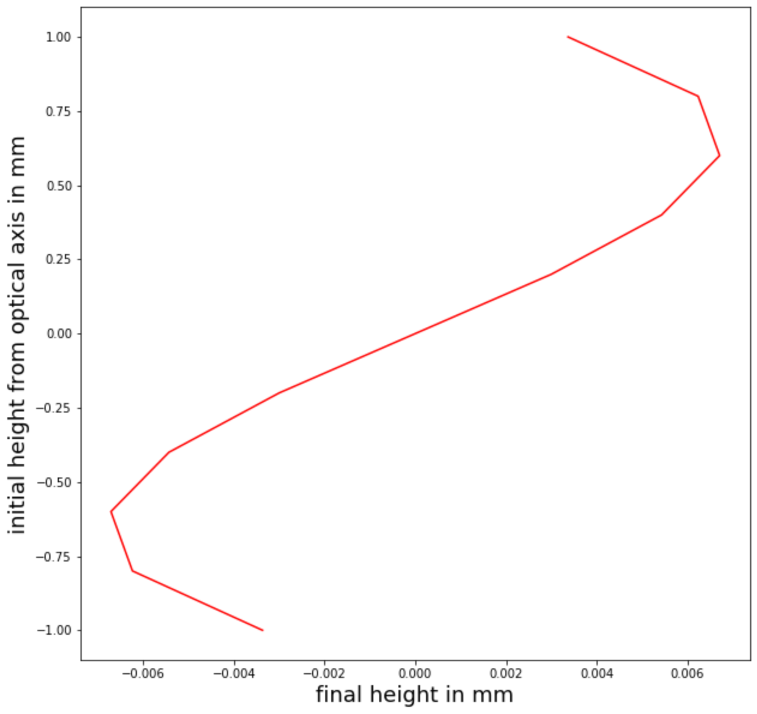

Figure 51: Spherical aberration : Final height of image vs Initial height at angle =0

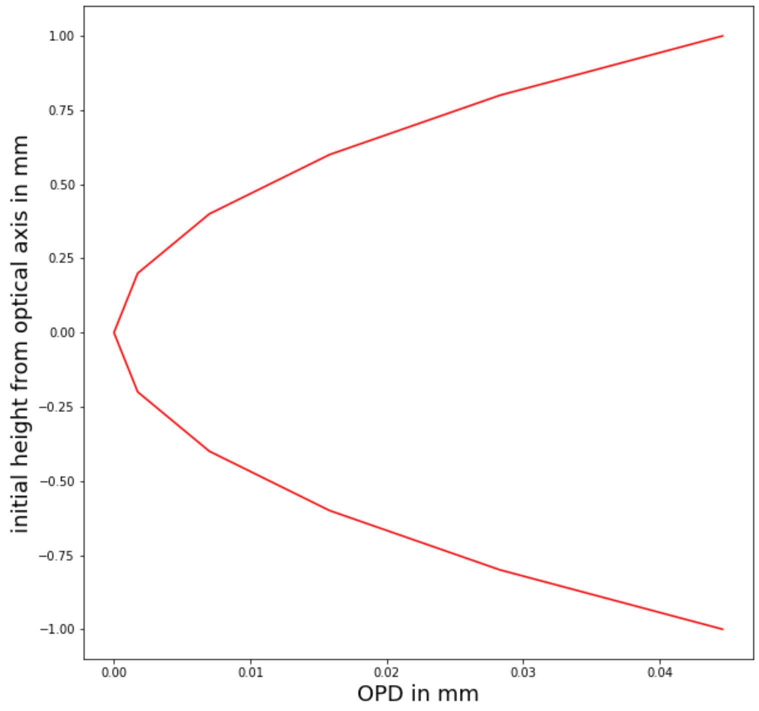

Figure 52: Spherical aberration : OPD vs initial height

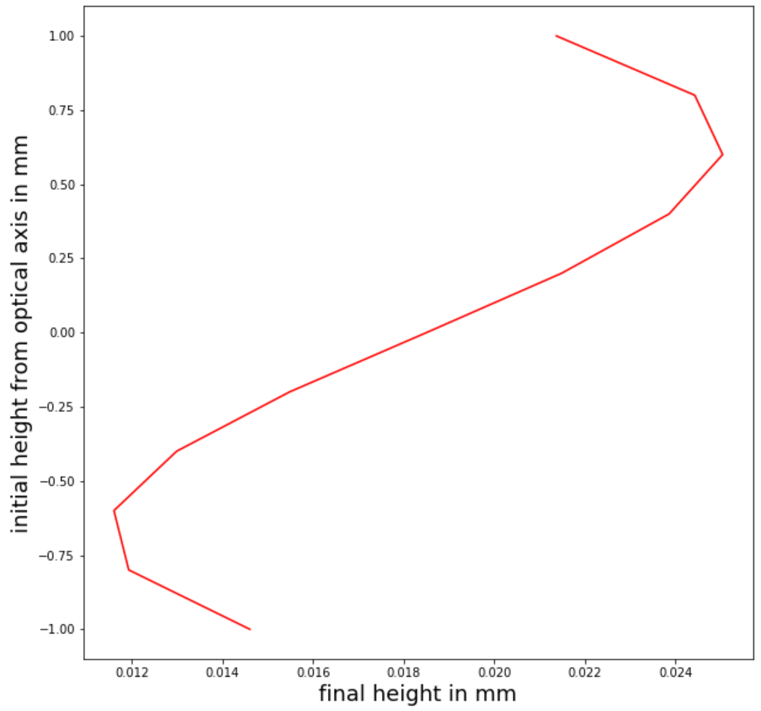

Figure 53: Coma aberration : Final height vs Initial height

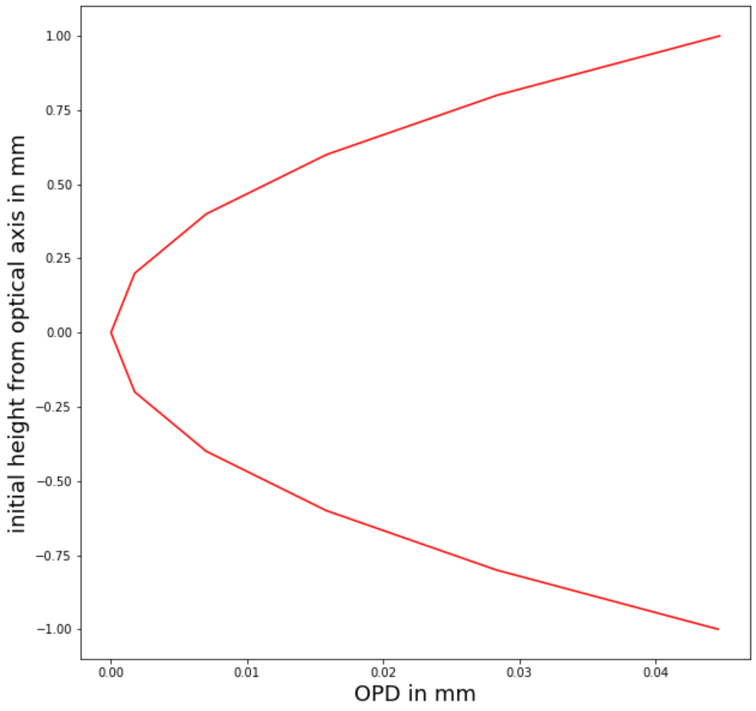

Figure 54: Coma aberration : OPD vs initial height

data for my project with good verification of each dataset.

14 Applications

The applications of the data is very useful to find dispersion relation for given glass type. As mentioned in the section 10.1.1, the plot between refractive index vs wavelength gives the dispersion relation of N-BK7. There are so many catalogues from manufacturer which are useful to find the dispersion relation for given glass type. We can also find the chromatic aberration from the dispersion relation. Chromatic aberration occurs when different wavelengths of light focus on different points.

Thermal analysis of the glass gives the proper idea about how temperature affects the refractive index of the medium as shown in the section 10.1.1 for N-BK7 and other glass types. In astronomy, telescope uses the different glass types for imaging process. Telescopes are at higher altitude from ground where the temperature and pressure are different from ground conditions. Glass made with reference temperature and pressure which are given in catalogues during manufacturing of it. Also the data and constants in catalogues are at reference temperature. To find the refractive index at given temperature and pressure we need thermal analysis of it. At higher altitudes, telescope's glass has very different conditions than ground. From thermal analysis of glass, we can find the refractive index at given conditions. Telescope needs precise image formation without aberrations. To reduce the aberrations, we need to get precise dispersion relation which is changing with temperature and pressure. SO thermal analysis of glass is very useful in telescopes.

To reduce the aberrations of the telescope or optical systems, aspheric surfaces are very useful tool. Hyperbolic curved surface has less spherical aberration than spherical curved surface. From this we can say that, to reduce higher order aberrations, we need to define the arbitrary aspheric surface and optical system which can reduce the aberrations. Asphericss surfaces and its radius at each point is defined in section 11 and 11.1 .

Ray tracing method is very reliable application for modelling of optical systems like telescope to get precise image with reduction of higher order aberrations. Ray tracing method only use the simple geometry with given aspheric surface. Using geometry, we can find the optical path length, optical path difference, final image height at image plane. These are sufficient outputs for analysing the image formation. From OPD, we can find the phase difference of the wave front. So these are the main applications of the each calculated data from this project. Below section includes the some useful references to get these data.

15 References

(1)https://docs.gitlab.com/ee/gitlab-basics/create-your-ssh-keys.html

(2)https://www.schott.com/d/advanced_optics/02ffdb0d-00a6-408f-84a5-19de56652849/1.2/tie_29_refractive _index_and_dispersion_eng.pdf

(3)https://refractiveindex.info/?shelf=glassbook=SCHOTT-SFpage=P-SF67

(4)https://nptel.ac.in/courses/112105165/lec28.pdf

(5)https://www.schott.com/d/advanced_optics/3794eded-edd2-461d-aec5-0a1d2dc9c523/1.1/schott_tie-19_temperature_coefficient_of_refractive_index_eng.pdf

(6)https://www.photonics.byu.edu/ABCD_Matrix_tut.phtml

(7)https://pdfs.semanticscholar.org/fcd9/50b8fb504522eca8e1f5bcf3c154eb8a4594.pdf

(8)https://wp.optics.arizona.edu/jcwyant/wp-content/uploads/sites/13/2016/08/03-BasicAberrations_and_Optical_Testing.pdf

(9)https://www.edmundoptics.com/ViewDocument/all-about-aspheric-lenses-en.pdf

(10)http://libres.uncg.edu/ir/uncc/f/Purcell_uncc_0694D_10102.pdf

(11)https://ophysics.com/l14.html

(12)https://www.telescope-optics.net/aberrations.htm

List of Figures

List of Tables

Listing 1: Main source code for getting all refractive index data and thermal analysis

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 29 14:42:32 2019

@author: crystal
"""
import os
import Formulas as formula_file
import Thermal_analysis as thermal_file

#providing specific glass type by its name i.e. N-SK10

def optical_describtion_glass(glass_name):

    #open the directory and files
    for filename in os.listdir(os.path.abspath('AGF\\')):
        file_name_string=str(filename)

        #open each of the file here file_string is name of the
            ↪ each file
        with open(os.path.abspath('AGF\\'+file_name_string)) as
            ↪ file:

            #file_contents_list is list of contents of files
            contents=file.read()
            file_contents_list = contents.split()

```

```
#check the condition for name in that list
if glass_name in file_contents_list :
    #print the glass name
    print('\nGlass␣name␣is␣in␣the␣'+ file_name_string + '␣
        ↪ catalog')
    print('\nGlass␣name␣is␣' + glass_name)

    #find the index of the glass name
    glass_name_index=int(file_contents_list.index(
        ↪ glass_name))

    #find the index of formula number
    formula_number=file_contents_list[glass_name_index+1]

    j=0
    with open(os.path.abspath('AGF\\'+file_name_string))
        ↪ as file:
         file_lines_list = file.readlines()

    for file_lines in file_lines_list:
        #find the details of the glass
        file_lines_wordlist = file_lines.split()
        if glass_name in file_lines_wordlist :

            for file_lines in file_lines_list[j:j+20]:

                #display constants for formula
                if 'CD' in file_lines:
                    constants_list = file_lines.split()
                    length_constants = len(constants_list)

                #wavelength range list
                if 'LD' in file_lines:
                    wavelength_ranges_list = file_lines.
                        ↪ split()

                #temperature list
                if 'TD' in file_lines:
                    temperature_list = file_lines.split()

        j=j+1

    #Total lines
    #print(j)

    #display the validity range
```

```
print("\nWavelength␣range␣for␣validity␣of␣the␣
    ↪ dispersion␣formula,␣")
print("\nMinimum␣wavelenth␣is␣" +
    ↪ wavelength_ranges_list[1] +
      \'␣micrometer␣' + 'and␣Maximum␣wavelength␣is␣␣' +
          ↪  wavelength_ranges_list[2]
      \+'␣micrometer')

#defining the validity range for dispersion formula
min_wavelength=float(wavelength_ranges_list[1])
max_wavelength=float(wavelength_ranges_list[2])

#print the temperature of that glass
print("\nReference␣temperature␣is␣" + temperature_list
    ↪ [7]+ '␣celcius')

#define the constants as per length of line in the
    ↪ file
if length_constants == 11:
     A0 = float(constants_list[1])
     A1 = float(constants_list[2])
     A2 = float(constants_list[3])
     A3 = float(constants_list[4])
     A4 = float(constants_list[5])
     A5 = float(constants_list[6])
     A6 = float(constants_list[7])
     A7 = float(constants_list[8])
     A8 = float(constants_list[9])
     A9 = float(constants_list[10])
if length_constants == 10:

     A0 = float(constants_list[1])
     A1 = float(constants_list[2])
     A2 = float(constants_list[3])
     A3 = float(constants_list[4])
     A4 = float(constants_list[5])
     A5 = float(constants_list[6])
     A6 = float(constants_list[7])
     A7 = float(constants_list[8])
     A8 = float(constants_list[9])
     A9=0

if length_constants == 9:

     A0 = float(constants_list[1])
     A1 = float(constants_list[2])
```

```
        A2 = float(constants_list[3])
        A3 = float(constants_list[4])
        A4 = float(constants_list[5])
        A5 = float(constants_list[6])
        A6 = float(constants_list[7])
        A7 = float(constants_list[8])
        A8=A9=0

    if length_constants == 8 :

        A0 = float(constants_list[1])
        A1 = float(constants_list[2])
        A2 = float(constants_list[3])
        A3 = float(constants_list[4])
        A4 = float(constants_list[5])
        A5 = float(constants_list[6])
        A6 = float(constants_list[7])
        A7=A8=A9=0

    if length_constants == 7:
        A0 = float(constants_list[1])
        A1 = float(constants_list[2])
        A2 = float(constants_list[3])
        A3 = float(constants_list[4])
        A4 = float(constants_list[5])
        A5 = float(constants_list[6])
        A6=A7=A8=A9=0

    if length_constants == 6 :

        A0 = float(constants_list[1])
        A1 = float(constants_list[2])
        A2 = float(constants_list[3])
        A3 = float(constants_list[4])
        A4 = float(constants_list[5])
        A5=A6=A7=A8=A9=0

    if length_constants == 5:
        A0 = float(constants_list[1])
        A1 = float(constants_list[2])
        A2 = float(constants_list[3])
        A3 = float(constants_list[4])
        A4=A5=A6=A7=A8=A9=0

    if length_constants == 4:
     A0 = float(constants_list[1])
     A1 = float(constants_list[2])
     A2 = float(constants_list[3])
     A3=A4=A5=A6=A7=A8=A9=0

if length_constants == 3:

     A0 = float(constants_list[1])
     A1 = float(constants_list[2])
     A2=A3=A4=A5=A6=A7=A8=A9=0

if length_constants == 2:

     A0 = float(constants_list[1])
     A1=A2=A3=A4=A5=A6=A7=A8=A9=0

#make the dictionary for all formulas
formulas_dictionary = {1: 'schott' , 2: 'sellmeier1' ,
    ↪  3: 'sellmeier2'
                      \, 4 : 'sellmeier3' , 5 : '
                          ↪ sellmeier4' , 6 :
                      \'sellmeier5', 7 : 'herzberger' ,
                          ↪ 8 : 'condrady' ,
                      \9 : 'handbook_optics1', 10 :'
                          ↪ handbook_optics2' ,
                      \11 : 'extended1', 12 : 'extended2
                          ↪ ', 13 : 'extended3' }

#apply loop for find the appropriate formula number
    ↪ and name
for formulas_number in formulas_dictionary.keys():
    if formulas_number == int(formula_number) :
        formula_name1 = formulas_dictionary[
            ↪ formulas_number]
        print('\nFormula␣name␣is␣␣:␣␣' + formula_name1)
        formula_file.formula_name(formula_name1,
            ↪ glass_name,formula_number
                                 \,min_wavelength,
                                     ↪ max_wavelength,A0
                                     ↪ ,A1,A2
                                 \,A3,A4,A5,A6,A7,A8,A9)

#thermal analysis
thermal_file.thermal_describtion_glass(glass_name,
    ↪ temperature_list,formula_number
                                      \,formula_name1,A0,
                                                                ↪ A1,A2,A3,A4,
                                                                ↪ A5,A6,A7,A8,
                                                                ↪ A9)
            #verification
            thermal_file.verification_thermal(glass_name,
                ↪ temperature_list,formula_number
                                                     \,formula_name1,A0,A1,A2
                                                        ↪ ,A3,A4,A5,A6,A7,
                                                        ↪ A8,A9)
        else:
            print('\nThere␣is␣no␣glass␣with␣name␣' + glass_name +
                ↪ '␣in␣file␣' + filename)

if __name__ == "__main__":
    glass_name = input("\nEnter␣the␣name␣of␣glass␣type␣:␣␣")
    optical_describtion_glass(glass_name)

#save the output in file
```

Here the main file uses the FORMULAS and THERMAL DESCRIPTION file to get data. Also main file uses the AGF folder which contains the all catalogues for example OHARA, SCHOTT etc. These two files are given below,

Listing 2: Formulas file

```
# -*- coding: utf-8 -*-
"""
Created on Thu May 30 12:00:25 2019

@author: crystal
"""
import os
import numpy as np
import matplotlib.pyplot as plt







#query function
def name_formula(abbe_name_formula,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)
    ↪ :
    while True:
        get_refractive_index = input("\nDo␣you␣want␣to␣get␣
            ↪ refractive␣index␣of␣specific␣wavelength␣(y/n)␣:␣␣"
            ↪ )
        if get_refractive_index == 'y':
            wavelength = input("\n\n␣Please␣enter␣the␣wavelength␣
                ↪ in␣nanometer␣:␣␣")
            wavelength = float(wavelength)/1000

            print("\nRefractive␣index␣is␣␣")
            print(abbe_name_formula(wavelength,A0,A1,A2,A3,A4,A5,
                ↪ A6,A7,A8,A9))
        else:
            break








#defining function or data set for each formula
def describtion_glass(glass_name,formula_number,A0,A1,A2,A3,A4,A5
    ↪ ,A6,A7,A8,A9):

    #details of glass
    print('\nGlass␣dispersion␣formula␣number␣is␣' +
        ↪ formula_number)
    print("\nConstants␣values␣for␣␣are,␣\n\nA0␣:␣" , end='')
    print(A0)
    print('\nA1␣:␣',end='' )
    print(A1)
    print('\nA2␣:␣',end='' )
    print(A2)
    print('\nA3␣:␣',end='' )
    print(A3)
    print('\nA4␣:␣',end='' )
    print(A4)
    print('\nA5␣:␣',end='' )
    print(A5)
    print('\nA6␣:␣',end='' )
    print(A6)
    print('\nA7␣:␣',end = '' )
    print(A7)
    print('\nA8␣:␣',end = '' )
    print(A8)
    print('\nA9␣:␣' ,end='')
    print(A9)
```

```
def plot(n,wavelength,glass_name,formula_number):
     plot_name = str(glass_name +'_'+ str(formula_number))
     wavelength_nanometer=wavelength*1000
     plt.plot(wavelength_nanometer,n,'r-')
     plt.xlabel('Wavelength␣\u03BB␣(nm)␣' , fontsize= 14)
     plt.ylabel('Refractive␣Index␣n␣' , fontsize = 14 )
     plt.savefig(os.path.abspath('Plots\\'+'RI_Wavlength_'+
         ↪ plot_name+'.png'))
     plt.show()









#abbe number
def abbe_schott(wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):
    n = ((A0) + (A1*(wavelength*wavelength)) + (A2/(wavelength*
        ↪ wavelength)) + (A3/(wavelength*wavelength*wavelength*
        ↪ wavelength)) + (A4/(wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength)) + (A5/(wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength)))**(1/2)
    return n
#formulas for all dispersion
def schott(glass_name,formula_number,min_wavelength,
    ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):

    describtion_glass(glass_name,formula_number,A0,A1,A2,A3,A4,A5,
        ↪ A6,A7,A8,A9)

    #plot
    wavelength=np.arange(min_wavelength,max_wavelength,0.0001)
    n = ((A0) + (A1*(wavelength*wavelength)) + (A2/(wavelength*
        ↪ wavelength)) + (A3/(wavelength*wavelength*wavelength*
        ↪ wavelength)) + (A4/(wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength)) + (A5/(wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength)))**(1/2)
    plot(n,wavelength,glass_name,formula_number)

    #abbe number
    #def abbe_schott(wavelength):
        #n = ((A0) + (A1*(wavelength*wavelength)) + (A2/(
            ↪ wavelength*wavelength)) + (A3/(wavelength*
            ↪ wavelength*wavelength*wavelength)) + (A4/(
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength)) + (A5/(wavelength*
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength*wavelength)))**(1/2)
        #return n

    abbe_number = (abbe_schott(0.5892,A0,A1,A2,A3,A4,A5,A6,A7,A8,
        ↪ A9)-1)/(abbe_schott(0.4861,A0,A1,A2,A3,A4,A5,A6,A7,A8,
        ↪ A9)-abbe_schott(0.6563,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9))
    print("Abbe␣number␣of␣this␣glass␣type␣is␣␣" )
    print(abbe_number)

    #calling function for query function
    name_formula(abbe_schott,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)









#abbe number
def abbe_sellmeier1(wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):
    n = np.sqrt(1.0000000000 + (A0*(wavelength*wavelength)/(
        ↪ wavelength*wavelength - A1)) + (A2*wavelength*
        ↪ wavelength/(wavelength*wavelength - A3)) + (A4*
        ↪ wavelength*wavelength/(wavelength*wavelength - A5)))
    return n
def sellmeier1(glass_name,formula_number,min_wavelength,
    ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):

    describtion_glass(glass_name,formula_number,A0,A1,A2,A3,A4,A5,
        ↪ A6,A7,A8,A9)
    #plot
    wavelength=np.arange(min_wavelength,max_wavelength,0.0001)
    n = np.sqrt(1.0000000000 + (A0*(wavelength*wavelength)/(
        ↪ wavelength*wavelength - A1)) + (A2*wavelength*
        ↪ wavelength/(wavelength*wavelength - A3)) + (A4*
        ↪ wavelength*wavelength/(wavelength*wavelength - A5)))
    plot(n,wavelength,glass_name,formula_number)

    #abbe number
    #def abbe_sellmeier1(wavelength):
        #n = np.sqrt(1.0000000000 + (A0*(wavelength*wavelength)/(
            ↪ wavelength*wavelength - A1)) + (A2*wavelength*
            ↪ wavelength/(wavelength*wavelength - A3)) + (A4*
            ↪ wavelength*wavelength/(wavelength*wavelength - A5))
            ↪ )
        #return n

    abbe_number = (abbe_sellmeier1(0.5892,A0,A1,A2,A3,A4,A5,A6,A7,
        ↪ A8,A9)-1)/(abbe_sellmeier1(0.4861,A0,A1,A2,A3,A4,A5,A6,
        ↪ A7,A8,A9)-abbe_sellmeier1(0.6563,A0,A1,A2,A3,A4,A5,A6,
        ↪ A7,A8,A9))
    print("Abbe␣number␣of␣this␣glass␣type␣is␣␣")
    print(abbe_number)

    #calling function for query function
    name_formula(abbe_sellmeier1,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)









#abbe number
def abbe_sellmeier2(wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):
    n = (1.0000000 + A0 + A1*(wavelength*wavelength)/((wavelength*
        ↪ wavelength)-(A2*A2)) + A3/((wavelength*wavelength)-(A4*
        ↪ A4)))**(1/2)
    return n
def sellmeier2(glass_name,formula_number,min_wavelength,
    ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):

    describtion_glass(glass_name,formula_number,A0,A1,A2,A3,A4,A5,
        ↪ A6,A7,A8,A9)

    #plot
    wavelength=np.arange(min_wavelength,max_wavelength,0.0001)
    n = (1.0000000 + A0 + A1*(wavelength*wavelength)/((wavelength*
        ↪ wavelength)-(A2*A2)) + A3/((wavelength*wavelength)-(A4*
        ↪ A4)))**(1/2)
    plot(n,wavelength,glass_name,formula_number)

    #abbe number
    #def abbe_sellmeier2(wavelength):
        #n = (1.0000000 + A0 + A1*(wavelength*wavelength)/((
            ↪ wavelength*wavelength)-(A2*A2)) + A3/((wavelength*
            ↪ wavelength)-(A4*A4)))**(1/2)
        #return n

    abbe_number = (abbe_sellmeier2(0.5892,A0,A1,A2,A3,A4,A5,A6,A7,
        ↪ A8,A9)-1)/(abbe_sellmeier2(0.4861,A0,A1,A2,A3,A4,A5,A6,
        ↪ A7,A8,A9)-abbe_sellmeier2(0.6563,A0,A1,A2,A3,A4,A5,A6,
        ↪ A7,A8,A9))
    print("Abbe␣number␣of␣this␣glass␣type␣is␣␣")
    print(abbe_number)

    #calling function for query function
    name_formula(abbe_sellmeier2,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)







#abbe number
def abbe_sellmeier3(wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):
    n = (1.0000000000 + (A0*(wavelength*wavelength)/(wavelength*
        ↪ wavelength - A1)) + (A2*wavelength*wavelength/(
        ↪ wavelength*wavelength - A3)) + (A4*wavelength*
        ↪ wavelength/(wavelength*wavelength - A5)) + (A6*
        ↪ wavelength*wavelength/(wavelength*wavelength - A7)))
        ↪ **(1/2)
    return n
def sellmeier3(glass_name,formula_number,min_wavelength,
    ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):

    describtion_glass(glass_name,formula_number,A0,A1,A2,A3,A4,A5,
        ↪ A6,A7,A8,A9)

    #plot
    wavelength=np.arange(min_wavelength,max_wavelength,0.0001)
    n = (1.0000000000 + (A0*(wavelength*wavelength)/(wavelength*
        ↪ wavelength - A1)) + (A2*wavelength*wavelength/(
        ↪ wavelength*wavelength - A3)) + (A4*wavelength*
        ↪ wavelength/(wavelength*wavelength - A5)) + (A6*
        ↪ wavelength*wavelength/(wavelength*wavelength - A7)))
        ↪ **(1/2)
    plot(n,wavelength,glass_name,formula_number)

    #abbe number
    #def abbe_sellmeier3(wavelength):
        #n = (1.0000000000 + (A0*(wavelength*wavelength)/(
            ↪ wavelength*wavelength - A1)) + (A2*wavelength*
            ↪ wavelength/(wavelength*wavelength - A3)) + (A4*
            ↪ wavelength*wavelength/(wavelength*wavelength - A5))
            ↪  + (A6*wavelength*wavelength/(wavelength*wavelength
            ↪  - A7)))**(1/2)
        #return n

    abbe_number = (abbe_sellmeier3(0.5892,A0,A1,A2,A3,A4,A5,A6,A7,
        ↪ A8,A9)-1)/(abbe_sellmeier3(0.4861,A0,A1,A2,A3,A4,A5,A6,
        ↪ A7,A8,A9)-abbe_sellmeier3(0.6563,A0,A1,A2,A3,A4,A5,A6,
        ↪ A7,A8,A9))
    print("Abbe␣number␣of␣this␣glass␣type␣is␣␣")
    print(abbe_number)

    #calling function for query function
    name_formula(abbe_sellmeier3,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)









#abbe number
def abbe_sellmeier4(wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):
    n = (A0 + A1*wavelength*wavelength/(wavelength*wavelength - A2
        ↪ ) + (A3*wavelength*wavelength/(wavelength*wavelength -
        ↪ A4)))**(1/2)
    return n
def sellmeier4(glass_name,formula_number,min_wavelength,
    ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):

    describtion_glass(glass_name,formula_number,A0,A1,A2,A3,A4,A5,
        ↪ A6,A7,A8,A9)

    #plot
    wavelength=np.arange(min_wavelength,max_wavelength,0.0001)
    n = (A0 + (A1*wavelength*wavelength/((wavelength*wavelength) -
        ↪  A2)) + (A3*wavelength*wavelength/((wavelength*
        ↪ wavelength) - A4)))**(1/2)
    plot(n,wavelength,glass_name,formula_number)

    #abbe number
    #def abbe_sellmeier4(wavelength):
        #n = (A0 + A1*wavelength*wavelength/(wavelength*wavelength
            ↪  - A2) + (A3*wavelength*wavelength/(wavelength*
            ↪ wavelength - A4)))**(1/2)
        #return n

    abbe_number = (abbe_sellmeier4(0.5892,A0,A1,A2,A3,A4,A5,A6,A7,
        ↪ A8,A9)-1)/(abbe_sellmeier4(0.4861,A0,A1,A2,A3,A4,A5,A6,
        ↪ A7,A8,A9)-abbe_sellmeier4(0.6563,A0,A1,A2,A3,A4,A5,A6,
        ↪ A7,A8,A9))
    print("Abbe␣number␣of␣this␣glass␣type␣is␣␣")
    print(abbe_number)

    #calling function for query function
    name_formula(abbe_sellmeier4,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)








#abbe number
def abbe_sellmeier5(wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):
    n = (1.00000000 + (A0*wavelength*wavelength/(wavelength*
        ↪ wavelength - A1)) + (A2*wavelength*wavelength/(
        ↪ wavelength*wavelength - A3)) + (A4*wavelength*
        ↪ wavelength/(wavelength*wavelength - A5)) + (A6*
        ↪ wavelength*wavelength/(wavelength*wavelength - A7)) + (
        ↪ A8*wavelength*wavelength/(wavelength*wavelength - A9)))
        ↪ **(1/2)
    return n
def sellmeier5(glass_name,formula_number,min_wavelength,
    ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):

    describtion_glass(glass_name,formula_number,A0,A1,A2,A3,A4,A5,
        ↪ A6,A7,A8,A9)

    #plot
    wavelength=np.arange(min_wavelength,max_wavelength,0.0001)
    n = (1.00000000 + (A0*wavelength*wavelength/(wavelength*
        ↪ wavelength - A1)) + (A2*wavelength*wavelength/(
        ↪ wavelength*wavelength - A3)) + (A4*wavelength*
        ↪ wavelength/(wavelength*wavelength - A5)) + (A6*
        ↪ wavelength*wavelength/(wavelength*wavelength - A7)) + (
        ↪ A8*wavelength*wavelength/(wavelength*wavelength - A9)))
        ↪ **(1/2)
    plot(n,wavelength,glass_name,formula_number)

    #abbe number
    #def abbe_sellmeier5(wavelength):
        #n = (1.00000000 + (A0*wavelength*wavelength/(wavelength*
            ↪ wavelength - A1)) + (A2*wavelength*wavelength/(
            ↪ wavelength*wavelength - A3)) + (A4*wavelength*
            ↪ wavelength/(wavelength*wavelength - A5)) + (A6*
            ↪ wavelength*wavelength/(wavelength*wavelength - A7))
            ↪  + (A8*wavelength*wavelength/(wavelength*wavelength
            ↪  - A9)))**(1/2)
        #return n

    abbe_number = (abbe_sellmeier5(0.5892,A0,A1,A2,A3,A4,A5,A6,A7,
        ↪ A8,A9)-1)/(abbe_sellmeier5(0.4861,A0,A1,A2,A3,A4,A5,A6,
        ↪ A7,A8,A9)-abbe_sellmeier5(0.6563,A0,A1,A2,A3,A4,A5,A6,
        ↪ A7,A8,A9))
    print("Abbe␣number␣of␣this␣glass␣type␣is␣␣")
    print(abbe_number)

    #calling function for query function
    name_formula(abbe_sellmeier5,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)
```

```
#abbe number
def abbe_herzberger(wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):
    n = ( A0 + A1*(1/((wavelength*wavelength)-0.028)) + A2*(1/((
        ↪ wavelength*wavelength)-0.028))*(1/((wavelength*
        ↪ wavelength)-0.028)) + A3*wavelength*wavelength + A4*
        ↪ wavelength*wavelength*wavelength*wavelength + A5*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength)**(1/2)
    return n
def herzberger(glass_name,formula_number,min_wavelength,
    ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):

    describtion_glass(glass_name,formula_number,A0,A1,A2,A3,A4,A5,
        ↪ A6,A7,A8,A9)

    #plot
    wavelength=np.arange(min_wavelength,max_wavelength,0.0001)
    n = ( A0 + A1*(1/((wavelength*wavelength)-0.028)) + A2*(1/((
        ↪ wavelength*wavelength)-0.028))*(1/((wavelength*
        ↪ wavelength)-0.028)) + A3*wavelength*wavelength + A4*
        ↪ wavelength*wavelength*wavelength*wavelength + A5*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength)**(1/2)
    plot(n,wavelength,glass_name,formula_number)

    #abbe number
    #def abbe_herzberger(wavelength):
        #n = ( A0 + A1*(1/((wavelength*wavelength)-0.028)) + A2
            ↪ *(1/((wavelength*wavelength)-0.028))*(1/((
            ↪ wavelength*wavelength)-0.028)) + A3*wavelength*
            ↪ wavelength + A4*wavelength*wavelength*wavelength*
            ↪ wavelength + A5*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength*wavelength)**(1/2)
        #return n

    abbe_number = (abbe_herzberger(0.5892,A0,A1,A2,A3,A4,A5,A6,A7,
        ↪ A8,A9)-1)/(abbe_herzberger(0.4861,A0,A1,A2,A3,A4,A5,A6,
        ↪ A7,A8,A9)-abbe_herzberger(0.6563,A0,A1,A2,A3,A4,A5,A6,
        ↪ A7,A8,A9))
    print("Abbe␣number␣of␣this␣glass␣type␣is␣␣")
    print(abbe_number)
    #calling function for query function
    name_formula(abbe_herzberger,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)










#abbe number
def abbe_conrady(wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):
    n = ( A0 + A1/(wavelength) + (A2/(wavelength)**(3.5)))
    return n
def conrady(glass_name,formula_number,min_wavelength,
    ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):

    describtion_glass(glass_name,formula_number,A0,A1,A2,A3,A4,A5,
        ↪ A6,A7,A8,A9)

    #plot
    wavelength=np.arange(min_wavelength,max_wavelength,0.0001)
    n = ( A0 + A1/(wavelength) + (A2/(wavelength)**(3.5)))
    plot(n,wavelength,glass_name,formula_number)

    #abbe number
    #def abbe_conrady(wavelength):
        #n = ( A0 + A1/(wavelength) + (A2/(wavelength)**(3.5)))
        #return n

    abbe_number = (abbe_conrady(0.5892,A0,A1,A2,A3,A4,A5,A6,A7,A8,
        ↪ A9)-1)/(abbe_conrady(0.4861,A0,A1,A2,A3,A4,A5,A6,A7,A8,
        ↪ A9)-abbe_conrady(0.6563,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9))
    print("Abbe␣number␣of␣this␣glass␣type␣is␣␣")
    print(abbe_number)

    #calling function for query function
    name_formula(abbe_conrady,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)
```

```
#abbe number
def abbe_handbook_optics1(wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,
    ↪ A9):
    n = ((A0 + A1/((wavelength*wavelength)-A2) - A3*wavelength*
        ↪ wavelength))**(1/2)
    return n
def handbook_optics1(glass_name,formula_number,min_wavelength,
    ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):

    describtion_glass(glass_name,formula_number,A0,A1,A2,A3,A4,A5,
        ↪ A6,A7,A8,A9)

    #plot
    wavelength=np.arange(min_wavelength,max_wavelength,0.0001)
    n = ((A0 + A1/((wavelength*wavelength)-A2) - A3*wavelength*
        ↪ wavelength))**(1/2)
    plot(n,wavelength,glass_name,formula_number)

    #abbe number
    #def abbe_handbook_optics1(wavelength):
        #n = ((A0 + A1/((wavelength*wavelength)-A2) - A3*
            ↪ wavelength*wavelength))**(1/2)
        #return n

    abbe_number = (abbe_handbook_optics1(0.5892,A0,A1,A2,A3,A4,A5,
        ↪ A6,A7,A8,A9)-1)/(abbe_handbook_optics1(0.4861,A0,A1,A2,
        ↪ A3,A4,A5,A6,A7,A8,A9)-abbe_handbook_optics1(0.6563,A0,
        ↪ A1,A2,A3,A4,A5,A6,A7,A8,A9))
    print("Abbe␣number␣of␣this␣glass␣type␣is␣␣")
    print(abbe_number)

    #calling function for query function
    name_formula(abbe_handbook_optics1,A0,A1,A2,A3,A4,A5,A6,A7,A8,
        ↪ A9)
```

```
#abbe number
def abbe_handbook_optics2(wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,
    ↪ A9):
    n = ((A0 + (A1*wavelength*wavelength)/((wavelength*wavelength)
        ↪ -A2) - A3*wavelength*wavelength))**(1/2)
    return n
def handbook_optics2(glass_name,formula_number,min_wavelength,
    ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):

    describtion_glass(glass_name,formula_number,A0,A1,A2,A3,A4,A5,
        ↪ A6,A7,A8,A9)

    #plot
    wavelength=np.arange(min_wavelength,max_wavelength,0.0001)
    n = ((A0 + (A1*wavelength*wavelength)/((wavelength*wavelength)
        ↪ -A2) - A3*wavelength*wavelength))**(1/2)
    plot(n,wavelength,glass_name,formula_number)

    #abbe number
    #def abbe_handbook_optics2(wavelength):
        #n = ((A0 + (A1*wavelength*wavelength)/((wavelength*
            ↪ wavelength)-A2) - A3*wavelength*wavelength))**(1/2)
        #return n

    abbe_number = (abbe_handbook_optics2(0.5892,A0,A1,A2,A3,A4,A5,
        ↪ A6,A7,A8,A9)-1)/(abbe_handbook_optics2(0.4861,A0,A1,A2,
        ↪ A3,A4,A5,A6,A7,A8,A9)-abbe_handbook_optics2(0.6563,A0,
        ↪ A1,A2,A3,A4,A5,A6,A7,A8,A9))
    print("Abbe␣number␣of␣this␣glass␣type␣is␣␣")
    print(abbe_number)

    #calling function for query function
    name_formula(abbe_handbook_optics2,A0,A1,A2,A3,A4,A5,A6,A7,A8,
        ↪ A9)









#abbe number
def abbe_extended1(wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):
    n = ((A0) + (A1*(wavelength*wavelength)) + (A2/(wavelength*
        ↪ wavelength)) + (A3/(wavelength*wavelength*wavelength*
        ↪ wavelength)) + (A4/(wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength)) + (A5/(wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength)) + (A6/(wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength)) + (A7/(wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength)))**(1/2)
    return n
def extended1(glass_name,formula_number,min_wavelength,
    ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):

    describtion_glass(glass_name,formula_number,A0,A1,A2,A3,A4,A5,
        ↪ A6,A7,A8,A9)

    #plot
    wavelength=np.arange(min_wavelength,max_wavelength,0.0001)
    n = ((A0) + (A1*(wavelength*wavelength)) + (A2/(wavelength*
        ↪ wavelength)) + (A3/(wavelength*wavelength*wavelength*
        ↪ wavelength)) + (A4/(wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength)) + (A5/(wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength)) + (A6/(wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength)) + (A7/(wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength)))**(1/2)
    plot(n,wavelength,glass_name,formula_number)

    #abbe number
    #def abbe_extended1(wavelength):
        #n = ((A0) + (A1*(wavelength*wavelength)) + (A2/(
            ↪ wavelength*wavelength)) + (A3/(wavelength*
            ↪ wavelength*wavelength*wavelength)) + (A4/(
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength)) + (A5/(wavelength*
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength*wavelength)) + (A6/(
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength)) + (A7/(wavelength*
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength*wavelength)))**(1/2)
        #return n

    abbe_number = (abbe_extended1(0.5892,A0,A1,A2,A3,A4,A5,A6,A7,
        ↪ A8,A9)-1)/(abbe_extended1(0.4861,A0,A1,A2,A3,A4,A5,A6,
        ↪ A7,A8,A9)-abbe_extended1(0.6563,A0,A1,A2,A3,A4,A5,A6,A7
        ↪ ,A8,A9))
    print("Abbe␣number␣of␣this␣glass␣type␣is␣␣")
    print(abbe_number)

    #calling function for query function
    name_formula(abbe_extended1,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)










#abbe number
def abbe_extended2(wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):
    n = ((A0) + (A1*(wavelength*wavelength)) + (A2/(wavelength*
        ↪ wavelength)) + (A3/(wavelength*wavelength*wavelength*
        ↪ wavelength)) + (A4/(wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength)) + (A5/(wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength)) + (A6*wavelength*wavelength*
        ↪ wavelength*wavelength) + (A7*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength))**(1/2)
    return n
def extended2(glass_name,formula_number,min_wavelength,
    ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):

    describtion_glass(glass_name,formula_number,A0,A1,A2,A3,A4,A5,
        ↪ A6,A7,A8,A9)

    #plot
    wavelength=np.arange(min_wavelength,max_wavelength,0.0001)
    n = ((A0) + (A1*(wavelength*wavelength)) + (A2/(wavelength*
        ↪ wavelength)) + (A3/(wavelength*wavelength*wavelength*
        ↪ wavelength)) + (A4/(wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength)) + (A5/(wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength)) + (A6*wavelength*wavelength*
        ↪ wavelength*wavelength) + (A7*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength))**(1/2)
    plot(n,wavelength,glass_name,formula_number)

    #abbe number
    #def abbe_extended2(wavelength):
        #n = ((A0) + (A1*(wavelength*wavelength)) + (A2/(
            ↪ wavelength*wavelength)) + (A3/(wavelength*
            ↪ wavelength*wavelength*wavelength)) + (A4/(
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength)) + (A5/(wavelength*
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength*wavelength)) + (A6*wavelength
            ↪ *wavelength*wavelength*wavelength) + (A7*wavelength
            ↪ *wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength))**(1/2)
        #return n

    abbe_number = (abbe_extended2(0.5892,A0,A1,A2,A3,A4,A5,A6,A7,
        ↪ A8,A9)-1)/(abbe_extended2(0.4861,A0,A1,A2,A3,A4,A5,A6,
        ↪ A7,A8,A9)-abbe_extended2(0.6563,A0,A1,A2,A3,A4,A5,A6,A7
        ↪ ,A8,A9))
    print("Abbe␣number␣of␣this␣glass␣type␣is␣␣")
    print(abbe_number)

    #calling function for query function
    name_formula(abbe_extended2,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)






#abbe number
def abbe_extended3(wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):
    n = ((A0) + (A1*(wavelength*wavelength)) + (A2*wavelength*
        ↪ wavelength*wavelength*wavelength) + (A3/(wavelength*
        ↪ wavelength)) + (A4/(wavelength*wavelength*wavelength*
        ↪ wavelength)) + (A5/(wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength)) + (A6/(wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength)) + (A7/(wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength)) + (A8/(wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength)))**(1/2)
    return n
def extended3(glass_name,formula_number,min_wavelength,
    ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9):

    describtion_glass(glass_name,formula_number,A0,A1,A2,A3,A4,A5,
        ↪ A6,A7,A8,A9)

    #plot
    wavelength=np.arange(min_wavelength,max_wavelength,0.0001)
    n = ((A0) + (A1*(wavelength*wavelength)) + (A2*wavelength*
        ↪ wavelength*wavelength*wavelength) + (A3/(wavelength*
        ↪ wavelength)) + (A4/(wavelength*wavelength*wavelength*
        ↪ wavelength)) + (A5/(wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength)) + (A6/(wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength)) + (A7/(wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength)) + (A8/(wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength*wavelength*wavelength*wavelength*wavelength*
        ↪ wavelength)))**(1/2)
    plot(n,wavelength,glass_name,formula_number)

    #abbe number
    #def abbe_extended3(wavelength):
        #n = ((A0) + (A1*(wavelength*wavelength)) + (A2*wavelength
            ↪ *wavelength*wavelength*wavelength) + (A3/(
            ↪ wavelength*wavelength)) + (A4/(wavelength*
            ↪ wavelength*wavelength*wavelength)) + (A5/(
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength)) + (A6/(wavelength*
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength*wavelength)) + (A7/(
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength)) + (A8/(wavelength*
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength*wavelength*wavelength*
            ↪ wavelength*wavelength*wavelength)))**(1/2)
        #return n

    abbe_number = (abbe_extended3(0.5892,A0,A1,A2,A3,A4,A5,A6,A7,
        ↪ A8,A9)-1)/(abbe_extended3(0.4861,A0,A1,A2,A3,A4,A5,A6,
        ↪ A7,A8,A9)-abbe_extended3(0.6563,A0,A1,A2,A3,A4,A5,A6,A7
        ↪ ,A8,A9))
    print("Abbe␣number␣of␣this␣glass␣type␣is␣␣")
    print(abbe_number)

    #calling function for query function
    name_formula(abbe_extended3,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)









#to check formula number and name
def formula_name(formula_name1,glass_name,formula_number,
    ↪ min_wavelength,max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,
    ↪ A9):

        if formula_name1 == 'schott':
            schott(glass_name,formula_number,min_wavelength,
                ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)
        if formula_name1 == 'sellmeier1':
            sellmeier1(glass_name,formula_number,min_wavelength,
                ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)
        if formula_name1 == 'sellmeier2':
            sellmeier2(glass_name,formula_number,min_wavelength,
                ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)
        if formula_name1 == 'sellmeier3':
            sellmeier3(glass_name,formula_number,min_wavelength,
                ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)
        if formula_name1 == 'sellmeier4':
            sellmeier4(glass_name,formula_number,min_wavelength,
                ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)
        if formula_name1 == 'sellmeier5':
            sellmeier5(glass_name,formula_number,min_wavelength,
                ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)
        if formula_name1 == 'herzberger':
            herzberger(glass_name,formula_number,min_wavelength,
                ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)
        if formula_name1 == 'condrady':
            conrady(glass_name,formula_number,min_wavelength,
                ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)
        if formula_name1 == 'handbook_optics1':
            handbook_optics1(glass_name,formula_number,
                ↪ min_wavelength,max_wavelength,A0,A1,A2,A3,A4,A5,
                ↪ A6,A7,A8,A9)
        if formula_name1 == 'handbook_optics2':
            handbook_optics2(glass_name,formula_number,
                ↪ min_wavelength,max_wavelength,A0,A1,A2,A3,A4,A5,
                ↪ A6,A7,A8,A9)
        if formula_name1 == 'extended1':
            extended1(glass_name,formula_number,min_wavelength,
                ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)
        if formula_name1 == 'extended2':
            extended2(glass_name,formula_number,min_wavelength,
                ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)
        if formula_name1 == 'extended3':
            extended3(glass_name,formula_number,min_wavelength,
                ↪ max_wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)








#purpose of this function to find RI for thermal analysis
def formula_thermal_name(formula_name1,wavelength,A0,A1,A2,A3,A4,
    ↪ A5,A6,A7,A8,A9):
        if formula_name1 == 'schott':
            n0 = abbe_schott(wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8
                ↪ ,A9)
            return n0
        if formula_name1 == 'sellmeier1':
            n0 = abbe_sellmeier1(wavelength,A0,A1,A2,A3,A4,A5,A6,
                ↪ A7,A8,A9)
            return n0
        if formula_name1 == 'sellmeier2' :
            n0 = abbe_sellmeier2(wavelength,A0,A1,A2,A3,A4,A5,A6,
                ↪ A7,A8,A9)
            return n0
        if formula_name1 == 'sellmeier3':
            n0 = abbe_sellmeier3(wavelength,A0,A1,A2,A3,A4,A5,A6,
                ↪ A7,A8,A9)
            return n0
        if formula_name1 == 'sellmeier4':
            n0 = abbe_sellmeier4(wavelength,A0,A1,A2,A3,A4,A5,A6,
                ↪ A7,A8,A9)
            return n0
        if formula_name1 == 'sellmeier5':
            n0 = abbe_sellmeier5(wavelength,A0,A1,A2,A3,A4,A5,A6,
                ↪ A7,A8,A9)
            return n0
        if formula_name1 == 'herzberger':
            n0 = abbe_herzberger(wavelength,A0,A1,A2,A3,A4,A5,A6,
                ↪ A7,A8,A9)
            return n0
        if formula_name1 == 'condrady':
            n0 = abbe_conrady(wavelength,A0,A1,A2,A3,A4,A5,A6,A7,
                ↪ A8,A9)
            return n0
        if formula_name1 == 'handbook_optics1':
            n0 = abbe_handbook_optics1(wavelength,A0,A1,A2,A3,A4,
                ↪ A5,A6,A7,A8,A9)
            return n0
        if formula_name1 == 'handbook_optics2':
            n0 = abbe_handbook_optics2(wavelength,A0,A1,A2,A3,A4,
                ↪ A5,A6,A7,A8,A9)
            return n0
        if formula_name1 == 'extended1':
            n0 = abbe_extended1(wavelength,A0,A1,A2,A3,A4,A5,A6,A7
                ↪ ,A8,A9)
            return n0
        if formula_name1 == 'extended2':
            n0 = abbe_extended2(wavelength,A0,A1,A2,A3,A4,A5,A6,A7
                ↪ ,A8,A9)
            return n0
        if formula_name1 == 'extended3':
            n0 = abbe_extended3(wavelength,A0,A1,A2,A3,A4,A5,A6,A7
                ↪ ,A8,A9)
            return n0
```

Thermal description file is given below which is imported in the main file,

Listing 3: Thermal description file

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jun 5 12:27:35 2019

@author: crystal
"""
import os
import numpy as np
import Formulas as formula_file
from prettytable import PrettyTable
import matplotlib.pyplot as plt
from itertools import cycle
import csv
```

```
color_cycle = cycle('bgrcmk')


def thermal_describtion_glass(glass_name,temperature_list,
    ↪ formula_number,formula_name1,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9
    ↪ ):


    #display all thermal coefficients
    print("\nThermal␣coefficients␣are␣␣,")
    print("\nD0␣:␣␣" + temperature_list[1])
    print("\nD1␣:␣␣" + temperature_list[2])
    print("\nD2␣:␣␣" + temperature_list[3])
    print("\nE0␣:␣␣" + temperature_list[4])
    print("\nE1␣:␣␣" + temperature_list[5])
    print("\n\u03BB(Tk)␣␣:␣␣" + temperature_list[6])
    print("\nReference␣temperature␣T0␣:␣␣" + temperature_list[7])

    #assigning values
    D0 = float(temperature_list[1])
    D1 = float(temperature_list[2])
    D2 = float(temperature_list[3])
    E0 = float(temperature_list[4])
    E1 = float(temperature_list[5])
    Ltk = float(temperature_list[6])
    #reference temperature in celcius
    T0 = float(temperature_list[7])



    while True:
        response = input("Do␣you␣want␣to␣comparision␣with␣
            ↪ different␣wavelengths␣of␣same␣glass␣type␣for␣
            ↪ DERIVATIVE␣OF␣ABSOLUTE␣REFRACTIVE␣INDEX␣WITH␣
            ↪ TEMPERATURE(y/n)␣␣:␣␣")
        if response == 'y':

            #get input of wavelength in nanometer and temperature
                ↪ in celcius
            #T = temperature = float(input("\nPlease enter the
                ↪ temperature in celcius for thermal analysis : ")
                ↪ )
            wavelength = float(input("\nPlease␣enter␣the␣
                ↪ wavelength␣in␣(nm)␣for␣thermal␣analysis␣␣:␣␣"))
            wavelength = wavelength/1000
            #print the refractive index of particular wavelength
                ↪ at reference temperature
            n0 = formula_file.formula_thermal_name(formula_name1,
                ↪ wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)
            print("\nRefractive␣index␣at␣reference␣temperature␣is␣
                ↪ n0␣:␣␣",end= '␣')
            print(float(n0))

            #plot of derivative of absolute refractive index with
                ↪ temperature
            plot_name = str(glass_name+ '_'+formula_number)
            T = np.arange(-100.00,140.00,0.01)
            n_abs_derivative = ((((n0*n0)-1.00)/(2.00*n0))*(D0 +
                ↪ (2.00*D1*(T-T0)) + (3.00*D2*(T-T0)*(T-T0)) + ((
                ↪ E0 + (2.00*E1*(T-T0)))/((wavelength*wavelength)
                ↪ -(Ltk*Ltk))) ))
            n_abs_dcon = (n_abs_derivative)*1000000
            plt.plot(T,n_abs_dcon,'r-',label = float(wavelength)
                ↪ *1000,c=next(color_cycle))
            plt.xlabel('Surface␣Temperature',fontsize = 14)
            plt.ylabel('Derivative␣of␣n_abs(10^-6)' , fontsize =
                ↪ 14)
            plt.legend()
            plt.savefig(os.path.abspath('Plots\\'+'
                ↪ n_abs_derivative_vs_temperature'+plot_name+'.png
                ↪ '))

        else :
            break

    plt.show()

    while True:
        response = input("Do␣you␣want␣to␣get␣refractive␣index␣at␣
            ↪ specific␣temperature␣(y/n)␣:␣")
        if response == 'y' :

            #Find the refractive index at any temperature
            wavelength = float(input("Please␣enter␣the␣wavelength␣
                ↪ in␣nm␣␣:␣␣"))
            wavelength = wavelength/1000
            P = float(input("\nEnter␣the␣pressure␣(Pa)␣to␣find␣
                ↪ refractive␣index␣:␣␣"))
            T= float(input("\nEnter␣the␣temperature␣to␣find␣
                ↪ refractive␣index␣:␣"))
            #print the refractive index of particular wavelength
                ↪ at reference temperature
            n0 = formula_file.formula_thermal_name(formula_name1,
                ↪ wavelength,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9)
            print("\nRefractive␣index␣at␣reference␣temperature␣is␣
                ↪ n0␣:␣␣",end= '')
            print(float(n0))
            print("Temperature␣is␣:␣␣",end= '␣' )
            print(T0)

            #required equations
            #change of absolute refractive index with temperature
            n_abs_change = ((((n0*n0)-1.00)/(2.00*n0))*((D0*(T-T0)
                ↪ ) + (D1*(T-T0)*(T-T0)) + (D2*(T-T0)*(T-T0)*(T-T0
                ↪ )) + (((E0*(T-T0)) + (E1*(T-T0)*(T-T0)))/((
                ↪ wavelength*wavelength)-(Ltk*Ltk))) ))

            #refractive index(air) at temperature 15 and pressure
                ↪ 101330 Pa
            n_air_15 = (1.0000 + ((0.00000001)*(6432.8 +
                ↪ ((2949810*wavelength*wavelength)/((146*
                ↪ wavelength*wavelength)-1.0000)) + ((25540*
                ↪ wavelength*wavelength)/((41*wavelength*
                ↪ wavelength)-1)) )))

            #refractive index(air) at temperature T0 and pressure
                ↪ 101330 Pa
            n_air_catT0 = (1.0000 + (((n_air_15 - 1)*101325)
                ↪ /((1.0000 + 0.0034785*(T0 - 15))*101325) ))

            #refractive index(air) at temperature T and Pressure P
                ↪  Pa
            n_air_catT = (1.0000 + (((n_air_15 - 1)*P)/((1.0000 +
                ↪ 0.0034785*(T - 15))*101325) ) )

            n_rel_givenT = (n0 + (n_abs_change/n_air_catT0))*(
                ↪ n_air_catT0/n_air_catT)

            print("\nRefractive␣index␣at␣given␣temperature␣is␣:␣",
                ↪ end = '␣' )
            print(n_rel_givenT)
            print("Temperature␣is␣:␣␣",end ='␣')
            print(T)
        else :
            break
    plt.figure(figsize=(10,10))
```

```
while True:
    response = input("Do␣you␣want␣to␣get␣plot␣between␣
        ↪ refractive␣index␣and␣temperature␣(y/n)␣:␣␣")

    if response == 'y':


        wavelength = float(input("Please␣enter␣the␣wavelength␣
            ↪ in␣nm␣␣:␣␣"))
        wavelength = wavelength/1000
        P = float(input("\nEnter␣the␣pressure␣(Pa)␣to␣find␣
            ↪ refractive␣index␣:␣␣"))
        #print the refractive index of particular wavelength
            ↪ at reference temperature
        n0 = formula_file.formula_thermal_name(formula_name1,
            ↪ wavelength,\
                                                    A0,A1,A2,A3,A4,A5,
                                                        ↪ A6,A7,A8,A9)

        plot_name2 = str(glass_name+ '_'+formula_number)
        T = np.arange(-100,140,0.01)
        n_rel_givenT = ((n0 + (((((n0*n0)-1.00)/(2.00*n0))*((
            ↪ D0*(T-T0)) + (D1*(T-T0)*(T-T0)) + (D2*(T-T0)*(T-
            ↪ T0)*(T-T0)) + (((E0*(T-T0)) + (E1*(T-T0)*(T-T0))
            ↪ )/((wavelength*wavelength)-(Ltk*Ltk))) )))
            ↪ /((1.0000 + (((((1.0000 + ((0.00000001)*(6432.8
            ↪ + ((2949810*wavelength*wavelength)/((146*
            ↪ wavelength*wavelength)-1.0000)) + ((25540*
            ↪ wavelength*wavelength)/((41*wavelength*
            ↪ wavelength)-1)) )))) - 1)*101325)/((1.0000 +
            ↪ 0.0034785*(T0 - 15))*101325) )))))*(((1.0000 +
            ↪ (((((1.0000 + ((0.00000001)*(6432.8 + ((2949810*
            ↪ wavelength*wavelength)/((146*wavelength*
            ↪ wavelength)-1.0000)) + ((25540*wavelength*
            ↪ wavelength)/((41*wavelength*wavelength)-1)) ))))
            ↪  - 1)*101325)/((1.0000 + 0.0034785*(T0 - 15))
            ↪ *101325) )))/((1.0000 + (((((1.0000 +
            ↪ ((0.00000001)*(6432.8 + ((2949810*wavelength*
            ↪ wavelength)/((146*wavelength*wavelength)-1.0000)
            ↪ ) + ((25540*wavelength*wavelength)/((41*
            ↪ wavelength*wavelength)-1)) )))) - 1)*P)/((1.0000
            ↪  + 0.0034785*(T - 15))*101325) ) ))))
        plt.plot(T,n_rel_givenT,'r-',label = float(wavelength)
            ↪ *1000,c=next(color_cycle))
        plt.xlabel("Temperature",fontsize = 22)
        plt.ylabel("Refractive␣index",fontsize =22)
        plt.legend()
        plt.savefig(os.path.abspath('Plots\\'+'
            ↪ n_vs_temperature'+plot_name2+'.png'))
    else :
        break
plt.show()

#plot for relative change
while True:
    response = input("Do␣you␣want␣to␣get␣comaparison␣between␣
        ↪ different␣wavelengths␣of␣same␣glass␣type␣for␣
        ↪ DERIVATIVE␣OF␣RELATIVE␣REFRACTIVE␣INDEX␣CHANGE␣WITH
        ↪ ␣TEMPERATURE␣(y/n)␣:␣␣")
    if response == 'y':

        #get input of wavelength in nanometer
        wavelength = float(input("\nPlease␣enter␣the␣
            ↪ wavelength␣in␣(nm)␣for␣thermal␣analysis␣␣:␣␣"))
        wavelength = wavelength/1000
        #pressure
        P = float(input("\nEnter␣the␣pressure␣(Pa)␣to␣find␣
            ↪ refractive␣index␣:␣␣"))

        #print the refractive index of particular wavelength
            ↪ at reference temperature
        n0 = formula_file.formula_thermal_name(formula_name1,
            ↪ wavelength,\
                                               A0,A1,A2,A3,A4,A5,
                                                   ↪ A6,A7,A8,A9)
        print("\nRefractive␣index␣at␣reference␣temperature␣is␣
            ↪ n0␣:␣␣",end= '')
        print(float(n0))

        #derivative of refractive index of air
        #n_air_derivative = ((-0.00367)*((n_air_catT-1)
            ↪ /(1+0.00367*T)))

        #plot of derivative of absolute refractive index with
            ↪ temperature
        plot_name3 = str(glass_name+ '_'+formula_number)
        T = np.arange(-100.00,140.00,0.01)

        n_rel_derivative = (((((((n0*n0)-1.00)/(2.00*n0))*(D0
            ↪  + (2.00*D1*(T-T0)) + (3.00*D2*(T-T0)*(T-T0)) +
            ↪ ((E0 + (2.00*E1*(T-T0)))/((wavelength*wavelength
            ↪ )-(Ltk*Ltk))) ))) - ((((n0 + (((((n0*n0)-1.00)
            ↪ /(2.00*n0))*((D0*(T-T0)) + (D1*(T-T0)*(T-T0)) +
            ↪ (D2*(T-T0)*(T-T0)*(T-T0)) + (((E0*(T-T0)) + (E1
            ↪ *(T-T0)*(T-T0)))/((wavelength*wavelength)-(Ltk*
            ↪ Ltk))) )))/((1.0000 + (((((1.0000 +
            ↪ ((0.00000001)*(6432.8 + ((2949810*wavelength*
            ↪ wavelength)/((146*wavelength*wavelength)-1.0000)
            ↪ ) + ((25540*wavelength*wavelength)/((41*
            ↪ wavelength*wavelength)-1)) )))) - 1)*101325)
            ↪ /((1.0000 + 0.0034785*(T0 - 15))*101325) )))))
            ↪ *(((1.0000 + (((((1.0000 + ((0.00000001)*(6432.8
            ↪  + ((2949810*wavelength*wavelength)/((146*
            ↪ wavelength*wavelength)-1.0000)) + ((25540*
            ↪ wavelength*wavelength)/((41*wavelength*
            ↪ wavelength)-1)) )))) - 1)*101325)/((1.0000 +
            ↪ 0.0034785*(T0 - 15))*101325) )))/((1.0000 +
            ↪ (((((1.0000 + ((0.00000001)*(6432.8 + ((2949810*
            ↪ wavelength*wavelength)/((146*wavelength*
            ↪ wavelength)-1.0000)) + ((25540*wavelength*
            ↪ wavelength)/((41*wavelength*wavelength)-1)) ))))
            ↪  - 1)*P)/((1.0000 + 0.0034785*(T - 15))*101325)
            ↪ ) )))))*(((-0.00367)*((((1.0000 + (((((1.0000 +
            ↪ ((0.00000001)*(6432.8 + ((2949810*wavelength*
            ↪ wavelength)/((146*wavelength*wavelength)-1.0000)
            ↪ ) + ((25540*wavelength*wavelength)/((41*
            ↪ wavelength*wavelength)-1)) )))) - 1)*P)/((1.0000
            ↪  + 0.0034785*(T - 15))*101325) ) ))-1)
            ↪ /(1+0.00367*T))))))/((1.0000 + (((((1.0000 +
            ↪ ((0.00000001)*(6432.8 + ((2949810*wavelength*
            ↪ wavelength)/((146*wavelength*wavelength)-1.0000)
            ↪ ) + ((25540*wavelength*wavelength)/((41*
            ↪ wavelength*wavelength)-1)) )))) - 1)*P)/((1.0000
            ↪  + 0.0034785*(T - 15))*101325) ) ))))*1000000
        plt.plot(T,n_rel_derivative,'r-',label = (float(
            ↪ wavelength)*1000),c=next(color_cycle))
        plt.xlabel('Surface␣Temperature',fontsize = 14)
        plt.ylabel('Derivative␣of␣n_rel(10^-6)' , fontsize =
            ↪ 14)
        plt.legend()
        plt.savefig(os.path.abspath('Plots\\'+'
            ↪ n_rel_derivative_vs_temperature'+plot_name3+'.
            ↪ png'))

    else:
        break
plt.show()
```

```
def verification_thermal(glass_name,temperature_list,
    ↪ formula_number,formula_name1,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9
    ↪ ):

    if glass_name == 'N-BK7' or glass_name == 'F2' or glass_name
        ↪ == 'N-PK51' or glass_name == 'SF57' or glass_name == 'N
        ↪ -LAF2':

        #display all thermal coefficients
        print("\nThermal␣coefficients␣are␣␣,")
        print("\nD0␣:␣␣" + temperature_list[1])
        print("\nD1␣:␣␣" + temperature_list[2])
        print("\nD2␣:␣␣" + temperature_list[3])
        print("\nE0␣:␣␣" + temperature_list[4])
        print("\nE1␣:␣␣" + temperature_list[5])
        print("\n\u03BB(Tk)␣␣:␣␣" + temperature_list[6])
        print("\nReference␣temperature␣T0␣:␣␣" + temperature_list
            ↪ [7])

        #assigning values
        D0 = float(temperature_list[1])
        D1 = float(temperature_list[2])
        D2 = float(temperature_list[3])
        E0 = float(temperature_list[4])
        E1 = float(temperature_list[5])
        Ltk = float(temperature_list[6])
        #reference temperature in celcius
        T0 = float(temperature_list[7])


        #verification for n_abs_derivative
        while True:
            response = input("Do␣you␣want␣to␣verification␣of␣the␣
                ↪ derivative␣of␣absolute␣change␣in␣refractive␣
                ↪ index␣only␣for␣N-BK7,F2,N-LAF2,N-PK51,SF57␣␣at␣
                ↪ 1060␣nm␣wavelength␣(y/n)␣:␣␣")
            if response == 'y':

                #give the value of wavelength
                wavelength = float(1060)
                wavelength=wavelength/1000

                #open the file
                with open(os.path.abspath('Data\\
                    ↪ Verification_of_temperature_effect\\
↪ Temp_data\\'+glass_name+'\\'+glass_name+'.
↪ txt')) as file:

#make the list of data
file_content = file.read()
file_content_list = file_content.split()
index = file_content_list.index('#1')
#given data
data_list = file_content_list[(index+1) : ]
#print(data_list)

#calculated data list
data_cal_n_list =[]
temp_list=[]
for i in range(0,len(data_list),2):
    #temperature
    T=float(data_list[i])
    temp_list.append(data_list[i])

    #refractive index at reference temperature
    n0 = formula_file.formula_thermal_name(
        ↪ formula_name1,wavelength,A0,A1,A2,A3,
        ↪ A4,A5,A6,A7,A8,A9)

    #putting in formulas
    n_abs_derivative = ((((n0*n0)-1.00)/(2.00*n0
        ↪ ))*(D0 + (2.00*D1*(T-T0)) + (3.00*D2
        ↪ *(T-T0)*(T-T0)) + ((E0 + (2.00*E1*(T-
        ↪ T0)))/((wavelength*wavelength)-(Ltk*
        ↪ Ltk))) ))
    n_abs_dcon_cal = (n_abs_derivative)*1000000
    data_cal_n_list.append(n_abs_dcon_cal)

#make the list for given data
data_giv_n_list=[]
for i in range(1,len(data_list),2):
    data_giv_n_list.append(data_list[i])

#print the table for verification
print("Table␣is␣for␣glass␣type␣" + glass_name)
print("wavelength␣is␣",end='␣')
print(wavelength*1000)

with open(os.path.abspath("Plots\\Verification
    ↪ \\" + 'derivative_n_abs_change_verify.
    ↪ csv'),mode = 'w') as data:
```

```
                file = csv.writer(data)
                file.writerow(["Given␣change␣in␣reference","
                    ↪ Calculated␣by␣formula","Difference
                    ↪ (10^-6)"])
                #make the diff list
                diff_list=[]
                for i in range(0,len(data_giv_n_list),1):
                    diff = float(data_giv_n_list[i]) - float
                        ↪ (data_cal_n_list[i])
                    diff_list.append(diff)
                    table = PrettyTable()
                    table.field_names = ["Given␣change␣in␣
                        ↪ reference","Calculated␣by␣formula
                        ↪ ", "Difference(10^-6)"]
                    table.add_row([data_giv_n_list[i],
                        ↪ data_cal_n_list[i],diff])
                    print(table)

                    #save the file
                    file.writerow([data_giv_n_list[i],
                        ↪ data_cal_n_list[i],diff_list[i]])

            #plot the difference
            plt.figure(figsize=(15,10))
            for i in range(0,len(data_giv_n_list),1):

                plt.scatter((float(temp_list[i])),(float(
                    ↪ diff_list[i])),s=60)
                plt.title(glass_name)
                plt.xlabel("Temperature␣(C)",fontsize = 19)
                plt.xticks(rotation = 90)
                plt.ylabel('Difference␣of␣derivative␣of␣
                    ↪ absolute␣refractive␣index(10^-6)' ,
                    ↪ fontsize = 19)

                plt.savefig(os.path.abspath('Plots\\
                    ↪ Verification\\'+'
                    ↪ derivative_n_abs_change_plot.png'))
            plt.show()

    else:
        break

#verification for n_rel_derivative
while True:
    response = input("Do␣you␣want␣to␣verification␣of␣the␣
    ↪ derivative␣of␣relative␣change␣in␣refractive␣
    ↪ index␣only␣for␣N-BK7,F2,N-LAF2,N-PK51,SF57␣␣at␣
    ↪ 1060␣nm␣wavelength␣(y/n)␣:␣␣")
if response == 'y':

    #give the value of wavelength
    wavelength = float(1060)
    wavelength=wavelength/1000
    P=float(101325)

    #open the file
    with open(os.path.abspath('Data\\
        ↪ Verification_of_temperature_effect\\
        ↪ Temp_data\\'+glass_name+'\\'+glass_name+'_2.
        ↪ txt')) as file:

        #make the list of data
        file_content = file.read()
        file_content_list = file_content.split()
        index = file_content_list.index('#1')
        #given data
        data_list = file_content_list[(index+1) : ]
        #print(data_list)

        #calculated data list
        data_cal_n_list =[]
        temp_list=[]
        for i in range(0,len(data_list),2):
            #temperature
            T=float(data_list[i])
            temp_list.append(data_list[i])

            #refractive index at reference temperature
            n0 = formula_file.formula_thermal_name(
                ↪ formula_name1,wavelength,A0,A1,A2,A3,
                ↪ A4,A5,A6,A7,A8,A9)

            #putting in formulas
            n_rel_derivative = ((((((((n0*n0)-1.00)
                ↪ /(2.00*n0))*(D0 + (2.00*D1*(T-T0)) +
                ↪ (3.00*D2*(T-T0)*(T-T0)) + ((E0 +
                ↪ (2.00*E1*(T-T0)))/((wavelength*
                ↪ wavelength)-(Ltk*Ltk))) ))) - ((((n0
                ↪ + ((((((n0*n0)-1.00)/(2.00*n0))*((D0
                ↪ *(T-T0)) + (D1*(T-T0)*(T-T0)) + (D2*(
                ↪ T-T0)*(T-T0)*(T-T0)) + (((E0*(T-T0))
            ↪ + (E1*(T-T0)*(T-T0)))/((wavelength*
            ↪ wavelength)-(Ltk*Ltk))) )))/((1.0000
            ↪ + (((((1.0000 + ((0.00000001)*(6432.8
            ↪  + ((2949810*wavelength*wavelength)
            ↪ /((146*wavelength*wavelength)-1.0000)
            ↪ ) + ((25540*wavelength*wavelength)
            ↪ /((41*wavelength*wavelength)-1)) ))))
            ↪  - 1)*101325)/((1.0000 + 0.0034785*(
            ↪ T0 - 15))*101325) )))))*(((1.0000 +
            ↪ (((((1.0000 + ((0.00000001)*(6432.8 +
            ↪  ((2949810*wavelength*wavelength)
            ↪ /((146*wavelength*wavelength)-1.0000)
            ↪ ) + ((25540*wavelength*wavelength)
            ↪ /((41*wavelength*wavelength)-1)) ))))
            ↪  - 1)*101325)/((1.0000 + 0.0034785*(
            ↪ T0 - 15))*101325) )))/((1.0000 +
            ↪ (((((1.0000 + ((0.00000001)*(6432.8 +
            ↪  ((2949810*wavelength*wavelength)
            ↪ /((146*wavelength*wavelength)-1.0000)
            ↪ ) + ((25540*wavelength*wavelength)
            ↪ /((41*wavelength*wavelength)-1)) ))))
            ↪  - 1)*P)/((1.0000 + 0.0034785*(T -
            ↪ 15))*101325) ) )))))*(((-0.00367)
            ↪ *((((1.0000 + (((((1.0000 +
            ↪ ((0.00000001)*(6432.8 + ((2949810*
            ↪ wavelength*wavelength)/((146*
            ↪ wavelength*wavelength)-1.0000)) +
            ↪ ((25540*wavelength*wavelength)/((41*
            ↪ wavelength*wavelength)-1)) )))) - 1)*
            ↪ P)/((1.0000 + 0.0034785*(T - 15))
            ↪ *101325) ) ))-1)/(1+0.00367*T))))))
            ↪ /((1.0000 + (((((1.0000 +
            ↪ ((0.00000001)*(6432.8 + ((2949810*
            ↪ wavelength*wavelength)/((146*
            ↪ wavelength*wavelength)-1.0000)) +
            ↪ ((25540*wavelength*wavelength)/((41*
            ↪ wavelength*wavelength)-1)) )))) - 1)*
            ↪ P)/((1.0000 + 0.0034785*(T - 15))
            ↪ *101325) ) ))))
        n_rel_dcon_cal = (n_rel_derivative)*1000000
        data_cal_n_list.append(n_rel_dcon_cal)

    #make the list for given data
    data_giv_n_list=[]
    for i in range(1,len(data_list),2):
        data_giv_n_list.append(data_list[i])
```

```
#print the table for verification
print("Table␣is␣for␣glass␣type␣" + glass_name)
print("wavelength␣is␣",end='␣')
print(wavelength*1000)

with open(os.path.abspath("Plots\\Verification
    ↪ \\" + 'derivative_n_rel_change_verify.
    ↪ csv'),mode = 'w') as data:
    file = csv.writer(data)
    file.writerow(["Given␣change␣in␣reference","
        ↪ Calculated␣by␣formula","Difference
        ↪ (10^-6)"])
    #make the diff list
    diff_list=[]
    for i in range(0,len(data_giv_n_list),1):
        diff = float(data_giv_n_list[i]) - float
            ↪ (data_cal_n_list[i])
        diff_list.append(diff)
        table = PrettyTable()
        table.field_names = ["Given␣change␣in␣
            ↪ reference","Calculated␣by␣formula
            ↪ ", "Difference(10^-6)"]
        table.add_row([data_giv_n_list[i],
            ↪ data_cal_n_list[i],diff])
        print(table)

        #save the file
        file.writerow([data_giv_n_list[i],
            ↪ data_cal_n_list[i],diff_list[i]])

#plot the difference
plt.figure(figsize=(15,10))
for i in range(0,len(data_giv_n_list),1):

    plt.scatter(float(temp_list[i]),(float(
        ↪ diff_list[i])),s=60)
    plt.title(glass_name)
    plt.xlabel("Temperature␣(C)",fontsize = 19)
    plt.xticks(rotation = 90)
    plt.ylabel('Difference␣of␣derivative␣of␣
        ↪ relative␣refractive␣index␣(10^-6)' ,
        ↪ fontsize = 19)

    plt.savefig(os.path.abspath('Plots\\
        ↪ Verification\\'+'
```

```
                    ↪ derivative_n_rel_change_plot.png'))
                plt.show()

        else:
            break
```

In above file, it also includes the verification with reference's data for thermal analysis. For verification, it needs one reference data with specific folder.

Below code is for modelling of the optical system.

Listing 4: Aberration

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 1 14:51:55 2019

@author: crystal
"""

from sympy.solvers import solve
from sympy import Symbol
from sympy.functions import re
import math as m
import get_refractive_index_temperature as n_m
import matplotlib.pyplot as plt




glass_name = input("\nEnter␣the␣name␣of␣glass␣type␣:␣␣")
wavelength = float(input("\nEnter␣the␣wavelength␣in␣nm␣:␣"))
T =float(input("\nEnter␣the␣temeperature␣of␣surrounding␣in␣
    ↪ celcius␣:␣"))
P=float(input("\nEnter␣the␣pressure␣of␣surrounding␣in␣Pa␣:␣"))



#pupil define
range_pupil_max = float(input("Enter␣the␣maximum␣value␣of␣pupil␣
    ↪ in␣mm␣␣:␣"))
range_pupil_min = float(input("Enter␣the␣minimum␣value␣of␣pupil␣
    ↪ in␣mm␣␣:␣"))
equal_divide_pupil = float((range_pupil_max - range_pupil_min)
    ↪ /10)
frag_pupil_list=[]
frag_pupil_list.append(range_pupil_min)
x = range_pupil_min
```

```
for i in range(10):
    x = x + equal_divide_pupil
    frag_pupil_list.append(x)








def sag_convex(h,R,semi_diameter):
    sag = float( ((h*h)/(R + ((R*R) - (h*h))**(1/2))) )
    return sag
def sag_concave(h,R,semi_diameter):
    sag_max = sag_convex(h,R,semi_diameter)
    sag = sag_max - float( ((h*h)/(R + ((R*R) - (h*h))**(1/2))) )
    return sag








def air_vertex(angle,initial_object_dist,i):
    air_vertex.angle = angle

    n_m.thermal_describtion_glass(glass_name,wavelength,P,T)
    air_vertex.n_air = n_m.thermal_describtion_glass.n_air_catT
    air_vertex.n_medium = n_m.thermal_describtion_glass.
        ↪ n_rel_givenT

    air_vertex.h = frag_pupil_list[i]


    #height at the plane passing through vertex
    if air_vertex.angle != 0 :
        air_vertex.h = air_vertex.h + ((m.tan(air_vertex.angle*m.
            ↪ pi/180)) * (initial_object_dist ))
    if air_vertex.angle == 0:
        air_vertex.h =float(air_vertex.h)

    #distance
    OP = air_vertex.n_air*(initial_object_dist)/(m.cos(air_vertex.
        ↪ angle*m.pi/180))
    OP_r = air_vertex.n_air * (initial_object_dist)
    air_vertex.OPD = OP - OP_r
    print("\n␣Initial␣height␣is␣",end = '␣')
    print(air_vertex.h)
    print("\n␣Initial␣incident␣angle␣is␣",end='␣')
    print(air_vertex.angle)






def convex_air_medium(R,semi_diameter):
    n_m.thermal_describtion_glass(glass_name,wavelength,P,T)
    n_medium = n_m.thermal_describtion_glass.n_rel_givenT
    n_air = air_vertex.n_air

    R = R
    semi_diameter = semi_diameter

    #height at the surface
    if air_vertex.angle != 0 :
        x = Symbol('x')
        sol = solve((air_vertex.h + (m.tan(air_vertex.angle*m.pi
            ↪ /180)) * (( (x*x)/(R + ((R*R) - (x*x))**(1/2)) )) -
            ↪  x) ,x)
        air_vertex.h = float(re(sol[0]))
    if air_vertex.angle == 0:
        air_vertex.h =float(air_vertex.h)



    #distance
    OP1 = n_air*(sag_convex(air_vertex.h,R,semi_diameter))/(m.cos(
        ↪ air_vertex.angle*m.pi/180))
    OP1_r = n_air*(sag_convex(air_vertex.h,R,semi_diameter))
    OPD1 = OP1 - OP1_r



    #angles at surface
    if air_vertex.h>0:
        angle_r_1 = (m.asin( air_vertex.h/R ))*180/m.pi
        t_angle = angle_r_1 + air_vertex.angle
        air_vertex.angle = ((m.asin( (n_air*(m.sin(t_angle*m.pi
            ↪ /180))/n_medium) ))*180/m.pi - angle_r_1)
    if air_vertex.h == 0:
        angle_r_1 = (m.asin( air_vertex.h/R ))*180/m.pi
        t_angle = angle_r_1 + air_vertex.angle
        air_vertex.angle = ((m.asin( (n_air*(m.sin(t_angle*m.pi
            ↪ /180))/n_medium) ))*180/m.pi - angle_r_1)
    if air_vertex.h<0:
         angle_r_1 = float(m.asin(-air_vertex.h/R ) )*180/m.pi
         t_angle = float(angle_r_1 - air_vertex.angle)
         air_vertex.angle = (angle_r_1 - float(m.asin( (n_air*(m.
             ↪ sin(t_angle*m.pi/180))/n_medium) ))*180/m.pi )


    #distance
    sag_max = sag_convex(semi_diameter,R,semi_diameter)
    sag = sag_convex(air_vertex.h,R,semi_diameter)
    OP2 = n_medium*(sag_max - sag)/m.cos(air_vertex.angle*m.pi
        ↪ /180)
    OP2_r = n_medium*(sag_max - sag)
    OPD2 = (OP2 - OP2_r)
    convex_air_medium.OPD = OPD1+OPD2


    #Final height
    sag_max = sag_convex(semi_diameter,R,semi_diameter)
    sag = sag_convex(air_vertex.h,R,semi_diameter)
    if air_vertex.angle != 0 :
        air_vertex.h = air_vertex.h + ((m.tan(air_vertex.angle*m.
            ↪ pi/180)) * (sag_max - sag))
    if air_vertex.angle == 0:
        air_vertex.h =float(air_vertex.h)


    print("\n␣Final␣height␣at␣end␣of␣the␣lens␣is␣:␣␣",end='␣' )
    print(air_vertex.h)
    print("\n␣Angle␣at␣surface␣is␣␣:␣",end='␣')
    print(air_vertex.angle)





def convex_medium_air(R,semi_diameter):

    n_m.thermal_describtion_glass(glass_name,wavelength,P,T)
    n_medium = n_m.thermal_describtion_glass.n_rel_givenT
    n_air = air_vertex.n_air
```

```
R = R
semi_diameter = semi_diameter
#height at the surface
if air_vertex.angle != 0 :
    x = Symbol('x')
    sol = solve((air_vertex.h + (m.tan(air_vertex.angle*m.pi
        ↪ /180)) * (( (x*x)/(R + ((R*R) - (x*x))**(1/2)) )) -
        ↪  x) ,x)
    air_vertex.h = float(re(sol[0]))
if air_vertex.angle == 0:
    air_vertex.h =float(air_vertex.h)




#distance
OP1 = n_medium*(sag_convex(air_vertex.h,R,semi_diameter))/(m.
    ↪ cos(air_vertex.angle*m.pi/180))
OP1_r = n_medium*(sag_convex(air_vertex.h,R,semi_diameter))
OPD1 = OP1 - OP1_r




#angles at surface
if air_vertex.h>0:
    angle_r_1 = (m.asin( air_vertex.h/R ))*180/m.pi
    t_angle = angle_r_1 + air_vertex.angle
    air_vertex.angle = ((m.asin( (n_medium*(m.sin(t_angle*m.pi
        ↪ /180))/n_air) ))*180/m.pi - angle_r_1)
if air_vertex.h == 0:
    angle_r_1 = (m.asin( air_vertex.h/R ))*180/m.pi
    t_angle = angle_r_1 + air_vertex.angle
    air_vertex.angle = ((m.asin( (n_medium*(m.sin(t_angle*m.pi
        ↪ /180))/n_air) ))*180/m.pi - angle_r_1)
if air_vertex.h<0:
     angle_r_1 = float(m.asin(-air_vertex.h/R ) )*180/m.pi
     t_angle = float(angle_r_1 - air_vertex.angle)
     air_vertex.angle = (angle_r_1 - float(m.asin( (n_medium*(
         ↪ m.sin(t_angle*m.pi/180))/n_air) ))*180/m.pi )


#distance
sag_max = sag_convex(semi_diameter,R,semi_diameter)
sag = sag_convex(air_vertex.h,R,semi_diameter)
OP2 = n_air*(sag_max - sag)/m.cos(air_vertex.angle*m.pi/180)
OP2_r = n_air*(sag_max - sag)
OPD2 = OP2 - OP2_r
```

```
    convex_medium_air.OPD = OPD1+OPD2


    #Final height
    sag_max = sag_convex(semi_diameter,R,semi_diameter)
    sag = sag_convex(air_vertex.h,R,semi_diameter)
    if air_vertex.angle != 0 :
        air_vertex.h = air_vertex.h + ((m.tan(air_vertex.angle*m.
            ↪ pi/180)) * (sag_max - sag))
    if air_vertex.angle == 0:
        air_vertex.h =float(air_vertex.h)



    print("\n␣Final␣height␣at␣end␣of␣the␣lens␣is␣:␣␣",end='␣' )
    print(air_vertex.h)
    print("\n␣Angle␣at␣surface␣is␣␣:␣",end='␣')
    print(air_vertex.angle)






def convex_medium_medium(R,semi_diameter):

    n_m.thermal_describtion_glass(glass_name,wavelength,P,T)
    n_medium = n_m.thermal_describtion_glass.n_rel_givenT

    R = R
    semi_diameter = semi_diameter
    #height at the surface
    if air_vertex.angle != 0 :
        x = Symbol('x')
        sol = solve((air_vertex.h + (m.tan(air_vertex.angle*m.pi
            ↪ /180)) * (( (x*x)/(R + ((R*R) - (x*x))**(1/2)) )) -
            ↪  x) ,x)
        air_vertex.h = float(re(sol[0]))
    if air_vertex.angle == 0:
        air_vertex.h =float(air_vertex.h)



    #distance
    OP1 = n_medium*(sag_convex(air_vertex.h,R,semi_diameter))/(m.
        ↪ cos(air_vertex.angle*m.pi/180))
```

```
OP1_r = n_medium*(sag_convex(air_vertex.h,R,semi_diameter))
OPD1 = OP1 - OP1_r



#angles at surface
if air_vertex.h>0:
    angle_r_1 = (m.asin( air_vertex.h/R ))*180/m.pi
    t_angle = angle_r_1 + air_vertex.angle
    air_vertex.angle = ((m.asin( (n_medium*(m.sin(t_angle*m.pi
        ↪ /180))/n_medium) ))*180/m.pi - angle_r_1)
if air_vertex.h == 0:
    angle_r_1 = (m.asin( air_vertex.h/R ))*180/m.pi
    t_angle = angle_r_1 + air_vertex.angle
    air_vertex.angle = ((m.asin( (n_medium*(m.sin(t_angle*m.pi
        ↪ /180))/n_medium) ))*180/m.pi - angle_r_1)
if air_vertex.h<0:
     angle_r_1 = float(m.asin(-air_vertex.h/R ) )*180/m.pi
     t_angle = float(angle_r_1 - air_vertex.angle)
     air_vertex.angle = (angle_r_1 - float(m.asin( (n_medium*(
         ↪ m.sin(t_angle*m.pi/180))/n_medium) ))*180/m.pi )


#distance
sag_max = sag_convex(semi_diameter,R,semi_diameter)
sag = sag_convex(air_vertex.h,R,semi_diameter)
OP2 = n_medium*(sag_max - sag)/m.cos(air_vertex.angle*m.pi
    ↪ /180)
OP2_r = n_medium*(sag_max - sag)
OPD2 = OP2 - OP2_r
convex_medium_medium.OPD = OPD1+OPD2


#Final height
sag_max = sag_convex(semi_diameter,R,semi_diameter)
sag = sag_convex(air_vertex.h,R,semi_diameter)
if air_vertex.angle != 0 :
    air_vertex.h = air_vertex.h + ((m.tan(air_vertex.angle*m.
        ↪ pi/180)) * (sag_max - sag))
if air_vertex.angle == 0:
    air_vertex.h =float(air_vertex.h)



print("\n␣Final␣height␣at␣end␣of␣the␣lens␣is␣:␣␣",end='␣' )
print(air_vertex.h)
```

```
    print("\n␣Angle␣at␣surface␣is␣␣:␣",end='␣')
    print(air_vertex.angle)





def concave_air_medium(R,semi_diameter):

    n_m.thermal_describtion_glass(glass_name,wavelength,P,T)
    n_medium = n_m.thermal_describtion_glass.n_rel_givenT
    n_air = air_vertex.n_air

    R = R
    semi_diameter = semi_diameter
    sag_max = sag_convex(semi_diameter,R,semi_diameter)

    #height at the surface
    if air_vertex.angle != 0 :
        x = Symbol('x')
        sol = solve((air_vertex.h + (m.tan(air_vertex.angle*m.pi
            ↪ /180)) * (( sag_max - (x*x)/(R + ((R*R) - (x*x))
            ↪ **(1/2)) )) - x) ,x)
        air_vertex.h = float(re(sol[0]))
    if air_vertex.angle == 0:
        air_vertex.h =float(air_vertex.h)



    #distance
    OP1 = n_air*(sag_max - sag_convex(air_vertex.h,R,semi_diameter
        ↪ ))/(m.cos(air_vertex.angle*m.pi/180))
    OP1_r = n_air*(sag_max - sag_convex(air_vertex.h,R,
        ↪ semi_diameter))
    OPD1 = OP1 - OP1_r





    #angles at surface
    if air_vertex.h>0:
        angle_r_1 = (m.asin( air_vertex.h/R ))*180/m.pi
        t_angle = angle_r_1 - air_vertex.angle
        air_vertex.angle = (-(m.asin( (n_air*(m.sin(t_angle*m.pi
            ↪ /180))/n_medium) ))*180/m.pi + angle_r_1)
    if air_vertex.h == 0:
        angle_r_1 = (m.asin( air_vertex.h/R ))*180/m.pi
        t_angle = angle_r_1 - air_vertex.angle
        air_vertex.angle = (-(m.asin( (n_air*(m.sin(t_angle*m.pi
            ↪ /180))/n_medium) ))*180/m.pi + angle_r_1)
    if air_vertex.h<0:
         angle_r_1 = float(m.asin(-air_vertex.h/R ) )*180/m.pi
         t_angle = float(angle_r_1 + air_vertex.angle)
         air_vertex.angle = (-angle_r_1 + float(m.asin( (n_air*(m.
             ↪ sin(t_angle*m.pi/180))/n_medium) ))*180/m.pi )


    #distance
    sag = sag_convex(air_vertex.h,R,semi_diameter)
    OP2 = n_medium*(sag)/m.cos(air_vertex.angle*m.pi/180)
    OP2_r = n_medium*(sag)
    OPD2 = OP2 - OP2_r
    concave_air_medium.OPD = OPD1+OPD2

    #Final height
    sag = sag_convex(air_vertex.h,R,semi_diameter)
    if air_vertex.angle != 0 :
        air_vertex.h = air_vertex.h + ((m.tan(air_vertex.angle*m.
            ↪ pi/180)) * ( sag))
    if air_vertex.angle == 0:
        air_vertex.h =float(air_vertex.h)





    print("\n␣Final␣height␣at␣end␣of␣the␣lens␣is␣:␣␣",end='␣' )
    print(air_vertex.h)
    print("\n␣Angle␣at␣surface␣is␣␣:␣",end='␣')
    print(air_vertex.angle)






def concave_medium_air(R,semi_diameter):
    n_m.thermal_describtion_glass(glass_name,wavelength,P,T)
    n_medium = n_m.thermal_describtion_glass.n_rel_givenT
    n_air = air_vertex.n_air

    R = R
```

```
semi_diameter = semi_diameter
sag_max = sag_convex(semi_diameter,R,semi_diameter)

#height at the surface
if air_vertex.angle != 0 :
    x = Symbol('x')
    sol = solve((air_vertex.h + (m.tan(air_vertex.angle*m.pi
        ↪ /180)) * (( sag_max - (x*x)/(R + ((R*R) - (x*x))
        ↪ **(1/2)) )) - x) ,x)
    air_vertex.h = float(re(sol[0]))
if air_vertex.angle == 0:
    air_vertex.h =float(air_vertex.h)



#distance
OP1 = n_medium*(sag_max - sag_convex(air_vertex.h,R,
    ↪ semi_diameter))/(m.cos(air_vertex.angle*m.pi/180))
OP1_r = n_medium*(sag_max - sag_convex(air_vertex.h,R,
    ↪ semi_diameter))
OPD1 =( OP1 - OP1_r)



#angles at surface
if air_vertex.h>0:
    angle_r_1 = (m.asin( air_vertex.h/R ))*180/m.pi
    t_angle = angle_r_1 - air_vertex.angle
    air_vertex.angle = (-(m.asin( (n_medium*(m.sin(t_angle*m.
        ↪ pi/180))/n_air) ))*180/m.pi + angle_r_1)
if air_vertex.h == 0:
    angle_r_1 = (m.asin( air_vertex.h/R ))*180/m.pi
    t_angle = angle_r_1 - air_vertex.angle
    air_vertex.angle = (-(m.asin( (n_medium*(m.sin(t_angle*m.
        ↪ pi/180))/n_air) ))*180/m.pi + angle_r_1)
if air_vertex.h<0:
     angle_r_1 = float(m.asin(-air_vertex.h/R ) )*180/m.pi
     t_angle = float(angle_r_1 + air_vertex.angle)
     air_vertex.angle = (-angle_r_1 + float(m.asin( (n_medium
         ↪ *(m.sin(t_angle*m.pi/180))/n_air) ))*180/m.pi )


#distance
sag = sag_convex(air_vertex.h,R,semi_diameter)
OP2 = n_air*(sag)/m.cos(air_vertex.angle*m.pi/180)
OP2_r = n_air*(sag)
```

```
    OPD2 = OP2 - OP2_r
    concave_medium_air.OPD = OPD1+OPD2


    #Final height
    sag = sag_convex(air_vertex.h,R,semi_diameter)
    if air_vertex.angle != 0 :
        air_vertex.h = air_vertex.h + ((m.tan(air_vertex.angle*m.
            ↪ pi/180)) * ( sag))
    if air_vertex.angle == 0:
        air_vertex.h =float(air_vertex.h)



    print("\n␣Final␣height␣at␣end␣of␣the␣lens␣is␣:␣␣",end='␣' )
    print(air_vertex.h)
    print("\n␣Angle␣at␣surface␣is␣␣:␣",end='␣')
    print(air_vertex.angle)






def concave_medium_medium(R,semi_diameter):
    n_m.thermal_describtion_glass(glass_name,wavelength,P,T)
    n_medium = n_m.thermal_describtion_glass.n_rel_givenT

    R = R
    semi_diameter = semi_diameter
    sag_max = sag_convex(semi_diameter,R,semi_diameter)

    #height at the surface
    if air_vertex.angle != 0 :
        x = Symbol('x')
        sol = solve((air_vertex.h + (m.tan(air_vertex.angle*m.pi
            ↪ /180)) * (( sag_max - (x*x)/(R + ((R*R) - (x*x))
            ↪ **(1/2)) )) - x) ,x)
        air_vertex.h = float(re(sol[0]))
    if air_vertex.angle == 0:
        air_vertex.h =float(air_vertex.h)



    #distance
    OP1 = n_medium*(sag_max - sag_convex(air_vertex.h,R,
        ↪ semi_diameter))/(m.cos(air_vertex.angle*m.pi/180))
```

```
OP1_r = n_medium*(sag_max - sag_convex(air_vertex.h,R,
    ↪ semi_diameter))
OPD1 =( OP1 - OP1_r)



#angles at surface
if air_vertex.h>0:
    angle_r_1 = (m.asin( air_vertex.h/R ))*180/m.pi
    t_angle = angle_r_1 - air_vertex.angle
    air_vertex.angle = (-(m.asin( (n_medium*(m.sin(t_angle*m.
        ↪ pi/180))/n_medium) ))*180/m.pi + angle_r_1)
if air_vertex.h == 0:
    angle_r_1 = (m.asin( air_vertex.h/R ))*180/m.pi
    t_angle = angle_r_1 - air_vertex.angle
    air_vertex.angle = (-(m.asin( (n_medium*(m.sin(t_angle*m.
        ↪ pi/180))/n_medium) ))*180/m.pi + angle_r_1)
if air_vertex.h<0:
     angle_r_1 = float(m.asin(-air_vertex.h/R ) )*180/m.pi
     t_angle = float(angle_r_1 + air_vertex.angle)
     air_vertex.angle = (-angle_r_1 + float(m.asin( (n_medium
         ↪ *(m.sin(t_angle*m.pi/180))/n_medium) ))*180/m.pi )


#distance
sag = sag_convex(air_vertex.h,R,semi_diameter)
OP2 = n_medium*(sag)/m.cos(air_vertex.angle*m.pi/180)
OP2_r = n_medium*(sag)
OPD2 = OP2 - OP2_r
concave_medium_medium.OPD = OPD1+OPD2


#Final height
sag = sag_convex(air_vertex.h,R,semi_diameter)
if air_vertex.angle != 0 :
    air_vertex.h = air_vertex.h + ((m.tan(air_vertex.angle*m.
        ↪ pi/180)) * ( sag))
if air_vertex.angle == 0:
    air_vertex.h =float(air_vertex.h)



print("\n␣Final␣height␣at␣end␣of␣the␣lens␣is␣:␣␣",end='␣' )
print(air_vertex.h)
print("\n␣Angle␣at␣surface␣is␣␣:␣",end='␣')
print(air_vertex.angle)
```

```
def iblock_air(length):
    length = length
    #height = float(input("Enter the height of the block : "))

    if air_vertex.angle != 0 :
        air_vertex.h = air_vertex.h + ((m.tan(air_vertex.angle*m.
            ↪ pi/180)) * (length))
    if air_vertex.angle == 0:
        air_vertex.h =float(air_vertex.h)

    #distance
    OP = air_vertex.n_air*(length)/(m.cos(air_vertex.angle*m.pi
        ↪ /180))
    OP_r = air_vertex.n_air * (length)
    iblock_air.OPD = OP - OP_r


    print("\n␣Final␣height␣is␣:␣␣",end= '␣')
    print(air_vertex.h)





def iblock_medium(length):
    length = length
    #height = float(input("Enter the height of the block : "))

    if air_vertex.angle != 0 :
        air_vertex.h = air_vertex.h + ((m.tan(air_vertex.angle*m.
            ↪ pi/180)) * (length))
    if air_vertex.angle == 0:
        air_vertex.h =float(air_vertex.h)

    #distance
    OP = air_vertex.n_medium*(length)/(m.cos(air_vertex.angle*m.pi
        ↪ /180))
    OP_r = air_vertex.n_medium * (length)
    iblock_medium.OPD = OP - OP_r
    print("\n␣Final␣height␣is␣:␣␣",end= '␣')
    print(air_vertex.h)




def refraction_air_medium_plane():
    n_air = air_vertex.n_air
    n_medium = air_vertex.n_medium

    #snell's law
    if air_vertex.angle != 0 :
        air_vertex.angle = (m.asin(n_air*(m.sin(air_vertex.angle*m
            ↪ .pi/180))/n_medium))*180/m.pi
    if air_vertex.angle == 0 :
        air_vertex.angle = air_vertex.angle

    print("\n␣Angle␣after␣refraction␣is␣:␣␣",end='␣')
    print(air_vertex.angle)




def refraction_medium_air_plane():
    n_air = air_vertex.n_air
    n_medium = air_vertex.n_medium

    #snell's law
    if air_vertex.angle != 0 :
        air_vertex.angle = (m.asin(n_medium*(m.sin(air_vertex.
            ↪ angle*m.pi/180))/n_air))*180/m.pi
    if air_vertex.angle == 0 :
        air_vertex.angle = air_vertex.angle


    print("\n␣Angle␣after␣refraction␣is␣:␣␣",end='␣')
    print(air_vertex.angle)
```

```
if __name__ == "__main__" :

    final_h = []
    OPD_list =[]
    for i in range(len(frag_pupil_list)):

        #air_vertex(0.0,10,i)
        #OPD1 = air_vertex.OPD

        #convex_air_medium(28.7356,12.5)
        #OPD2 =convex_air_medium.OPD

        #iblock_medium(0.9911)
        #OPD3 = iblock_medium.OPD

        #concave_medium_medium(21.3675,12.5)
        #OPD4 = concave_medium_medium.OPD

        #iblock_medium(2.8839)
        #OPD5 = iblock_medium.OPD

        #concave_medium_air(82.6446,4.38)
        #OPD6 = concave_medium_air.OPD

        #iblock_air(37.0)
        #OPD7 = iblock_air.OPD



        air_vertex(0.0,10,i)
        OPD8 = air_vertex.OPD




        convex_air_medium(82.6446,4.45)
        OPD9 = convex_air_medium.OPD

        iblock_medium(2.8801)
        OPD10 = iblock_medium.OPD

        convex_medium_medium(21.3675,12.5)
        OPD11 = convex_medium_medium.OPD

        iblock_medium(0.9911)
        OPD12 = iblock_medium.OPD

        concave_medium_air(28.7356,12.5)
        OPD13 = concave_medium_air.OPD

        iblock_air(37.0)
        OPD14 = iblock_air.OPD

        final_h.append(air_vertex.h)
        OPD = (OPD8 +OPD9+ OPD10+ OPD11+ OPD12+ OPD13+ OPD14)
        OPD_list.append(OPD)






    #plots
    plt.figure(figsize=(10,10))
    plt.plot(final_h,frag_pupil_list,'r-')
    plt.xlabel("final␣height␣in␣mm")
    plt.ylabel("initial␣height␣from␣optical␣axis␣in␣mm")
    plt.show()


    plt.figure(figsize=(10,10))
    plt.plot(OPD_list,frag_pupil_list,'r-')
    plt.xlabel("OPD␣in␣mm")
    plt.ylabel("initial␣height␣from␣optical␣axis␣in␣mm")
    plt.show()
```

To run this code, we need to define each and every parts by calling its corresponding function. All parts are depicted as block of the one part. Below code is example of the model which is given in the figure 50.

Listing 5: Main source code for getting all refractive index data and thermal analysis

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jul 5 13:25:42 2019

@author: crystal
"""
```

```
import matplotlib.pyplot as plt
import Final_aberration as fa






final_h = []
OPD_list =[]




#model1 for verification with link https://ophysics.com/l14.html
for i in range(len(fa.frag_pupil_list)):
    fa.air_vertex(0.0,10,i)
    OPD1 = fa.air_vertex.OPD

    fa.convex_air_medium(10,6)
    OPD2 = fa.convex_air_medium.OPD

    fa.concave_medium_air(10,6)
    OPD3 = fa.concave_medium_air.OPD

    fa.iblock_air(9.00)
    OPD4 = fa.iblock_air.OPD


    final_h.append(fa.air_vertex.h)
    OPD = OPD1 + OPD2 +OPD3 +OPD4
    OPD_list.append(OPD)







#plots
plt.figure(figsize=(10,10))
plt.plot(final_h,fa.frag_pupil_list,'r-')
plt.xlabel("final␣height␣in␣mm",fontsize=18)
plt.ylabel("initial␣height␣from␣optical␣axis␣in␣mm",fontsize=18)
plt.savefig("final_initial.png")
plt.show()


plt.figure(figsize=(10,10))
plt.plot(OPD_list,fa.frag_pupil_list,'r-')
plt.xlabel("OPD␣in␣mm",fontsize=18)
plt.ylabel("initial␣height␣from␣optical␣axis␣in␣mm",fontsize=18)
plt.savefig("OPD_initial.png")
plt.show()
```