Université de Neuchâtel
Institut d'Informatique

Knowledge-Based Optical-Systems Design

Thèse

Présentée à la faculté des sciences pour obtenir le grade de docteur ès sciences

par

Taoufik Nouri

Sous la direction de:

Prof. P. J. Erard, Institut d'informatiques de l'université de Neuchâtel

IMPRIMATUR POUR LA THÈSE

Knowledge-Based Optical-System Design

de Monsieur Taoufik Nouri

UNIVERSITÉ DE NEUCHÂTEL

FACULTÉ DES SCIENCES

La Faculté des sciences de l'Université de Neuchâtel
sur le rapport des membres du jury,

Messieurs P.-J. Erard, R. Dändliker,
H. Buczek, R. Czichy et R. Azzam (New
Orleans)

autorise l'impression de la présente thèse.

Neuchâtel, le 4 décembre 1991

Le doyen:

A. Robert

Abstract

This work is a new approach for the design of start optical systems and represents a new contribution of artificial intelligence techniques in the optical design field.
A **K**nowledge-**B**ased **O**ptical-**S**ystems **D**esign (**KBOSD**), based on artificial intelligence algorithms, first order logic, knowledge representation, rules and heuristics on lens design, is realized.
Cognitive and metacognitive optical systems, their identifications, classifications, heuristics, and rules on optical design are also reported and commented on.

This KBOSD is equipped with optical knowledge in the domain of centered dioptrical optical systems used at low aperture and small field angles.
This KBOSD generates centered dioptrical, on-axis and low-aperture optical-systems, which are used as start systems for the subsequent optimization by existing lens design programs.

In the design of optical systems, the KBOSD takes into account many user constraints such as cost, resistance of the optical material (glass) to chemical, thermal, and mechanical effects, as well as the optical quality such as minimal aberrations and chromatic aberrations corrections.
This KBOSD generates monochromatic or polychromatic optical systems, such as singlet lens, doublet lens, triplet lens, reversed singlet lens, reversed doublet lens, reversed triplet lens and telescopes.

This KBOSD is developed in the programming language Prolog and has knowledge on optical design principles and optical properties; it is made up of more than 3000 clauses. Inference engine and interconnections in the cognitive world of optical systems are described. This KBOSD uses neither a lens library nor a lens data base, it is completely based on optical design knowledge.

This KBOSD generates optical start systems that will be optimized using lens design optimization software.
In our case we used the Sigma [Ref. 17] program to optimize and evaluate our optical systems. This KBOSD is designed to be used with any other lens design software. During the optimization, the merit function converges rapidly to zero.
Some selected optical systems **entirely designed by the KBOSD** (without any human intervention) and optimized by a lens design optimization program are reported, discussed and compared with an optical system from the

literature and with optical systems optimized from a parallel plane plates start system. The quality of the KBOSD starting systems and their influence on the final optimization results are also discussed.

According to the results obtained by this KBOSD the validity and power of this program are established.

Contents

List Of Symbols and Designations

The main symbols and designations used in this text are collected in this list [Ref. 4], [Ref. 7], [Ref. 9].

Symbol and Designations

:-	rule symbol in prolog, edinburgh syntax
A	is n*i (i angle of incidence for the marginal ray)
$A_0...A_5$	Dispersion formula constants
ABAR	is n*ibar (ibar = angle of incidence for the chief ray)
Alphal	-30°C et 70°C, in 10^{-6}/K
Ar	Alkaline resistance
Astig.	Seidel astigmatism
B	shape or bending variable for thin lenses
bfl	back focal length
C	chief (or principal ray, ray label)
CD	chief (drawn, Ray label)
Clr. Rad	clear radius of a surface
Coma	Seidel coma
Cp	Middle Specific Heat in J/g.K
Cr	Climatic resistance
CURVE	curvature of surface (=1/RADIUS)
D	distance between two surfaces, measured along an exact ray
d	axial distance between two surfaces (separation)
D(U/N)	∂ (u/n)
Dispn.	dispersion
Distort.	Seidel distortion
DOTF	diffraction MTF
E	defined by hbar = HEh
efl	equivalent focal length
F, F'	principal foci
f, f'	principal focal lengths
FCurv.	Seidel field curvature (Petzval sum = S_4/H^2)
ffl	front focal length
Fr	Stain resistance
Freq	frequency (lines/mm or cycles/radian)
G	glass variable
GOTF	geometric MTF
H	paraxial marginal ray height
h, h'	object or image height
HBAR	paraxial chief ray height
HK	Knoop hardness
I	exact angle of incidence

Inc	incrementation
Index	refractive index before a surface
K	power of a surface or system
KBOSD	knowledge based optical-system design
λ	Wavelength
l, l'	conjugate distances from first surface distance, from last surface
L, M, N	direction cosines of exact ray
Lambda	Thermal conductibility in W/m.K
LChroma.	Seidel longitudinal chromatic aberration
M	meridian (ie a ray in y-z plane, Ray label)
MAG (m)	transverse magnification
MD	meridian (drawn, Ray label)
MTF	modulation transfer function
Mue	Poisson's ratio
n	$\mathrm{sqrt}(A_0+A_1*\lambda^2+A_2*\lambda^{-2}+A_3*\lambda^{-4}+A_4*\lambda^{-6}+A_5*\lambda^{-8})$ refractive index, dispersion formula
n, n'	refractive index (object medium, image medium)
OPD	optical path difference
P	paraxial (ray label)
R	radius of a surface (=1/c)
Rad	radian
RADIUS	radius of a surface
Rho	Density
S	sagital (astigmatism)
S	skew ray (ray label)
S	spherical (surface label)
Sag	sagital
Seidel	third-order aberrations
SEPN	separation between two surfaces
SpherAb.	Seidel spherical aberration
Sr	Acid resistance
T	tangential (astigmatism)
Tan	tangential
TChroma.	Seidel transverse (lateral) chromatic aberration or lateral color
Tg	Transformation temperature in degrees Celsius
U	paraxial marginal ray angle
u	paraxial ray angle
UBAR	paraxial chief ray angle
W	wave front aberration
WL	wavelength
x, y, z	ray coordinates at a surface

1. Introduction

One of the problems in existing commercial lens design software is its inability to produce a good optical system design from an optical system initially formed by plane surfaces [Ref. 19 page 91].

Under the best circumstances of the optimization process, the merit function of these programs leads to a local minimum and remains incapable of reaching the global minimum [Ref. 20 page 27], M. Kidger reports on reasons for the existence of many local minima [Ref. 25 page 71].

The final quality of an optical system is always a function of the start optical system; thus, it is necessary to start with a potentially good optical system.

The optical start systems have been designed until the present time by optical design experts. As G. Forbes from the University of Rochester(NY) stated, "design and optimization codes usually depend on the designer to furnish a starting point" [Ref. 25].

Some existing lens design programs, called expert system, propose optical start systems by consulting lens designs from the patent literature as in Synopsys [Ref. 26] or from a lens data base as in Genesee [Ref. 27] or in Sigma [Ref. 17].

One study asserted that the trend of the research in the lens design software is oriented towards the application of artificial intelligence in proposing and calculating the start optical system [Ref. 18].

Knowledge-based systems are applied in many domains such as in robotics and autonomous systems, medicine, computer diagnostics, aeronautics etc. In this research, we applied the concepts of knowledge-based systems to optical lens design.

This work represents a completely new approach for the optical design of start systems.
The goal of this work is to study the feasibility and the validity of knowledge-based optical systems design (KBOSD) based on algorithms utilized in the artificial intelligence field.

This KBOSD produces start optical systems for subsequent optimization with existing lens design programs.

Unlike the existing expert systems this KBOSD uses neither a lens library nor a lens data base; as its name indicates it, the KBOSD is entirely based on optical design knowledge and optical heuristics. That the KBOSD is entirely

based on optical design knowledge and optical heuristics is one aspect of the originality of this research.
The combination of two different disciplines, artificial intelligence techniques and optical design knowledge base is also new and represents an important contribution to optical design.

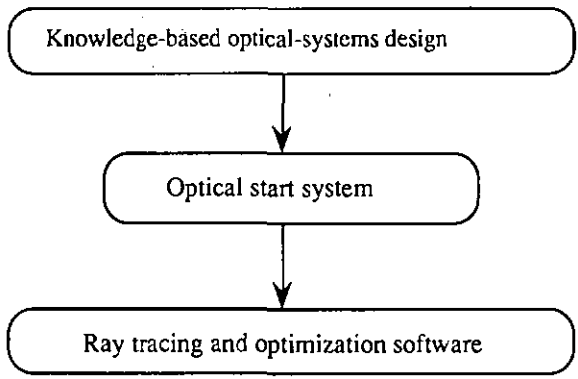


Fig. 1.1 The KBOSD generates start optical systems.

In this research we are not occupied with optimization and analysis of lens design programs and there is no competition between this KBOSD and the ray tracing programs; contrarily, KBOSD completes these programs.

This KBOSD is equipped with optical knowledge in the domain of centered dioptrical optical systems used at low aperture and small field angle.

The lens design knowledge is expressed by clauses (rules and facts) in the programming language Prolog, which is based on the first order logic [Ref. 14].
Concerning the optical glass, we use the Schott optical glass catalogue [Ref. 9].

At what point can artificial intelligence be useful in the design of the optical systems? Why do we select artificial intelligence techniques to solve optical design problems? In response to these questions, we must note that it is complicated to solve optical design problems in classical procedural programs where the solution must be written. In fact, the solution of a lens design problem resolution is based on innumerable knowledges such as heuristics, optical design rules as well as technical and numerical constraints. For example, in the selection of the optical glass, we have to take into account many constraints and we must explore many possibilities.
The modular structure of the optical element and its assembly process

typically poses a problem that can be solved by artificial intelligence algorithms. Additionally, in artificial intelligence the optical knowledge is expressed in terms of the optical user point of view. For this reason, the artificial intelligence exploration techniques are selected.

The artificial intelligence also helps us to:

- represent optical design knowledge
- express optical design knowledge in a formal method such as algebraic lens design models and physical laws
- express optical design knowledge in an informal way such as heuristics
- transfer the optical design knowledge from the optical expert to the computer in terms of clauses
- manipulate, treat and organize optical design knowledge
- solve optical design problems in a declarative or descriptive way
- search using heuristical algorithms
- backtrack in the inference engine.

Our approach to solving optical design problems is heuristic, as opposed to an algorithmic approach. An algorithm directly solves a specific problem by implementing the solution in a programming code. The resulting program solves only the specific problem it was written to solve and usually requires specific input without which it cannot proceed. A heuristic approach, on the other hand, is intended to have a much greater flexibility. Our implementation based on the heuristic approach is written so that it can find the solution to a range of problems, even when all the data is not available.

This work is realized in the programming language Prolog. Why do we select Prolog?
Using the programming language Prolog, equipped with an inference engine and the possibility of algorithmic programming, the user has only to describe the optical problem and the program will find all possible solutions. To solve an optical problem in Prolog, the desired optical system is described. Then, the inference engine explores the optical knowledge base and returns a solution. In using the backtracking principle, the inference engine explores all the possible solutions to a lens design problem.
This method of problem solving is typical of the declarative or descriptive programming languages such as Prolog. We are specially attracted by this declarative aspect that allows the resolution of optical problems in an intelligent way. For these reasons, Prolog was selected for the design and the implementation of this Knowledge based optical system design (KBOSD).

To attempt the goals mentioned above, the following procedures are defined for this KBOSD:

1) Calculate the opto-geometrical parameters of an optical system aside

from a minimum of given parameters, even if these parameters are incomplete.

2) Test the compatibility of the paraxial opto-geometrical parameters (between themselves) of the optical system and propose the adequate correction in case of an error.

3) Produce start optical systems for the ray tracing and optimization software existing on the market, see fig. 1.1.

This KBOSD is equipped with a friendly interface and is easy to use without having a profound knowledge in the optical design field. The user can enter a minimum of data on the desired optical system such as its geometrical dimensions, its application and its environment of utilization.

Subsequently, the KBOSD analyses the entered data and produces the start optical system. This start system fulfills the optimal conditions of utilization such as the proper optical glass with the optimal mechanical resistance, the optimal chemical resistance, the optimal climatic resistance, and is the most economical. The corrections of the achromatic aberrations and the heuristical configuration of the optical elements are also taken into consideration.
This KBOSD is a useful tool in assisting experts in the conception and the calculation of optical systems as well as a helpful tool for non-specialists who use optical systems.

This KBOSD is a program that performs like a human expert in a given field [Ref. 13]; in our case the field of competence is optical design, in such a way that someone who is not an expert in the aforementioned domain can utilize this expertise. An optical designer can easily update the optical knowledge base of this KBOSD and/or input his own optical knowledge.
This KBOSD, like a human expert, explains its compartments and its options. This reinforces the user's confidence in the decisions given by the KBOSD, or additionally aids the user by detecting possible faults in the reasoning of the system.

To develop a KBOSD, one must have knowledge in the following domains:

1) expertise in computer science to encode the optical knowledge, to find appropriate algorithms and to take advantage of the concept of artificial intelligence.

2) cognition in order to extract, to analyze, and to organize the optical knowledge in a coherent manner.

3) expertise in the application domain, in this case optical-systems design. The optical expert provides the optical knowledge.

In the first chapter of this thesis, we present the algorithms of artificial intelligence used in the realization of the KBOSD, its inference engine, its structure, and its mode of function.

The second chapter covers the cognitive and the metacognitive aspects of the optical systems. The cognitive aspect treats the identification, the functional and geometrical classification, the optical systems model as well as the heuristics conceming their utilization and their assembly. The metacognitive aspects treat the organization of the optical knowledge in the memory of the computer and the interconnection between the different cognitive areas. The transfer of the optical knowledge from the expert to the computer (clauses) as well as the reasoning of the computer to solve a problem of optical design are equally present.

The third chapter contains several selected optical systems entirely produced by the KBOSD and analyzed by a lens design optimization program; these optical systems are only examples.

Optical knowledge base, algorithms, optical rules and heuristics used in this work are also reported and commented on.

Many start optical systems produced by the KBOSD are reported and commented on, before the optimization. The optimization is processed according to the following facts: the thickness, the air separations and the curvature radii of the optical lenses are variables; the optical glass and the diameter are bound.
During the optimization, the merit function of the optimization converges rapidly to zero; generally, in less than 15 iterations.

The start optical systems are analyzed before and after the optimization procedures. The results are compared and commented on.

A doublet lens from the optical design literature, [Ref. 5, page 172], known by the name of Kingslake doublet, optimized and analyzed according to M. Kidger [Ref. 17, page 1.3] is displayed.
The doublet lens from the optical design literature is reported and analyzed in order to allow some comparisons between this doublet and a second one produced in the same conditions by the KBOSD. According to the analysis of the doublet lens from the literature, one can see the signification of an optical lens of "good quality".

Chapter five contains comparisons and discussions of optical systems optimized from parallel plane plates start systems with optical systems

generated by the KBOSD. One demonstrates the incapacity of the lens design optimization software to produce a "good quality" optical systems starting from parallel plane plates. However, it is possible to achieve good quality optical systems by starting from parallel plane plates and mastering the optimization programs and optical system design.

In chapter six one treats the optical quality of the KBOSD starting systems and their influence on the final optimization results and one discusses "why the KBOSD starting systems lead to a good quality optical systems after optimization". One demonstrates that the KBOSD predesigned starting optical systems have a low third order aberrations. The third order aberrations of the KBOSD start systems don't change significantly during the optimization.

In the domain of centered dioptrical optical systems used at low aperture, on axis and low field angle, such as singlet lens, doublet lens, triplet lens, reversed singlet lens, reversed doublet lens and reversed triplet lens the KBOSD generates a "good quality" optical start systems.

The KBOSD provides the results in the text file readable by the ray tracing and optimization programs.

The optical systems produced by the KBOSD are representative of the optical design field in the usual applications. The most complex optical systems can always be broken up into simple optical systems.

2 Artificial Intelligence and Optical Systems Design

2.0 Introduction

2.I Presentation of the Prolog Programming Language

2.1.1 Structure of a Program in Prolog
2.1.2 Fundamental Notions of the Prolog Programming Language
2.1.2.1 Prolog Facts
2.1.2.2 Prolog Rules
2.1.2.3 Conjunction in Prolog
2.1.2.4 Disjunction in Prolog
2.1.3 The Role of the Inference Engine

2.2 Structure of the KBOSD

2.2.1 Pyramidal Structure of the KBOSD
2.2.2 Multi-layered Structure of the KBOSD

2.3 Algorithms Used in Search of a Solution within the KBOSD

2.3.1 General Organization of the KBOSD
2.3.2 Algorithms of Calculation of Parameters of the Optical Systems
2.3.3 Algorithms of Test of Parameters of the Optical Systems
2.3.4 Algorithms of Determination of the Context of Use of the Optical System
2.3.5 Searching Algorithms of Focal Optical System
2.3.6 Searching Algorithms of Afocal Optical System
2.3.7 Searching Algorithms for Selection of an Optical Glass

2.4 Conclusion

2.0 Introduction

Artificial intelligence is the study of concepts that allow the machines intelligence [Ref. 6].

But what is intelligence? Is it the faculty of reasoning, that of learning and using knowledge or that of perceiving and of manipulating objects from the real world? Intelligence is composed of all of these facets in addition to others. Any exhaustive definition of intelligence does not seem possible. The intelligence is an amalgam of innumerable talents that permits us to represent information and to treat it.

At what point can artificial intelligence be useful in the design of optical systems? Why do we select artificial intelligence techniques to solve optical design problems?

In response to these questions, we must note that it is complicated to solve optical design problems in classical procedural programs where the solution must be written.

In fact, the solutionlen of the design problem is based on innumerable knowledges such as heuristics, optical design rules, as well as technical and numerical constraints. For example, in the selection of optical glass, we have to take into account many constraints and we must explore many possibilities. The modular structure of the optical element and its assembly process typically pose a problem that can be solved by artificial intelligence algorithms. Additionally, in artificial intelligence the optical knowledge is expressed in terms of the optical user point of view. For this reason, artificial intelligence exploration techniques are adopted.

Using the programming language Prolog, equipped with an inference engine and the possibility of algorithmic programming, the user has only to describe the optical problem and the program will find all possible solutions. The Prolog programming language is oriented toward the implementation of programs based on artificial intelligence principles and the transfer of knowledge from the expert to the computer. A knowledge base is implemented in Prolog in terms of clauses. In our case this knowledge base contains optical knowledge, rules, laws of lens design, heuristics, and constraints of lens design as well as a base knowledge of optical glass.

To solve an optical problem, the desired optical system is described. Then the inference engine explores the optical knowledge base and returns a solution. In using the backtracking principle, the inference engine explores all the possible solutions to a lens design problem. This method of problem solving is typical of the declarative or descriptive programming languages

such as Prolog. We are specially attracted by this declarative aspect that allows the solution of optical problems in an intelligent way. For these reasons, Prolog was selected for the design and the implementation of this Knowledge based optical system design (KBOSD).
Notions of a descriptive programming language and artificial intelligence such as the capacity of reasoning, organizing and using the knowledge as well as manipulating objects will be treated in this chapter as follows:

1. Fundamental Notions of the Prolog Programming Language
2. Expressions of Optical Knowledge in Prolog
3. Pyramidal and Multi-layered Structure of the KBOSD
4. Depth Search Strategy for Optical Design Problem Solving

These ideas will be illustrated by some examples in the optical design domain.

2.1 Presentation of the Prolog Programming Language

The name Prolog is a neologism for *Pro*gramming in *log*ic. Prolog is a **descriptive or relational** language. In a descriptive language problems are approached in an entirely different way than in classical procedural languages such as Fortran or Pascal. The procedures are not programmed, they are carefully described and Prolog uses its inference engine to solve problems. Prolog is based on predicate calculus, a powerful symbolism that can be used to represent the logical relations between objects and their attributes. Logic programming is a descriptive approach to knowledge representation and problem solving. Instead of telling the computer what to do in every situation, the logic programming describes the essential features of a problem and the result as a goal that is to be accomplished, without giving the computer a step-by-step list of what instructions it is to execute. Logic programming emphasizes the stable and structural aspects of knowledge: objects, events, facts and relationships. This approach is appropriate when the results that are needed can be placed in a form that allows the outcome to be determined through a process of deductive inference. Prolog is made to represent and utilize knowledge that exists in a certain domain. More precisely the domain is a set of objects and knowledge is materialized by a set of relations that describe the properties of these objects and their interactions. In our case, the domain in question is the design of optical systems [Ref. 15].

2.1.1 Structure of a Program in Prolog

A program in Prolog is made up of the following elements: facts, rules and commentaries. Facts describe particular proprieties of the elements and their

relations; they are the fundamentals element of the knowledge base. Rules describe general properties of the elements in the data base. In most cases, they contain variables that will be instantiated with particular constants or terms from the data base. Each rule is formed by a left member (or the head of the rule) and a right member (or body of the rule) linked by the connector ":-" and ending with the character ".". The head of the rule is reduced to a single predicate, and, the right member of the rule is a succession of predicates, possibly none. A fact is a rule without a right member. The rules serve to define the relations that make up the program. A set of rules and facts describing objects and their relationships constitute a Prolog program.

Execution of a program in Prolog consists of responding to a question concerning these relations. From the syntactical point of view, a question is a serie of goals, that represent a conjunction of relations to be satisfied. The answers to these questions are the constraints, carrying the apparent variables in a series of goals and satisfying the relations considered. For example, it is said that the "the basf2 glass has a refractive index 1.64498 at the wavelength 1.064 μm" is nothing other than an affirmation that a relation refractive_index links two objects, designated by their names: *basf2* and *refractive_index_1.064*, and that is written as:

%refractive_index(basf2, Refractive_index_1064).
refractive_index(basf2, 1.64498).

A querry such as, "which glass possesses the refractive index 1.64498 at the wavelength of 1.064 μm?" is stated in Prolog syntax as: *refractive_index(X, 1.64498).* Prolog answers: *X = basf2*. This is the same as looking for a relation refractive_index_1.064 that links variable X = basf2 to another object(1.64498).

If one defines a relationship between two objects, the order in which the objects are given is important. To express the preceding relation, identifiers are used to name the used objects and the relation that links them. The name of the relation refractive index is called "predicate," the objects which carry the relation are the "arguments."
The symbol "%" signifies a comment in the Prolog syntax; this symbol is used in this case to explain the significance and the content of the rules that it follows.

2.1.2 Fundamental Notions of the Prolog Programming Language

Optical knowledge is expressed in the Prolog programming language by clauses.

Prolog can store two basic types of information in its deductive data base: facts and rules; these are called clauses.

2.1.2.1 Prolog Facts

The simplest kind of statement is called a *fact*. A fact is a means of stating that a relationship holds between objects. A finite set of facts constitutes a program. This is the simplest form of a logic program. A set of facts is also a description of a situation. A fact is merely a special case of a rule. Facts are also called unit clauses.
A fact is the simplest way to satisfy a goal in Prolog. In such a way, the arguments of the predicate of the goal are unified with those of the selected fact.

Example 2.1.2,1-1

The following example, part of the knowledge base, furnishes the dispersion coefficients of optical glass. These dispersion coefficients are useful for calculating the refractive index of the glass. This means that the Glass_Name1 possesses the following dispersion coefficients: A0, A1, A2, A3, A4, A5. It is a fact in Prolog [Ref. 25].

% coefficient_dispersion(Glass_Name1, A0, A1, A2, A3, A4, A5).

coefficient_dispersion(bafn10,2.7293062,-1.0356456e-2,2.0236563e-2, 5.8969718e-4,-2.0288303e-5, 2.8521978e-6).

coefficient_dispersion(bk1,2.2513742,-9.3254015e-3,1.0539647e-2, 2.2595365e-4, -1.0729053e-5, 7.2832778e-7).

coefficient_dispersion(bk10,2.2177191,-1.0248661e-2,9.6627662e-3, 1.6782840e-4,-5.5328684e-6, 3.4747416e-7).

The facts above provide the refractive indices of three commom glasses as a function of wavelengths.

Example 2.1.2.1-2

The following facts express the refractive indices of some optical glasses at specific wavelengths.

% refractive_index(Glass_Name1,N2325, N1970, N1529, N1060, Nt,Ns, Nr).

refractive_index(bafn10, 1.63587, 1.64136, 1.64737, 1.65413, 1.65496, 1.65853, 1.66341).

refractive_index(bk1, 1.48383, 1.48913, 1.49468, 1.50015, 1.50075, 1.50317, 1.50621).

refractive_index(bk10, _, _, _, 1.48827, 1.48887, 1.49127, 1.49419).

This means that the optical glass mentioned by the variable Glass_Name1 has the refractive indices N2325, N1970, N1529, N1060, Nt, Ns, Nr at the following wavelengths[μm] 2325, 1970, 1529, 1060, 1014, 852.1, 707.

2.1.2.2 Prolog Rules

A rule enables us to define new relationships in terms of existing relationships. The rule base in a KBOSD is the result of the encoding of the expertise. It may take several rules to express the things that a human expert knows "instinctively." Even a simple KBOSD will contain many rules.

Example 2.1.2.2-1

Rules are statements of the form:

```
use_planoconvex(Optical_Power, Transversal_Magnification, l, Diameter) :-
        convergent(Optical_Power),
        low_aperture(Optical_Power, Diameter),
        negative_Transversal_Magnification(Transversal_Magnification),
        near_infinite_ratio(Transversal_Magnification),
        low_power_lens(Optical_Power),
        object_at_infinity(Optical_Power, l),
        outm ("plano-convex conditions --> "),nl.
```

The left hand side member:

use_planoconvex(Optical_Power, Transversal_Magnification, l, Diameter) is the predicate or the head of the rule, while the names which are enclosed within the parenthesis are called arguments. The right-hand side member is called the body of the rule.

To solve a problem in Prolog, one has to invoke a goal. The goal will be compared with the left-hand side member of a rule. In order to be satisfied, the left hand side member of a rule has to satisfy the right-side member. Also the rules are used to satisfy goals. Note that the right-hand side member of a rule is a set of sub-goals that must also be achieved.

Objects such as Optical_Power, Transversal_Magnification, I, Diameter in the predicate use_plano are called variables of the predicate. Objects of the right side such as *low_aperture(Optical_Power, Diameter), negative_Transversal_Magnification(Transversal_Magnification)...* can also be made up of rules or of facts.

Example 2.1.2.2-2

The following rule defines the optical glass of the crown type.

crown(Nd, Nued) :-
Nd < 1.6, Nued > 51.0.

Example 2.1.2.2-3

The following rule defines flint optical glass:

flint(Nd, Nued) :-
Nd >1.6, Nued < 30.0.

The Nd and Nued are the refractive index at the mercury d line and the Abbe number at the same wavelength.

Example 2.1.2.2-4

The following rule permits the selection of an optical glass of hardness Hr chosen by the user.

knoop_hardness(Glass_Name, Hr) :-
glass_hardness(Hardness),
Hr >= Hardness .

The clause glass_hardness(Hardness) contains the hardness coefficient of all the optical glasses available in the knowledge base.

Example 2.1.2.2-5

The following clause permits the selection of an optical glass described by its economical, thermal, chemical, and mechanical properties.

lens_index([Glass1_Name, Nd]) :-
glass_reference(Glass_Name1, Glass_Name2, Nd, Nued, Nf_Nc, Ne1, Nuee1, Nfprim_moins_Ncprim),
economical_glass(Glass_Name1),
thermal_properties(Glass_Name1, Alpha1, Alpha2, Tg, T, Cp, Thermal_Conductivity),

chemical_properties(Glass_Name1, B, Cr, Fr, Sr, Ar),
mechanical_properties(Glass_Name1, Rho, Emo, mue, Hk).

2.1.2.3 Conjunction in Prolog

Here is a Prolog conjunction:
P :- Q,.R.
Where P, Q and R are terms.
This clause is read: P is true if Q and R are true, Q and R imply P [Ref. 7].
Here are some examples of our KBOSD using the conjunction notion.

Example 2.1.2.3-1

```
low_aperture(Optical_Power, Diameter) :-
        wavelength([Lambda1]),
        entrance_pupil_diameter(Aperture),
        power_focal(Focal_Length, Optical_Power),
        abs(Focal_Length/Aperture) > 4.0,
        diameter(Diameter),
        outm(" low aperture "),nl.
```

This rule is interpreted so that the rule low-aperture(Optical_Power, Diameter) is true, only when the rule wavelength([Lambda1]) and the rule entrance_pupil_diameter(Aperture) and the rule power_focal(Focal_Length, Optical_Power) and the rule abs(Focal_Length/Aperture) > 4.0 and the rule diameter(Diameter) are true. In this way the conjunction in Prolog is expressed [Ref. 25].

Example 2.1.2.3-2

The following rule describes the chemical properties of the optical glass.

```
chemical_properties(Glass_Name; B, Cr, Fr, Sr, Ar) :-
        physical_properties2(Glass_Name, Rho, Emo, mue, Hk, B, Cr, Fr, Sr,
        Ar),
        presence_of_bubbles(Glass_Name, B),
        climatic_resistance(Glass_Name, Cr),
        stain_resistance(Glass_Name, Fr),
        acid_resistance(Glass_Name, Sr),
        alkaline_resistance(Glass_Name, Ar).
```

Example 2.1.2.3-3

The following rule describes the mechanical properties of an optical glass

```
mechanical_properties(Glass_Name. Rho, Emo, mue, Hk) :-
```

physical_properties2(Glass_Name, Rho, Emo, mue, Hk, B, Cr, Fr, Sr, Ar),
density(Glass_Name, Rho),
elasticity_coefficient(Glass_Name, Emo),
poisson_coefficient(Glass_Name, Mue),
knoop_hardness(Glass_Name, Hk).

2.1.2.4 Disjunction in Prolog

Here is a Prolog disjunction:
P :- Q; R.
Where P, Q and R are terms.
This clause is read: P is true if Q is true or R is true; or Q or R imply P [7].
This disjunction can also be written as:
P :- Q.
P :- R.
This is another manner of expressing the disjunction in Prolog.
Here are some examples of our KBOSD using the disjunction notion.

Example 2.1.2.4-1

Stating a disjunction in Prolog repeats the heading of the rule but with a different body. That is to say that the indented text is different in each one [Ref. 25].
Thus the two following rules are disjunctions.

```
low_aperture(Optical_Power, Diameter) :-
        wavelength([Lambda1]),
        entrance_pupil_diameter(Aperture),
        power_focal(Focal_Length, Optical_Power),
        abs(Focal_Length/Aperture) > 4.0,
        diameter(Diameter),
        outm(" low aperture ").

low_aperture(Optical_Power, Diameter) :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        entrance_pupil_diameter(Aperture),
        power_focal(Focal_Length, Optical_Power),
        abs(Focal_Length/Aperture) > 2.5,
        diameter(Diameter),
        outm(" low aperture ").
```

2.1.3 The Role of the Inference Engine

An inference engine is a program that can draw conclusions based on data. Using strategies borrowed from formal logic, rule analysis and search patterns, an inference engine provides reasoning power. The inference engine finds its way through a maze of possible paths to arrive at the solution. An inference engine is designed to solve a problem by applying the expertise that is coded into the system to the data of the optical problem. The inference engine selects and follows the pertinent rules, based on the data.

Prolog is a backward-chaining inference engine. It is a search strategy that starts with what you want to prove and tries to find out if you can obtain the desired goal from the given facts. To respond to a question, i.e. to satisfy a conjunction of relations represented by a series of predicates, Prolog will attempt to satisfy the goals, one by one, in the order in which they are presented. The inference engine permits interpretation of the questions and rules as follows:

-A question such as:

```
indice(Ni, No),
known_Variable([l', z']),
relation(re2,Optical_Power,No/(l'-z'),"No/(l'-z')","Optical_Power"),!,
calcul(Optical_Power, Transversal_Magnification, l, l', z, z').
```

is interpreted by the order: satisfy first the goal *indice(Ni,No)*, then satisfy *known_Variable([l', z'])*, then satisfy *relation(re2, Optical_Power, No/(l'-z'),"No/(l'- z')","Optical_Power"), !,* then satisfy *calculation(Optical_Power, Transver-sal_Magnification, l, l', z, z').*
Note in this example that the presence of "! ", called **cut**, controls the backtracking [Ref. 6], [Ref. 25].

- A rule such as:

```
calcul(Focal_Length, Transversal_Magnification, l, l', z, z') :-
    power_focal(Focal_Length, Optical_Power),
    startcalcul(Optical_Power, Transversal_Magnification, l, l', z, z'),
    transversal_Magnification( Transversal_Magnification),
    focal(Optical_Power),nl,
    test(Optical_Power, Transversal_Magnification, l, l', z, z'),!,
    lens(Optical_Power, Transversal_Magnification, l, l').
```

is interpreted as: To satisfy the goal *calcul(Focal_Length, Transversal_Magnification, l, l', z, z')*, satisfying first the goal *power_focal (Focal_Length,Optical_Power)*, then satisfying the goal *startcalcul*

(Optical_Power, Transversal_Magnification, l, l', z, z'), then satisfying the goal*Transversal_Magnification(Transversal_Magnification)*, then satisfying the goal *focal(Optical_Power)*, satisfying the goal *test(Optical_Power, Transversal_Magnification, l, l', z, z')*, and finally satisfying the goal *lens (Optical_Power, Transversal_Magnification, l, l')*.

A rule such as *dispersion_constants(Glass_Name, A0, A1, A2, A3, A4, A5) :-*, is interpreted as the fact *dispersion_constants(Glass_Name, A0, A1, A2, A3, A4, A5)* and will always be satisfied. The execution of a Prolog program is not deterministic i.e. the series of goals will be satisfied by all possible ways.
The preceding example allows us to demonstrate some points characteristic of Prolog. Additional structures of representation of the information and searching strategies are utilized in the realization of this KBOSD [Ref. 7], [Ref. 25].

2.2 Structure of the KBOSD

Among many notions, concerning the structure of the KBOSD, two particularly interesting ones merit discussion: they are the pyramidal structure and the multi-layered structure.

2.2.1 Pyramidal Structure of the KBOSD

In a pyramidal structure, the principal or key information is located at the top of the pyramid. The secondary information necessary for this key is situated at the base of the pyramid. This secondary information can have a pyramidal structure too. The principal and secondary information mentioned above are, in fact, rules. Such a representation of information offers great clarity for the comprehension and the management of the knowledge base.The rules at the base of the pyramid are held at the disposal of their masters at the top of the pyramid. This pyramidal structure is utilized to represent the optical knowledge in this KBOSD. All the examples reported in this section explain the pyramidal structure of the KBOSD.

Example 2.2.1-1

The following example describes the conditions utilized in a plano-convex optical system.

```
use_planoconvex(Optical_Power, Transversal_Magnification, l,
    Diameter) :-
    convergent(Optical_Power),
    low_aperture(Optical_Power, Diameter),
```

negative_Transversal_Magnification(Transversal_Magnification),
near_infinite_ratio(Transversal_Magnification),
low_power_lens(Optical_Power),
object_at_infinite(Optical_Power, l),
outm ("plano-convex conditions --> "),nl.

The rules *convergent(Optical_Power), low_aperture(Optical_Power, Diameter),negative_Transversal_Magnification(Transversal_Magnification), near_infinite_ratio(Transversal_Magnification), low_power_lens(Optical_Power), object_at_infinite(Optical_Power,l)* and *outm(" plano-convex conditions --> ")* forming the right part are equally rules that must be known by the Knowledge base; example 2.2.1-3 below explains the significance of the rule *low_aperture(Optical_Power, Diameter).*

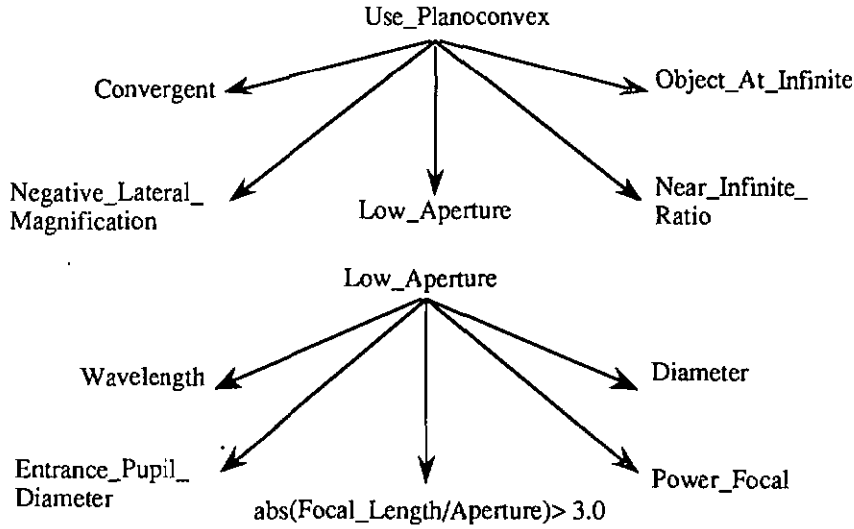


Fig. 2.2.1 Pyramidal Structure of the KBOSD

Example 2.2.1-2.

The following example describes the conditions of use of a biconvex optical system.

use_biconvex(Optical_Power, Transversal_Magnification, l, Diameter) *:-*
convergent(Optical_Power),
low_aperture(Optical_Power, Diameter),
negative_Transversal_Magnification(Transversal_Magnification),

high_optical_power(Optical_Power),
finite_ratio(Transversal_Magnification),
outm("biconvex conditions --> "),nl.

Example 2.2.1-3

This example explains the significance of the *low-aperture(Optical_Power, Diameter)* rule stated in the above example.

If an optical system is used at two wavelengths *wavelength([Lambda1, Lambda2])* , it is at low aperture if the ratio of its focal length and its diameter are larger than 3, *abs(Focal_Length/Aperture) > 3.0* ; that is a heuristic.

```
low_aperture(Optical_Power, Diameter) :-
        wavelength([Lambda1, Lambda2]),
        entrance_pupil_diametereter(Aperture),
        power_focal(Focal_Length, Optical_Power),
        abs(Focal_Length/Aperture) > 3.0,
        diameter(Diameter),
        outm (" low aperture "),
        nl.
```

Contrarily, we consider that an optical system used at three different wavelengths *wavelength([Lambda1, Lambda2,Lambda3])*, is at low aperture if the ratio between its focal length and its diameter is greater than 2.5, *abs(Focal_Length/Aperture) > 2.5*. This is also a heuristic.

```
low_aperture(Optical_Power, Diameter) :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        entrance_pupil_diameter(Aperture),
        power_focal(Focal_Length, Optical_Power),
        abs(Focal_Length/Aperture) > 2.5,
        diameter(Diameter),
        outm (" low aperture "),
        nl.
```

2.2.2 Multi-layered Structure of the KBOSD

A multi-layered structure is a structure of knowledge representation. The knowledge rules are represented at different hierarchical levels according to their importance and according to their functions. This multi-layered structure produces a reasoning structure and an abstract structure which are also multi-layered. Each layer is formed by many rules. The complexity of the rules is proportional to the level where they are situated in the hierarchy.

The rules located at higher levels are more important than the rules found at lower levels. Such a multi-layered structure offers a great advantage for the clarity of the knowledge representation, for the reasoning facility, as well as for the speed in order to search a solution.

The communication between these hierarchical levels is established according to connections between them. Each level executes certain tasks and delegates the rest of the work to the lower levels. For example, if one calculates a lens used at two wavelengths, it is important to ask first if it will be utilized at low or high aperture. This is the first level of reasoning. Subsequently, one is occupied by the selection of the optical glass according to the achromatic conditions and the conditions of use desired by the user; this is the second level of reasoning. Afterwards, one is occupied by the lenses forming the doublet, if the lenses are plano-convex, biconvex.This is the third level of reasoning. The final level of reasoning in this example is occupied with the calculation of parameters of each lens such as its thickness, its curvature radius etc. All the examples reported in this section explain the multi-layered structure of the KBOSD.

Example 2.2.2-1

The following rule *lens(Optical_Power, Transversal_Magnificatian, l, l')* explains to the KBOSD when it is necessary to use a plano-convex optical system formed by a simple lens *planoconvex_lens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index, Diameter).*

```
lens(Optical_Power, Transversal_Magnificatian, l, l') :-
        wavelength([Lambda1]),
        use_planoconvex(Optical_Power, Transversal_Magnificatian, l,
        Diameter),
        lens_index([Glass_Name, Refractive_Index]),
        planocanvex_lens(Optical_Power, Thickness, Radius1, Radius2,
        Refractive_Index, Diameter),nl,
        output_lens(Optical_Power, Thickness, Radius1, Radius2,
        Refractive_Index, Glass_Name, Diameter, Lambda1),
        stap_surface(Position, entrance_diameter).
```

This rule is written in a multi-layered structure, see Fig. 2.2.2.

Example 2.2.2-2

The following rule *lens(Optical_Power, Transversal_Magnification, l, l')* explains to the KBOSD when it is necessary to utilize a plano-convex optical system formed by a doublet *planoconvex_doublet(Optical_Pawer, Thickness1, Thickness2, Radius1, Radius2, Radius3, Radius4,*

Refractive_Index1, Refractive_Index2, Glass_Name1, Glass_Name2, Diameter, Lambda1, Lambda2, Separation).

```
lens(Optical_Power, Transversal_Magnification, l, l') :-
        wavelength([Lambda1, Lambda2]),
        use_planoconvex(Optical_Power, Transversal_Magnification, l,
        Diameter),
        planoconvex_doublet(Optical_Power, Thickness1, Thickness2, Radius1,
        Radius2, Radius3, Radius4, Refractive_Index1, Refractive_Index2,
        Glass_Name1, Glass_Name2, Diameter, Lambda1, Lambda2,
        Separation),
        nl,
        output_doublet_lens(Optical_Power, Thickness1, Thickness2, Radius1,
        Radius2, Radius3, Radius4, Refractive_Index1, Refractive_Index2,
        Glass_Name1, Glass_Name2, Diameter, Lambda1, Lambda2,
        Separation),
        stop_surface(Position, entrance_diameter).
```

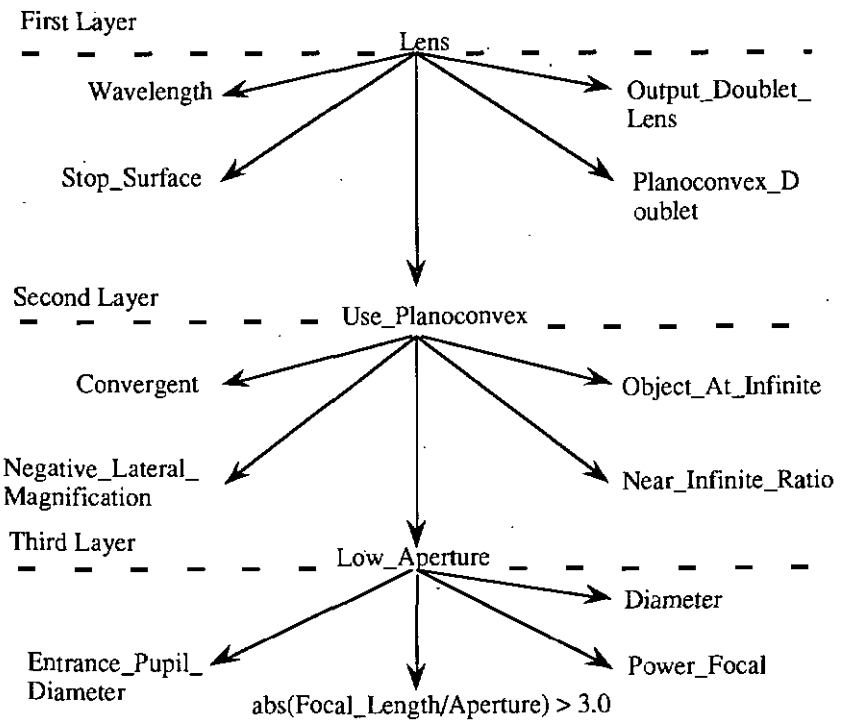


Fig. 2.2.2 Multi-layered Structure of the KBOSD

Example 2.2.2-3

The following rule *lens(Optical_Power, Transversal_Magnification, l, l')* instructs the KBOSD when to utilize a biconvex optical system formed by a triplet *biconvex_triplet(...)*.

```
lens(Optical_Power, Transversal_Magnification, l, l') :-
    wavelength([Lambda1, Lambda2, Lambda3]),
    use_biconvex(Optical_Power, Transversal_Magnificatian, l,
    Diameter),
    biconvex_triplet(Optical_Power, Thickness1, Thickness2, Thickness3,
    Radius1, Radius2, Radius3, Radius4, Radius5, Radius6,
    Refractive_Index1, Refractive_Index2, Refractive_Index3,
    Glass_Name1, Glass_Name2, Glass_Name3, Diameter, Lambda1,
    Lambda2, Lambda3, Separation1, Separation2),nl,
    output_tripletlens(Optical_Power, Thickness1, Thickness2, Thickness3,
    Radius1, Radius2, Radius3, Radius4, Radius5, Radius6,
    Refractive_Index1, Refractive_Index2, Refractive_Index3,
    Glass_Name1, Glass_Name2, Glass_Name3, Diameter, Lambda1,
    Lambda2, Lambda3, Separation1, Separation2),
    stop_surface(Position, entrance_diameter).
```

Example 2.2.2-4

The two following clauses permit the assembly of a doublet type optical system at low and high temperatures.

```
assemblingdoublet( Optical_Power, Power1, Power2, Refractive_Index1,Ref-
    ractive_Index2, Glass_Name1,Glass_Name2, Diameter, Separation) :-
    low_temperature(T),
    outm("assembling process doublet "),nl,
    lens_index2([Glass1_Name, Glass_Name2, Refractive_Index1,
    Refractive_Index2, V1, V2]),
    Separation is 0.0,
    Power1 is V1*Optical_Power/(V1-V2),
    Power2 is V2*Optical_Power/(V2-V1),
    abs((Power1/V1) + (Power2/V2)) < 1.0e-10,
    abs(Power1 + Power2 - Optical_Power) < 1.0e-10.

assemblingdoublet( Optical_Power, Power1, Power2, Refractive_Index1,
    Refractive_Index2, Glass_Name1, Glass_Name2, Diameter, Se-
    paration) :-
    high_temperature(T),
    outm("assembling process doublet "),
    nl,
    Separation is 0.4,
```

lens_index2([Glass1_Name, Glass_Name2, Refractive_Index1, Refractive_Index2, V1, V2]),
*Power1 is V1*Optical_Power/(V1-V2),*
*Power2 is V2*Optical_Power/(V2-V1),*
abs((Power1/V1) + (Power2/V2)) < 1.0e-10,
abs(Power1 + Power2 - Optical_Power) < 1.0e-10.

2.3 Algorithms Used in Search of a Solution within the KBOSD

2.3.1 General Organization of the KBOSD

The inference engine searches for a solution for the posed optical problems according to the diagram in Fig. 2.3.1.
The double-headed arrows(according to figure 2.3.1) signify that one can backtrack in case of a failure. The single-headed arrows signify that backtracking is not possible. Each of the bubbles in the diagram will be treated explicitly in the following section of this chapter.

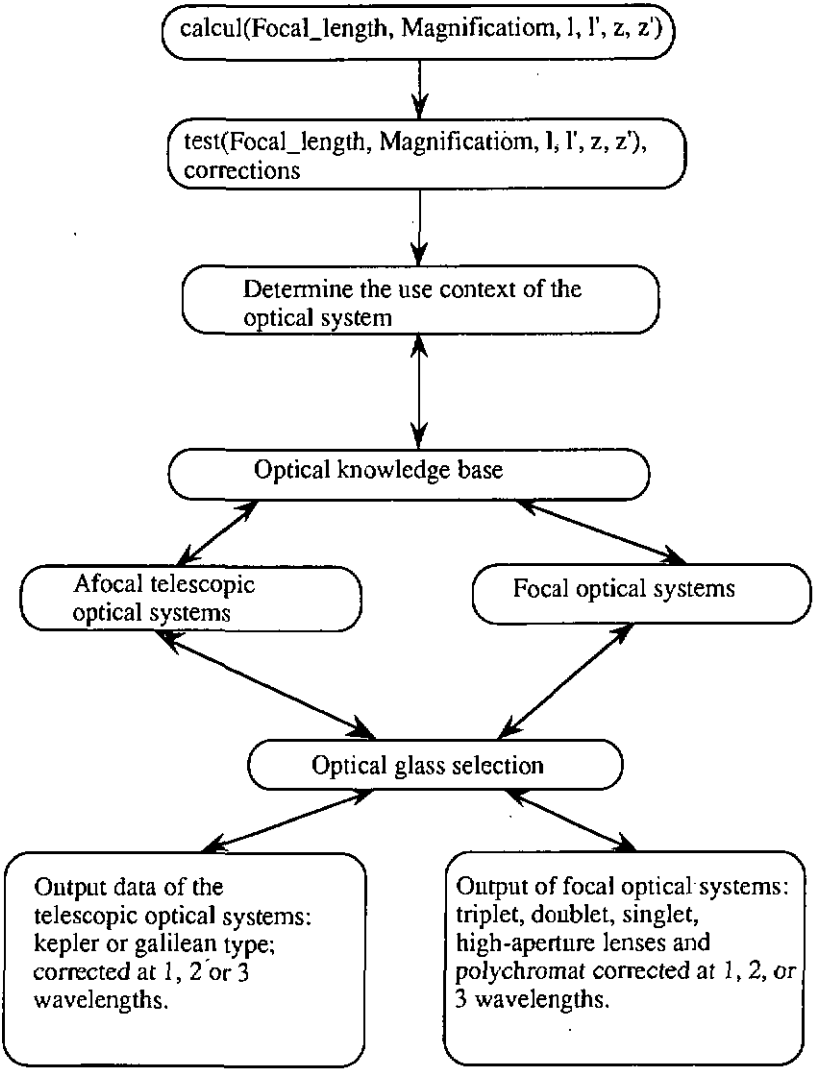


Fig. 2.3.1 General Structure of the KBOSD

2.3.2 Algorithms of Calculation of Parameters of the Optical Systems

The algorithm below serves to calculate the paraxial parameters of the optical system. The user must enter at least two parameters.

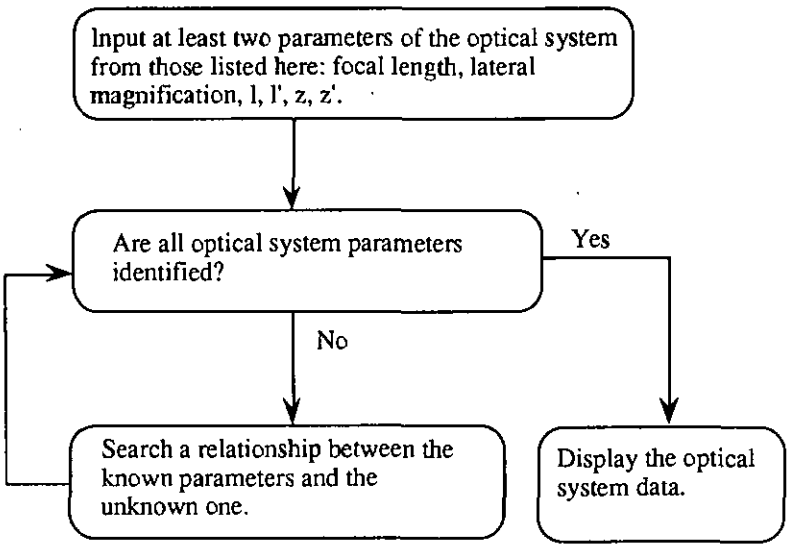


Fig. 2.3.2 Algorithms of Calculation of Optical System Parameters

2.3.3 Algorithms of Test of Parameters of the Optical Systems

The algorithm below tests the compatibility of the paraxial parameters of the optical system when the user enters more than two parameters.

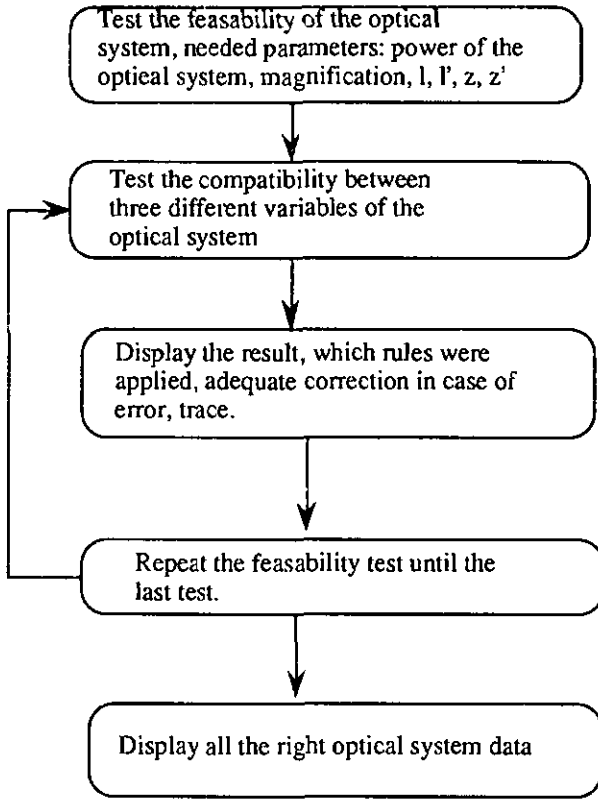


Fig. 2.3.3 Algorithms of the Test of Parameters of the Optical Systems

2.3.4 Algorithms of Determination of the Context of Use of the Optical System

The following algorithm determines the context of use of an optical system employing the relations between the opto-geometrical parameters such as diameter, entrance aperture, magnification, length of the optical system, focal length, object lens distance and lens image distance.

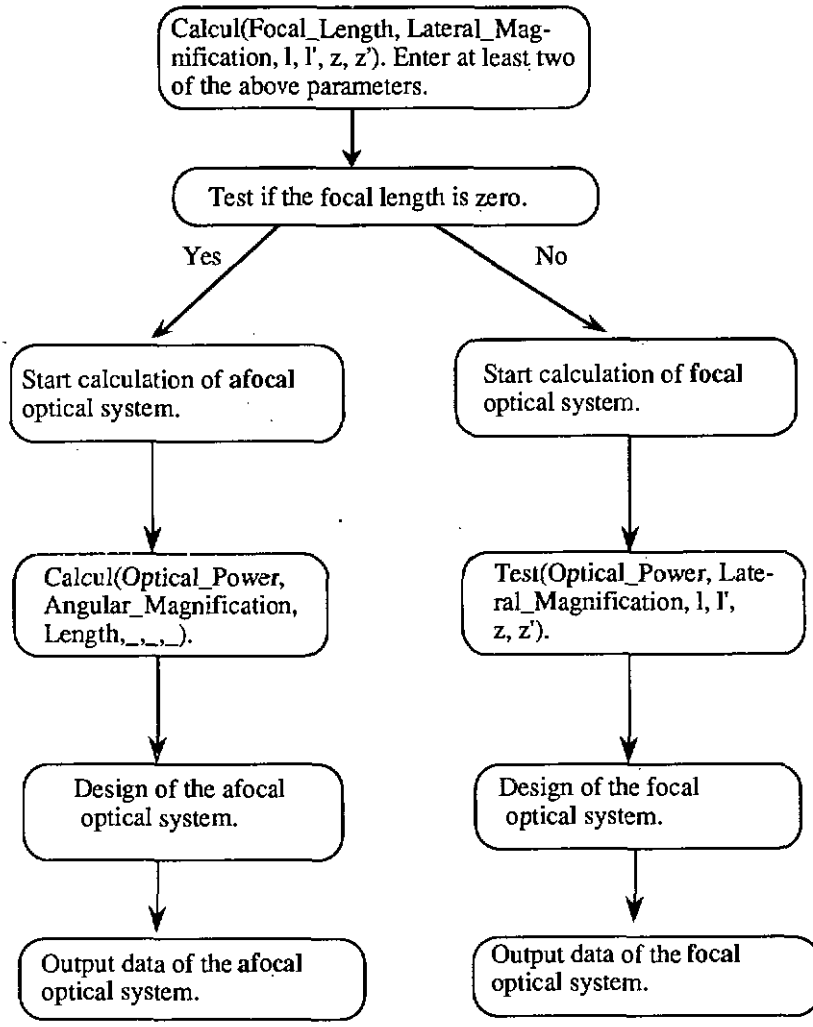


Fig. 2.3.4 Algorithms of Determination of the Context of Use of the Optical System

For more information concerning this section, see chapter 3.

2.3.5 Searching Algorithms of Focal Optical System

The following diagram serves to search the focal optical systems. Once the context of utilization of the optical system is identified according to the algorithm described in section 2.3.4 one proceeds to the specifications of the needed optical system. In addition to the opto-geometrical parameters, physical parameters such as the wavelengths of use are used for specifying and conceiving the optical system according to the algorithm below.

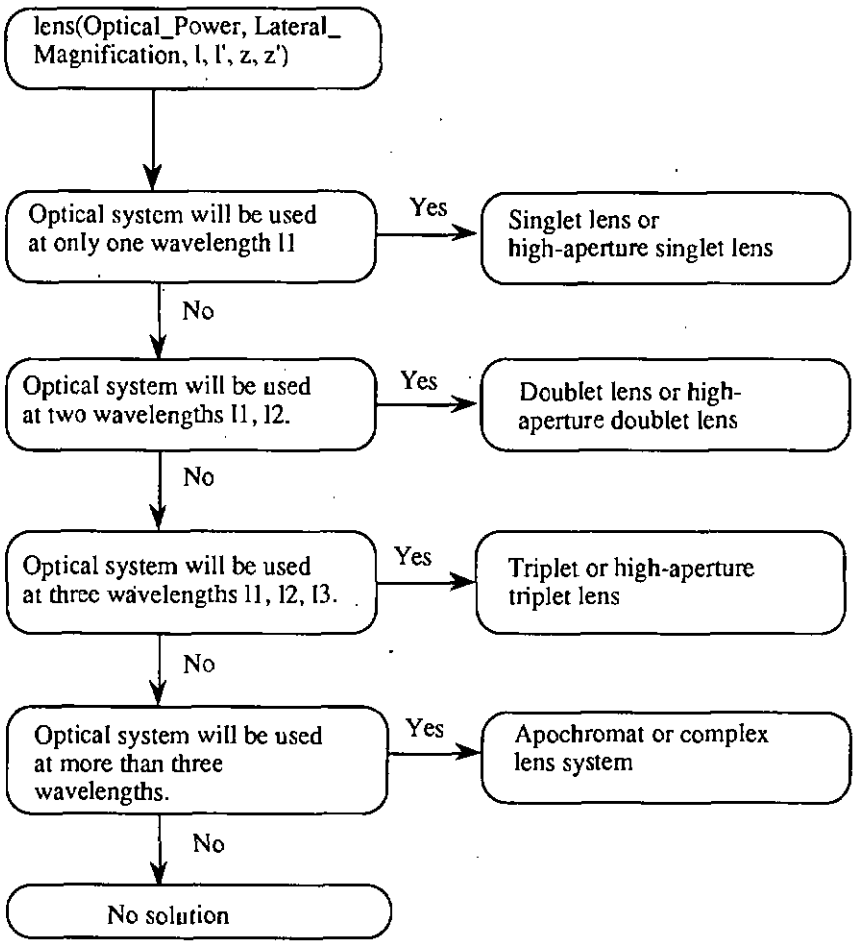


Fig. 2.3.5 Searching Algorithms of Focal Optical System

2.3.5.1 Searching Algorithms of a Singlet Lens

The algorithm below searches within the single lens and large-aperture single lens set in order to find the appropriate lens for the desired application. This search strategy is based on heuristics.

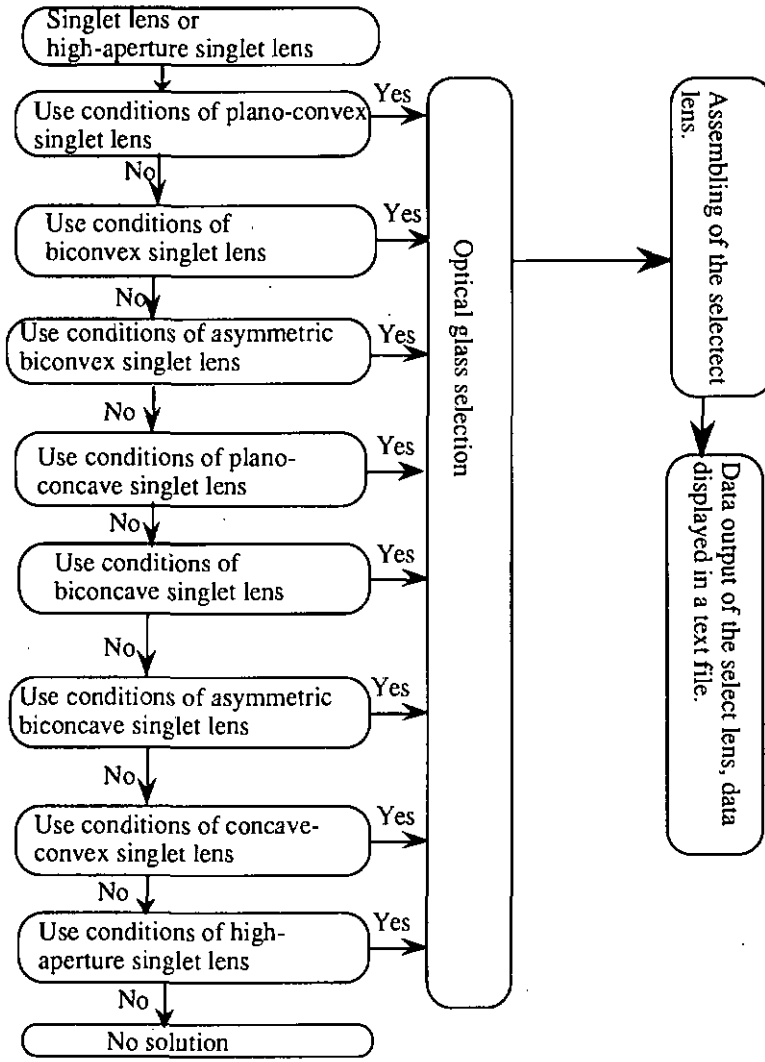


Fig. 2.3.5.1 Searching Algorithms of a Singlet Lens

2.3.5.2 Searching Algorithms of a Doublet Lens

The algorithm below searches within the doublet lens and large-aperture doublet lens set in order to find the appropriate lens for the desired application. This search strategy is also based on heuristics.

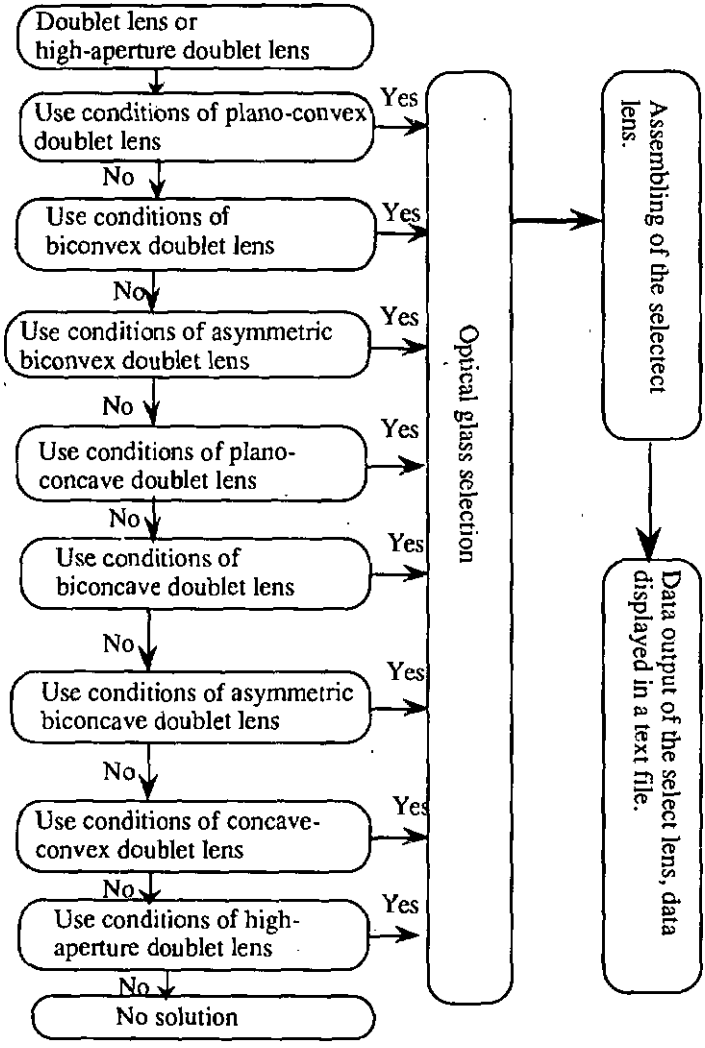


Fig. 2.3.5.2 Searching Algorithms of a Doublet Lens.

2.3.5.3 Searching Algorithms of a Triplet Lens

The algorithm below searches within the triplet lens and large-aperture triplet lens set in order to find the appropriate lens for the desired application. This search strategy is based on heuristics.

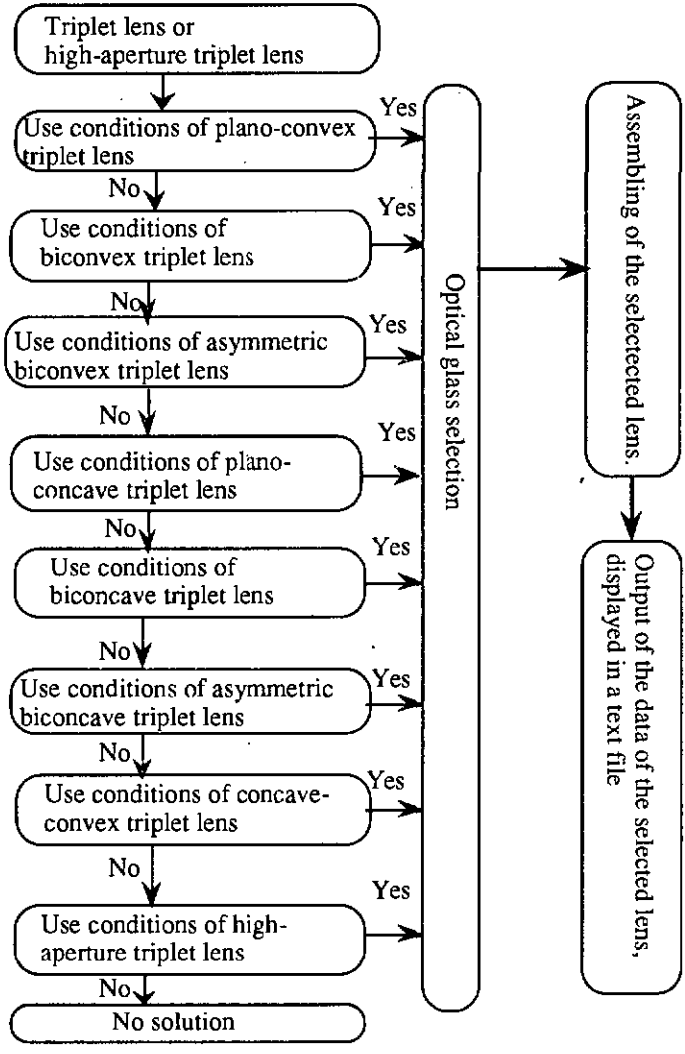


Fig. 2.3.5.3 Searching Algorithms of a Triplet Lens

2.3.6 Searching Algorithms of Afocal Optical System

This algorithm is concerned with searching for afocal optical systems. This search strategy is based on heuristics which uses the context of use of the optical system.

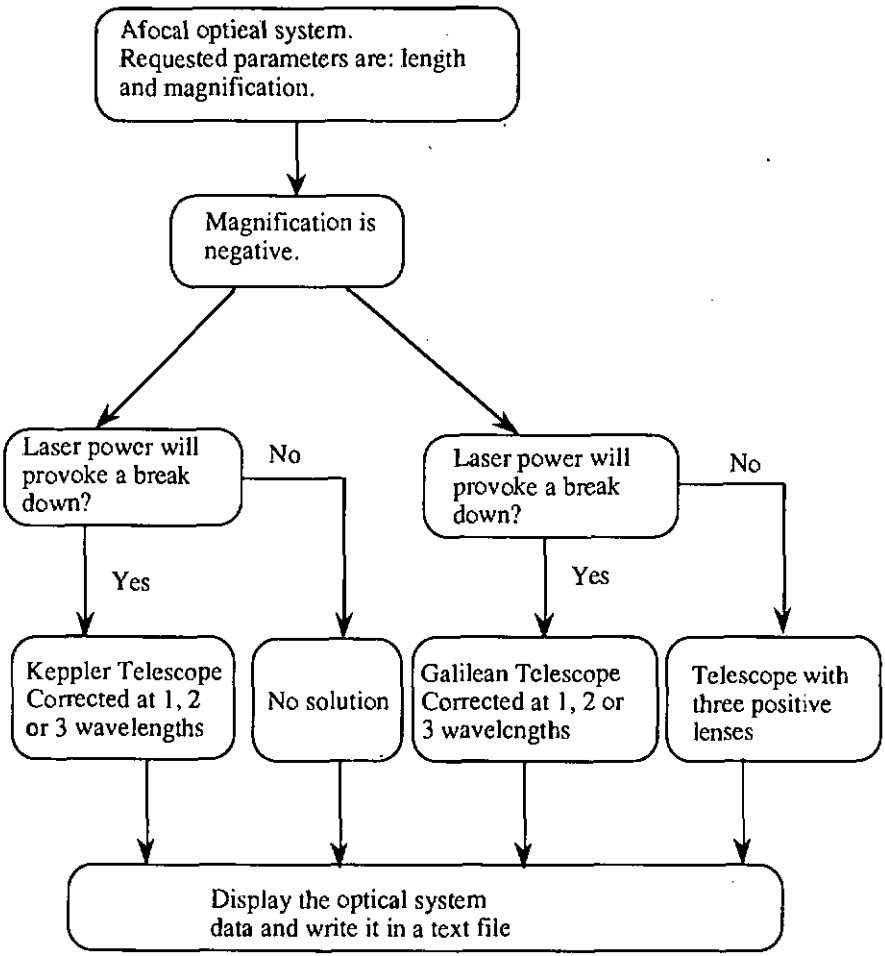


Fig. 2.3.6 Searching Algorithms of Afocal Optical System

2.3.6.1 Searching Algorithms of a Kepler Telescope

A Kepler telescope is formed by two optical systems:

- entrance optical system with a positive focal length
- exit optical system with a positive focal length

These entrance and exit optical systems can be formed by a combination of singlet, doublet, or triplet lenses according to the context of use of the telescope as displayed in the diagram below.

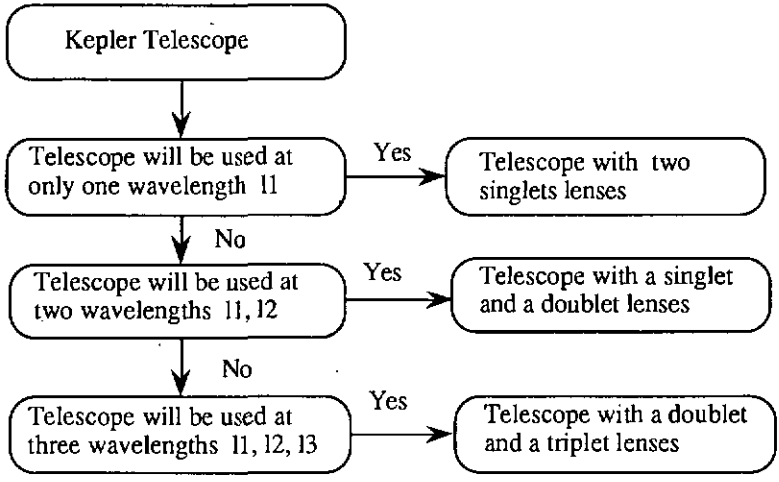


Fig. 2.3.6.1 Searching Algorithms of a Kepler Telescope

2.3.6.2 Searching Algorithms of a Galilean Telescope

A Galilean telescope is formed by two optical systems(see chap.3):

- entrance optical system with a negative focal length
- exit optical system with a positive focal length

These entrance and exit optical systems can be formed by singlet, doublet, or triplet lenses according to the context of use of the telescope as displayed in the below diagram.

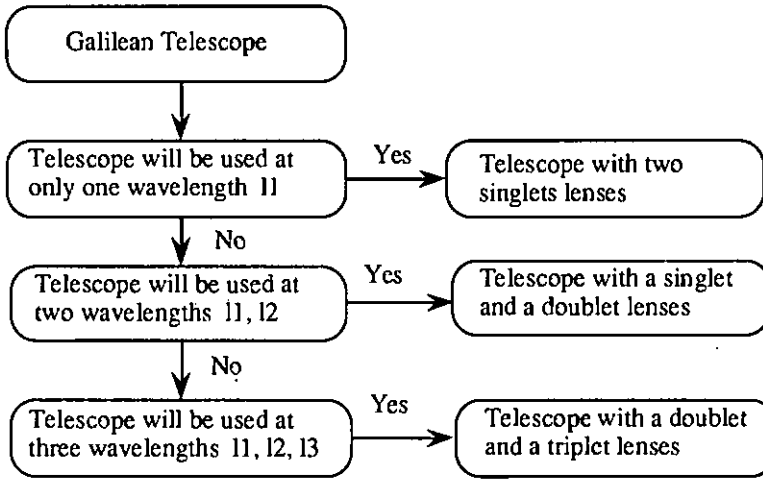


Fig. 2.3.6.2 Searching Algorithms of a Galilean Telescope

2.3.7 Searching Algorithms for Selection of an Optical Glass

The optical glass is one of the most important parameters in the calculation of an optical system. The search strategy of the optical glass is made according to a depth first searching.

The glass data base contains more than 250 different optical glasses.
The first element in the list of optical glasses is taken (depth-first searching) then it is tested according to the user's glass constraints, see Fig. 2.3.7. The constraint list is reviewed in depth searching too. This algorithm is repeated until the end of the list of optical glasses is reached.
This search strategy is used in the other section of this chapter.
This searching is also based on heuristics (see also chapter 3).

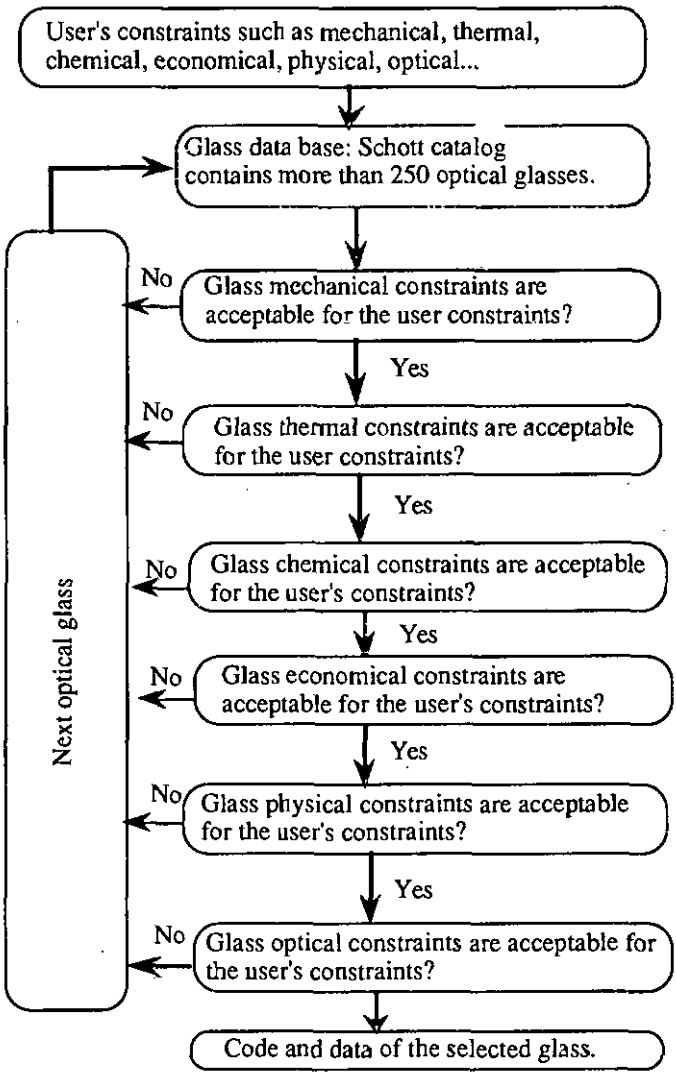


Fig. 2.3.7 Searching Algorithms for Selection of an Optical Glass

2.4 Conclusion

This chapter concerned some characteristics of the Prolog Programming Language as well as the artificial intelligence algorithms and topics applied to the optical system design.

The artificial intelligence approach helpes us to attempt many goals, especially in :

- making up base of facts containing optical glasses and their properties(Schott catalog)
- making up lens design knowledge base containing lens design constraints and rules, as well as heuristics on optical lens design, noting that a heuristic is a method of learning involve using reasoning and past experience rather than formulas or solutions that are given
- making up optical design knowledge models(formal and informal) such as:
  - heuristics and the context of use of optical systems are Informal.
  - algebraic models of optical systems
  - physical laws are formal
- implementation of algorithms of calculation and test of parameters of optical systems
- transfer of the optical design knowledge from the expert to the computer in terms of clauses
- solving optical design problems in a declarative way and not by a procedural algorithm
- thanks to the inference engine backtracking principle all solutions of a lens design problem are explored

Using the programming language Prolog, equipped with an inference engine, one has only to describe the base of facts, the optical rules, the heuristics and the relationship between the elements of the knowledge base as well as the optical problems and the program will find all possible solutions. For the mentionned above reasons Prolog is called descriptive or declarative programming language. The possibility of algorithmic or procedural programming in Prolog allows the control of the inference engine and the searching algorithms.

One must note that the algorithms presented in this chapter are simplified in order to facilitate their comprehension. The algorithms implemented in Prolog for the KBOSD are more complex. Ideas discussed in this chapter are reexamined in chapter 3 with a different approach.

3 Cognitive and Metacognitive Optical Systems Design

3.0 Introduction

3.1 Identification of the Optical Systems

3.1.1 Identification of the Optical Surfaces According to their Context of Use
3.1.2 Identification of Optical Elements According to their Context of Use
3.1.3.1 Identification of Optical Systems According to their Context of Use
3.1.3.2 Identification of Optical Systems According to their Geometrical Forms
3.1.4 Calculations of the Parameters of an Optical System
3.1.5 Test of the Parameters of an Optical System
3.1.6 The Influence of the Environment of Use on the Choice of the Optical Systems

3.2 Functional Classification of the Optical Systems

3.2.1 Focal Optical Systems
3.2.1.1 Simple Lens
3.2.1.2 Pair of Simple Reversed lenses
3.2.1.3 Doublets
3.2.1.4 Reversed Doublets
3.2.1.5 Triplets
3.2.1.6 Reversed Triplets
3.2.1.7 Apochromat and Complex Optical Systems
3.2.2 Afocal Optical Systems
3.2.2.1 Kepler's Afocal Optical System
3.2.2.2 Galilean Afocal Optical System

3.3 Heuristics on the Use and Assembly of the Optical System

3.3.1 Heuristics on the Use of the Optical System
3.3.2 Heuristics on the Assembly of the Optical Systems

3.4 Transfer of Optical Knowledge from the Optical Expert to the Computer and its Representation

3.4.1 Example of Rules of Optical Knowledge
3.4.2 Examples of Facts of Optical Knowledge
3.4.3 Examples of Algebraic Models of Optical Systems

3.5 The Organization of the Optical Knowledge in the Computer's Memory

3.5.1 Functional Organization
3.5.2 Metacognitive Organization
3.5.3 Interconnections Between the Components of the KBOSD

3.0 Introduction

This chapter treats:

- the identification, the classification and the heuristics of the assembly and the utilization of the spherical, centered, dioptrical optical systems.
- the transfer of the optical knowledge and optical expertise from the optical designer to the computer.
- the organization of knowledge and optical expertise in the computer's memory.
- the reasoning used by the computer to solve an optical design problem.

Thus, one is interested in the cognitive aspect of the optical systems, i.e. in their categorizations, their structures, their models, and their interconnections and how to make them computable and comprehensible by the machine.

The power of reasoning and knowledge strategies are adopted and do not rely on the power of the calculations in the classical sense such as processing speed, hardware etc. KBOSD is not occupied with extracting information from the optical expert; we suppose that the expertise is acquired and is at our disposal.

KBOSD contains around 3000 clauses; those presented are only illustrations.

3.1 Identification of the Optical Systems

A centered optical system is constituted by a series of media separated by the diopters; it revolves around an axis called the optical axis. An optical system, an optical element, or an optical surface can be identified by its function or geometrical forms.

3.1.1 Identification of the Optical Surfaces According to their Context of Use

An optical surface is the base element in forming an optical element; it is identified by:

- its curvature radius
- its diameter
- its optical power

The diameter is an arbitrary algebraic dimension which is always positive. The curvature radius is an algebraic dimension that can be negative, positive or infinite. The curvature radius attributes optical power to the optical surface. If

the curvature radius is positive (in relation to the incident direction of light, positive from left to right), it causes the light to converge; thus, one would say that it has a positive optical power. If the curvature radius is negative, it makes light diverge, thus its optical power is negative. These parameters determine the context of uses of the optical surfaces.

3.1.2 Identification of Optical Elements According to their Context of Use

An optical element (lens) consists of two refracting interfaces, where at least one of these is curved. An optical element can be identified by the following opto-geometrical parameters:

- its optical surfaces
- its optical power
- its material (in general of glass)

The power of an optical element is the algebraic sum of the power of its optical surfaces. The curvature radius of each optical surface allocates its usefulness to the optical element. One distinguishes two categories of optical elements according to the sign of their optical power:

- optical elements of positive power (convergent)
- optical elements of negative power (divergent)

In each of these two categories, one distinguishes a dozen different optical elements that are recognized according to the ratio and the signs of their curvature radii. The power, the signs and the ratio of the curvature radius give us the context of use of the optical element.

3.1.3.1 Identification of Optical Systems According to their Context of Use

An optical system is formed by an element or a series of optical elements. It is identified by its optical power and the number of elements it is made up of. The power of an optical system is the algebraic sum of the power of its elements.
Parameters such as power, magnification (-1/1'), (see Fig. 3.1.4) determine the context of use of the optical system.

3.1.3.2 Identification of Optical Systems According to their Geometrical Forms

The optical systems, as well as optical elements, can be identified by the geometrical forms of their external surfaces according to the table below. An optical element of positive power is thicker in the center than along the edges. An optical element of negative power is thicker on the edges than in the center.

First optical surface	Second optical surface	Optical system
Plane	Convex	Plano-convex
Plane	Concave	Plano-concave
Convex	Convex	Bi-convex(symmetric or asymmetric)
Convex	Concave	Meniscus concave, meniscus convex...
Concave	Concave	Bi-concave (symmetric or asymmetric)

Fig. 3.1.3.2 Example of identification of optical systems according to their geometrical forms

3.1.4 Calculations of the Parameters of an Optical System

One of the characteristics of the KBOSD is its capacity to supply a solution even if the input data for the given problem is not fully available in the beginning. Having knowledge of at least two of the parameters indicated in figure 3.1.4, one can rediscover all the data of an optical system and demonstrate its existence.
The four parameters: Power, Magnification (-1/1'), 1 and 1' (see Fig. 3.1.4) and their relationships perfectly identify the functions as well as the optimal use conditions of an optical system.

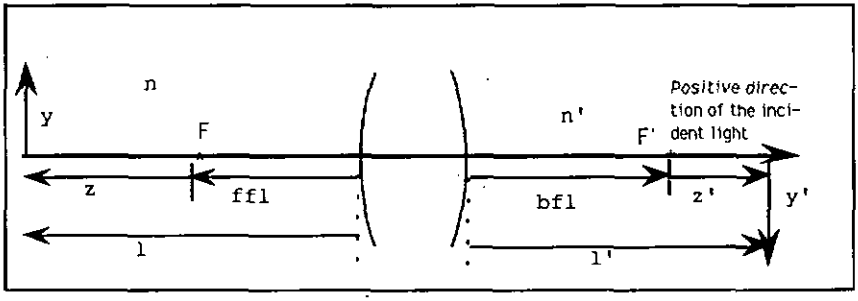


Fig. 3.1.4 Parameters of an optical system

Concerning the significance of the variables, see the list of symbols at the beginning of this text.

Here we describe an algorithm developed in the Prolog syntax [Ref. 14], that permits the calculation of these parameters (for more details, see chapter 2).

```
calcul(Optical_Power, Transversal_Magnification, l, l', z, z') :-
    known_variable([Optical_Power,Transversal_Magnification,l, l', z, z']),!,
    output_result(Optical_Power, Transversal_Magnification, l, l', z, z').

calcul(Optical_Power, Transversal_Magnification, l, l', z, z') :-
    known_variable([l,l']),
    relation(rules1, Optical_Power, (n/l) + (n'/l'), "(n/l) + (n'/l')",
    "Optical_Power"),!,
    calcul(Optical_Power, Transversal_Magnification, l, l', z, z').

calcul(Optical_Power, Transversal_Magnification, l, l', z, z') :-
    known_variable([l', z']),
    relation(rules2, Optical_Power, n'/(l'-z'), "n'/(l'-z')", "Optical_Power"),!,
    calcul(Optical_Power, Transversal_Magnification, l, l', z, z').

  calcul(Optical_Power, Transversal_Magnification, l, l', z, z') :-
    known_variable([z, z']),
    relation(rules3, Optical_Power, -sqrt((n*n')/(z*z')),!,
    calcul(Optical_Power, Transversal_Magnification, l, l', z, z').

  calcul(Optical_Power, Transversal_Magnification, l, l', z, z') :-
    known_variable([l,z]),
    relation(rulesn, Optical_Power, n/(l-z), "n/(l-z)", "Optical_Power"),!,
    calcul(Optical_Power, Transversal_Magnification, l, l', z, z').
```

The predicate *calcul(Optical_Power, Transversal_Magnification, l, l', z, z')* above calculates a third variable aside from the two known variables (A,B), *known_variable([A,B])*, variable A, B belong to the variable set (Optical_ Power, Transversal_Magnification, l, l', z, z').
The predicate *calcul(Optical_Power, Transversal_Magnification, l, l', z, z')* will be repeated until all the parameters of the optical system are determineted, thus satisfying the function *known_variable(Optical_Power,Transversal_Magnification, l, l', z, z')*.

The predicate *relation(rulesn, Optical_Power, n/(l-z),"n/(l-z)", "Optical_ Power")* establishes the relation between the researched variable, Optical_ Power and the variables already known n,l, and z.

3.1.5 Test of the Parameters of an Optical System

Knowing more than two of the optical system parameters, one is obliged to test their compatibility. If such an optical system exists, then KBOSD begins to process it. If the parameters are not compatible, an error message is displayed and an appropriate correction is proposed to the user. In reality, the rules of testing are numerous.

Here is an example of the algorithms of the parameters test of an optical system that have been developed in the Prolog syntax [Ref. 14]. The principle of these algorithms is demonstrated by the following example (for more detail see chapter 2):
Knowledge of the three parameters, *known_variable([l', z', Optical_Power])*, test if the value of the variable Optical_Power is equal to the value of the expression n'/(l'- z') and rectifies the result in case of an error.

```
test1(Optical_Power, Transversal_Magnification, l, l', z, z') :-
        known_variable([l', z', Optical_Power]),
        relationtest(test2,Optical_Power,n'/(l'-z'),"n'/(l'-z')",
        "Optical_Power"),!,
        test2(Optical_Power, Transversal_Magnification, l, l', z, z').

test2(Optical_Power, Transversal_Magnification, l, l', z, z') :-
        known_variable([z, z', Optical_Power]),
        relationtest(test3, Optical_Power, sqrt(n*n'/(z*z')),"sqrt(n*n'/(Opti
        cal_Power)"),!,
        test3(Optical_Power, Transversal_Magnification, l, l', z, z').

test3(Optical_Power, Transversal_Magnification, l, l', z, z') :-
        known_variable([l, z, Optical_Power]),
        relationtest(test3,Optical_Power,n/(l-z),"n/(lz)", "Optical_Power"),!,
        testn(Optical_Power, Transversal_Magnification, l, l', z, z').

testn(Optical_Power, Transversal_Magnification, l, l', z, z').
```

3.1.6 The Influence of the Environment of Utilization on the Choice of the Optical System

The medium of use of an optical system plays an important role in the choice and the realization of the optical system. The material forming the optical system (in general glass) must resist many factors in the utilization environment such as mechanical and climatic factors... .

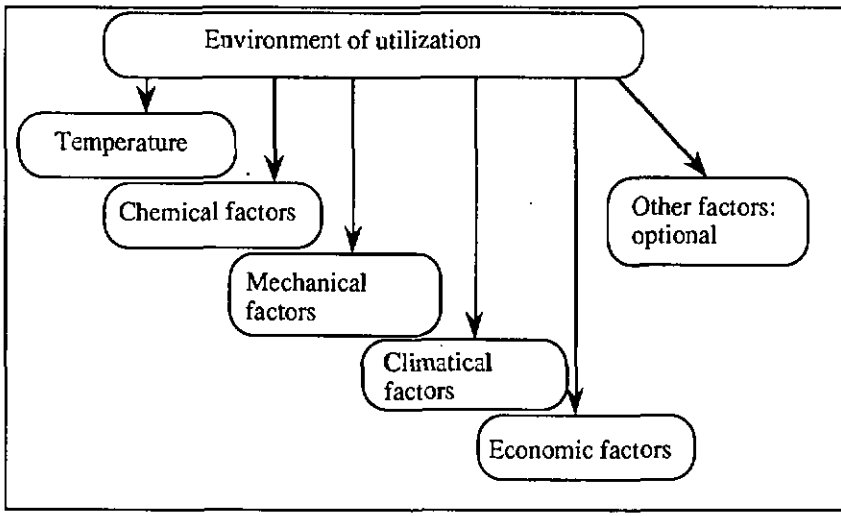


Fig. 3.1.6 Examples of constraints in the utilization environment

Other factors play a determining role in the final elaboration of the optical system, such as economic factors (e.g. the price of glass), the polishing time of the glass, and physical factors such as the intensity of the light used, the required precision of the final quality, as well as opto-geometrical factors like the diameter of the entrance pupil of the optical system. If these constraints are not known by the user, then the KBOSD uses values determined by heuristics.

3.2 Functional Classification of the Optical Systems

From the functional aspect, an optical system serves to form an image (real or virtual) of an object (real or virtual). Thus optical systems can be classified according to their function into two categories.

- an optical system with zero power is called an afocal system;

- an optical system whose power is not nil serves to focus the light; this system is called a focal system.

Each of the two categories is composed of a subset of elements or a subset of optical systems.

3.2.1 Focal Optical Systems

They can be formed by one or several optical elements, according to the wavelengths of the light that traverses them and according to the diameter of their entrance pupil. Illustrated below are some examples of focal optical systems.

3.2.1.1 *Simple Lens*

In effect, to design an optical system of acceptable quality, utilized at a low entrance pupil and at a single wavelength, a single optical element (lens) is in general sufficient.
This simple optical system called a singlet.

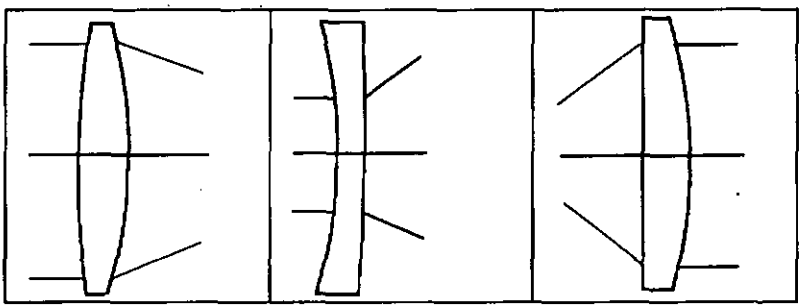


Fig. 3.2.1.1 Example of a singlet lens: Bi-Convex, Plano-Concave and Plano-Convex

3.2.1.2 Pair of Simple Reversed lenses

If the entrance pupil becomes large, two elements judiciously assembled would be necessary to reduce spherical aberrations. This system is called a large-aperture singlet.

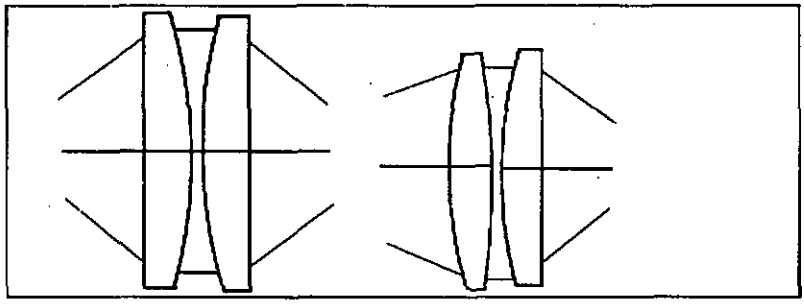


Fig. 3.2.1.2 Examples of Pair of Simple Reversed lenses

3.2.1.3 Doublets

To design optical systems at two different wavelengths, one assembles two optical elements to achieve the achromatic conditions. Such optical systems are called doublets.

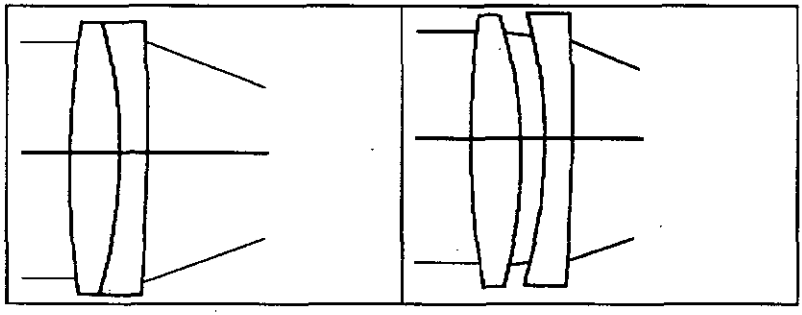


Fig.3.2.1.3 Cemented and Separated Doublets

3.2.1.4 Reversed Doublets

If the entrance pupil becomes large, two judiciously assembled doublets would be necessary to reduce the aspherical aberrations. Such a system is called a large aperture reversed doublet.

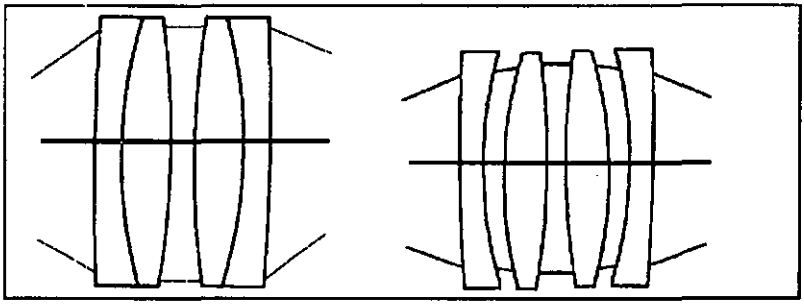


Fig. 3.2.1.4 Large-aperture reversed doublets

3.2.1.5 Triplets

With three different wavelengths one is can take three optical elements to achieve achromatic conditions.

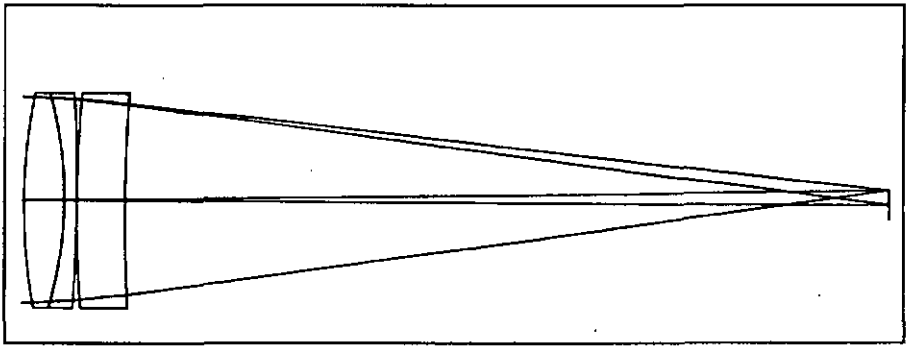


Fig. 3.2.1.5 Triplet

3.2.1.6 Reversed Triplets

If the entrance pupil becomes large, one is obliged to take two triplets judiciously assembled to reduce the spherical aberrations. This lead to an optical system consisting of six elements; such a system is called a reversed triplet.

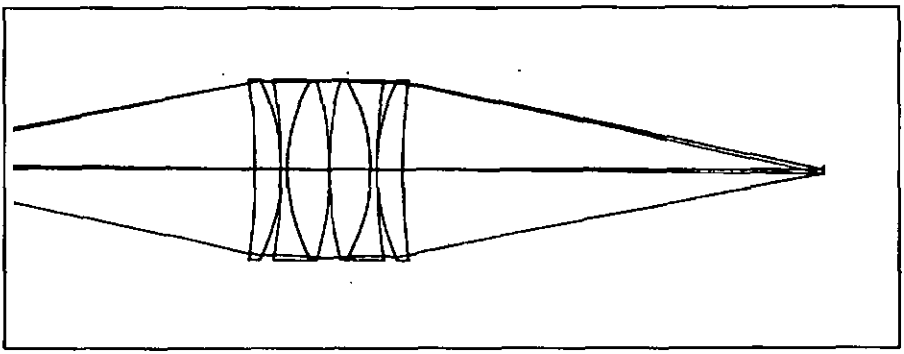


Fig. 3.2.1.6 Reversed Triplets

3.2.1.7 Apochromat and Complex Optical Systems

The number of optical elements grows with the number of wavelengths of the utilized light. To correct for the different aberrations one uses apochromat or complex lenses. These can always be formed by assembling the optical systems cited above.

3.2.2 Afocal Optical Systems

As mentioned above, if the power of an optical system is nil, it is recognized as afocal, one also says that its two focal points are at infinity. In this case all incident rays parallel to the axis of the optical system emerge parallel to the axis, as in the telescope.
Such an optical system is composed of two blocks:

- An entrance optical system
- An exit optical system

These two blocks are optical systems that can be composed of singlet, doublet or triplet lenses.

The exit block (which has a power greater than zero) is always positive and can be formed by:

- a simple lens, or a singlet
- a doublet
- a triplet.

The entrance block can be positive or negative and is formed by:

- a simple lens (a singlet)
- a doublet
- a triplet.

The number of optical elements forming an afocal system varies according to the entrance pupil, magnification, as well as according to the wavelengths of its utilization.

3.2.2.1 Kepler's Afocal Optical System

One uses a positive entrance block for:

- low light level and/or when
- the total available length of the optical system is relatively large.

This block assembled with a positive exit block is called Kepler's Telescopic Optical System.

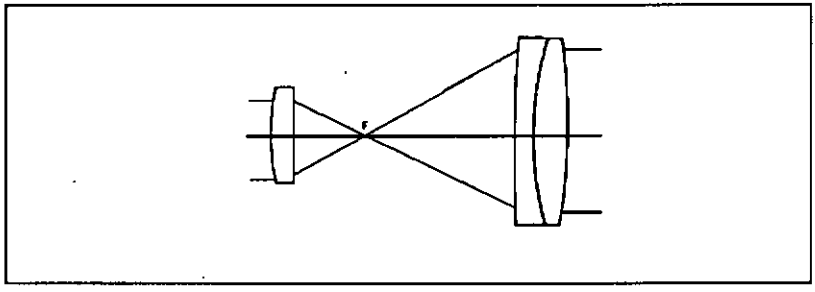


Fig. 3.2.2.1 Kepler's Telescopic System

The fact that the light beam is concentrated on a single point(F), between the two blocks, is not always advantageous. This is true especially in the case of high power lasers (see Fig. 3.2.2.1).

3.2.2.2 Galilean Afocal Optical System

One uses a negative entry block if:

- a high intensity light(e.g. laser) is going to be used and/or
- the total available length of the optical system is limited.

This system is called Galilean Telescopic Optical System.

If a positive optical system is used in the entry of an afocal optical system, transversed by a high intensity light, a breakdown phenomenon is provoked. Such a phenomenon is not produced with a negative optical system (divergence). This breakdown phenomenon is a microexplosion that deteriorates the quality of the wavefront. Such a situation is to be avoided.

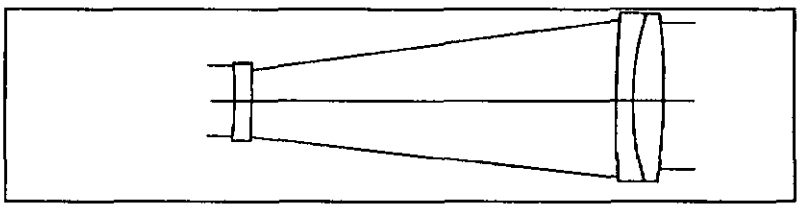


Fig. 3.2.2.2 a) Galilean Telescopic Optical System with Cemented Doublet

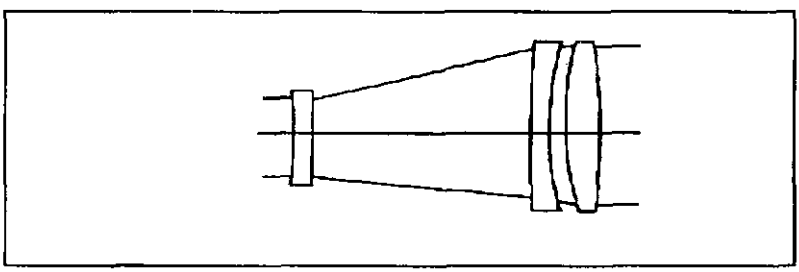


Fig. 3.2.2.2 b) Galilean Telescopic Optical System with Separated Doublet

3.3 Heuristics on the Use and Assembly of the Optical System

The KBOSD is equipped with the heuristics concemed with:

- mechanical, chemical, and thermal properties of optical glass

- the facility of polishing, the availability and the price of the optical glass
- calculations including, the correction of chromatic and geometrical aberrations as well as the quality of the optical systems
- conditions of use and selection criteria of the optical systems
- assembly of the optical systems

Presented below are some examples of heuristics.

3.3.1 Heuristics on the Use of the Optical System

Aside from the parameters such as the focal length, l, l', Transversal_Magnification; we recognize the context of utilization of the optical system. Thanks to the context and the identification of the optical system, we can apply the heuristics to determine the optimal utilization. Here are some examples of heuristics that are programmed in Prolog in terms of rules.

a) Plano-convex optical system

If the lateral magnification is less than -5, a plano-convex optical system is supposed to have minimum aberrations. If l' is infinite, the optical system serves to collimate the light.

```
use_planoconvex(Optical_Power, Transversal_Magnification, l, Diameter) :-
        convergent( Optical_Power),
        low_aperture( Optical_Power, Diameter),
        negative_transversal_magnification(Transversal_Magnification),
        near_infinite_ratio(Transversal_Magnification),
        low_optical_power( Optical_Power),
        object_at_infinite( Optical_Power, l).
```

b) Bi-convex optical system:

```
use_biconvex( Optical_Power, Transversal_Magnification, l, Diameter) :-
        convergent( Optical_Power),
        low_aperture( Optical_Power, Diameter),
        negative_transversal_magnification(Transversal_Magnification),
        high_optical_power( Optical_Power),
        finite_ratio(Transversal_Magnification).
```

c) Bi-convex optical system: If the lateral magnification is -1, the bi-convex optical system has a minimum of aberrations.

```
use_biconvex( Optical_Power, Transversal_Magnification, l, Diameter) :-
        convergent( Optical_Power),
```

low_aperture(Optical_Power, Diameter),
negative_transversal_magnification(Transversal_Magnification),
unitratio(Transversal_Magnification).

d) Plano-concave optical system:

use_planoconcave(Optical_Power,Transversal_Magnification, l, Diameter) :-
divergent(Optical_Power),
low_aperture(Optical_Power, Diameter),
positive_transversal_magnification(Transversal_Magnification),
near_infinite_ratio(Transversal_Magnification),
low_optical_power(Optical_Power),
object_at_infinite(Optical_Power, l).

e) High-aperture optical system:

use_highaperture(Optical_Power,Transversal_Magnification, l, Diameter) :-
convergent(Optical_Power),
large_aperture(Optical_Power, Diameter).

f) Bi-concave asymmetric optical system:

use_asymetricbiconcave(Optical_Power, Transversal_Magnification, l, Diame
ter):-
divergent(Optical_Power),
low_aperture(Optical_Power, Diameter),
positive_transversal_magnification(Transversal_Magnification),
low_optical_power(Optical_Power),
finite_ratio(Transversal_Magnification),
not_unit_ratio(Transversal_Magnification).

Clauses such as:

low_aperture(Optical_Power, Diameter),
near_infinite_ratio(Transversal_Magnification),
low_optical_power(Optical_Power),
object_at_infinite(Optical_Power, l).

are also heuristics that are explicitly described in other places of the KBOSD.

3.3.2 Heuristics on the Assembly of the Optical System

A heuristic is in general a rule of reasoning recognized by the specialists, It is, however, of an inductive nature and risks being modified if a contradiction is discovered [Ref. 12].

If the opto-geometrical parameters the goal and the condition of use of an optical system are known and identified; then one begins to assemble the different elements that constitute it. This assembly procedure is based on heuristics. The different elements forming an optical system must abide by certain physical constraints (achromatics, corrections of aberrations....), geometrical constraints (diameter, thickness,etc.), economical constraints (price, the facility of treating the glass making up the optical elements, resistance of the glass to chemical and climatic factors....).

Here are some examples of heuristics.

a) In order for a doublet to be achromatically corrected one must use two different glasses; one is the crwon type and the other is the flint type.

b) If two optical surfaces follow one another and if their curvature radii have the same sign(positive or negative) and they have to be cemented to each other, then, the second curvature radius must be smaller than the first; otherwise, the assemblage is complicated.

c) If the curvature radius of an optical surface is smaller than its diameter, its fabrication is equally impossible.

d) If two optical surfaces that have the same curvature radius follow one another and are utilized at low temperature, one can cement them together.

e) If the temperature is high, one cannot cement the optical surfaces together because the cement is not resistant to high temperatures. This heuristic can be modified if a cement is discovered that resists high temperatures.

f) If an optical system is formed by three optical elements, the middle element does not necessarily have to be resistant to chemical and climatic factors because it is protected on the two sides by the other elements.

g) If an optical system is used at three different wavelengths, it must be composed of three types of glass to correct for the different chromatic aberrations(two types of glass can be enough, but did not consider this heuristic).

3.4 Transfer of Optical Knowledge from the Optical Expert to the Computer and its Representation

The classification, identification, context of use, heuristics and assembly procedures of optical systems facilitate mastering the optical knowledge and its representation so that the human expertise can be transferred to the machine. The optical knowledge is represented in the form of clauses (rules and facts) and in algebraic models. Our representation of the knowledge can be examined

and comprehended in "human terms." Additionally, the rules, the facts, as well as the requests can be expressed in Prolog, in the form of clauses [Ref. 21], [Ref. 22].

3.4.1 Example of Rules of Optical Knowledge

Optical knowledge is expressed by rules. We display a few examples of these rules:

lens(Optical_Power, Transversal_Magnification, l, l') :-
wavelengths(L1, L2),
use_planoconvex(Optical_Power, Transversal_Magnification, l, l'),
planoconvex_doublet(Optical_Power, Th, R1, R2, R3, R4, N1, N2, D1, S1).

If such a rule is invoked, the inference engine begins by testing:

- if there are two wavelengths L1,L2; if the conditions used by a plano-convex optical system are satisfied, then, the conclusion is:
- the plano-convex doublet rule is applied that begins to calculate a known optical system under the name plano-convex doublet.

At the same time, each of these rules can in turn invoke other rules and other clauses.

assemble_doublet(Optical_Power, Optical_Power1, Optical_Power2, Refrac
tive_Index1, Refractive_Index2, Diameter, Separation) :-
Refractive_Index1 is crown(V1),
Refractive_Index2 is flint(V2),
Optical_Power1 is V1 Optical_Power/(V1-V2),*
Optical_Power2 is V2 Optical_Power/(V1-V2).*

This rule selects two glasses (Refractive Index 1, Refractive Index 2), one is of the flint type and the other is crown. Then it imposes strict constraints on Optical Power 1 and Optical Power 2.

3.4.2 Examples of Facts of Optical Knowledge

The data base contains indispensable data for the KBOSD [Ref. 9]. The information contained in this data base are static and permanent, they are readable and non-modifiable while being executed. Some examples are provided.

Example 3.4.2-1:

This clause gives the dispersion coefficient of an optical glass.

dispersion_coefficient(Glass_Name, A_0, A_1, A_2, A_3, A_4, A_5).
dispersion_coefficient(baf3, 2.4549347, $-8.3372035.10^{-3}$, $1.6841270.10^{-2}$, $5.0168527.10^{-4}$, $-1.4413749.10^{-5}$, $2.0771351.10^{-6}$).

This signifies that the optical glass called baf3 possesses the following dispersion coefficients A_0, A_1, A_2, A_3, A_4, A_5.

Example 3.4.2-2:

physical_properties(Glass_Name1, Alpha1, Alpha2, T_g, T, C_p, Lambda).
physical_properties(baf4, 7.9, 8.8, 521, 694, 0.557, 0.766).

This establishes a relation between the glass, called baf4, and the numerical values of certain of its physical properties.

Example 3.4.2-3:

glass_code(baf12, 639452, 1.63930, 45.18, 0.014151, 1.64266, 44.88, 0.014318).

Glass_Name1	: Name of glass is baf12.
Glass_Name2	: Coded name of the glass is 639452
Nd	: 1.6393 refractive index of the glass at a wavelength of 587.6 nm
Nued	: 45.18, Abbe number of the glass at a wavelength of 587.6 nm
Nf-Nc	: is 0.014151
Ne	: is 1.64266, refractive index of the glass at a wavelength of 546.1 nm
Nuee	: is 44.88 Abbe number of the glass at a wavelength of 587.6 nm
Nf- Nc'	: is 0.014318 dispersion coefficient

3.4.3 Examples of Algebraic Models of Optical Systems

Example 3.4.3-1:

The algebraic model permits the calculation of the refractive index of an optical glass according to the following formula:
*Refractive_Index is sqrt(A_0+A_1*l^2+A_2*l^{-2}+ A_3*l^{-4} + A_4*l^{-6} +A_5*l^{-8}).*
where:

l	: wavelength ;
A_0, A_1, A_2, A_3, A_4, A_5	: dispersion coefficient
sqrt	: square root.

Example 3.4.3-2:

This algebraic model links the parameters of plano-convex lens.

plano_convex_lens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index, Diameter) :-
Radius1 is (Refractive_Index - 1.0)/ Optical_Power,
Radius2 is Infinite,
lens_thickness(Optical_Power, Radius1, Radius2, Diameter, Thickness).

Example 3.4.3-3:

This algebraic model describes the composition of the plano-convex triplet, formed by three lenses: a plano-convex lens, a bi-concave lens, and a bi-convex lens which provide triplet lens parameters and each of the three lenses.

plano_convex_triplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,Radius2,Radius3,Radius4,Radius5,Radius6,Refractive_Index1, Refractive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3, Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2) :-
plano_convex_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1, Diameter),
biconcave_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refracttive_Index2, Diameter),
biconvex_lens(Power3, Thickness3, Radius5, Radius6, Refract ive_Index3, Diameter).

3.5 The Organization of the Optical Knowledge in the Computer's Memory

3.5.1 Functional Organization

One shares the computer's memory in cognitive areas having different dimensions. The cognitive areas with limited cognitive dimensions are available to those with larger cognitive dimensions. These cognitive zones include the procedures and the structures that assist them to satisfy their functions. A cognitive area represents optical systems having the same function and structure. For example, in the cognitive area called Triplet lens only optical

systems of the triplet lens kind can exist, just as the simple lens cognitive zone does not contain doublets or triplets. Such use of memory permits easy and rapid access of the desired information.

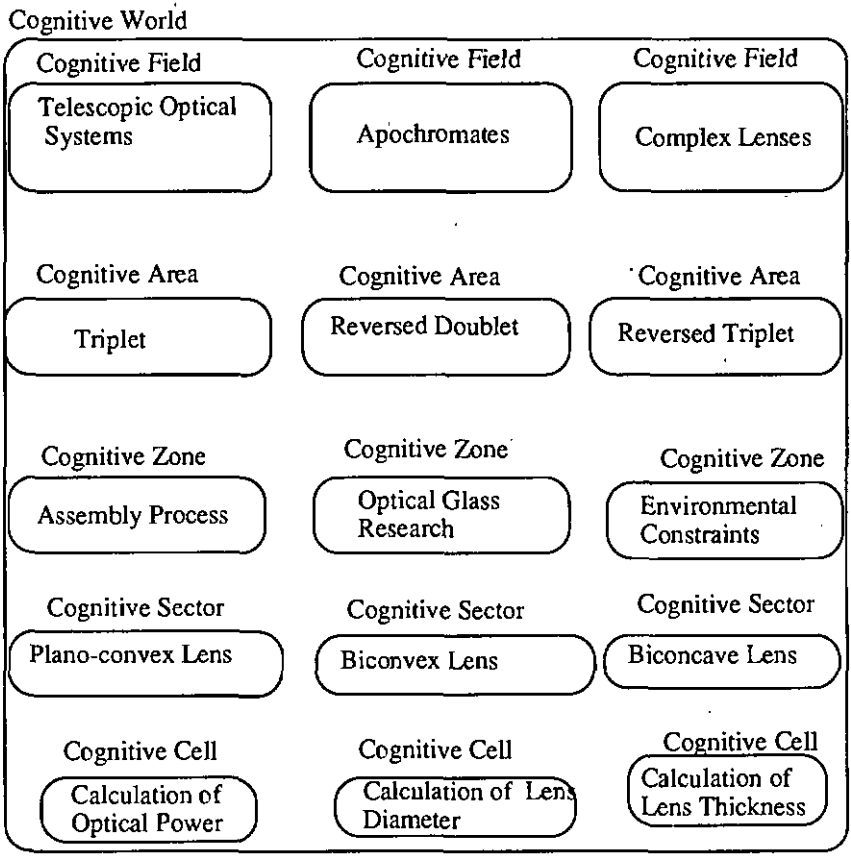


Fig. 3.5.1 Organization of the Optical Knowledge in the Memory of The Computer

3.5.2 Metacognitive Organization

The assembly of these optical systems is a complicated procedure that must satisfy many constraints. The classification, the order of the use of the information, the prioritization of the constraints, and the method with which the machine processes the tests of these conditions is called metacognitive

organization. It is the knowledge on knowledge or the organization of knowledge [Ref. 12].
This metacognitive organization is essential for the efficiency and the speed of reasoning. This is especially true if the KBOSD possesses a large number of rules. Supposing that the assembly procedure must satisfy five conditions, the first is the most complicated one. It has the least chance of success as the conditions are classified (according to heuristics) in the order of decreasing complexity and with the probability of growing success.

Example 3.5.2-1:

The following Prolog clause represents an optical system, concave-convex doublet:

```
lens( Optical_Power, Transversal_Magnification, l, l') :-
    wavelength( [Lambda1, Lambda2]),
    use_concaveconvex( Optical_Power, Transversal_Magnification, l, Dia-
    meter),
    concaveconvex_doublet( Optical_Power, Thickness, Radius1, Radius2,
    Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Name1,
    Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
    output_doublet_lens(Optical_Power, Thickness, Radius1, Radius2,
    Radius3, Radius4, Refractive_Index1, Refractive_Index2,
    Glass_Name1, Glass_Name2, Diameter, Lambda1, Lambda2, Separa
    tion),
    stopsurface(Position, Diameter_entree).
```

The first function is to look at rule one, then rule two, and then rule three as below:

The first of these rules is the most important to verify and it has the greatest chance of failing; if it fails, one looks for an alternative, otherwise one stops. In this way, one avoids supplementary work on the machine; in the event that it succeeds, the following rules (rules two and then rule three) have a greater possibility of success. Such a strategy assists us avoid unnecessary calculations.

3.5.3 Interconnections Between the Components of the KBOSD

Each of the cognitive areas possesses agents containing informations on:

- their own composition
- the functions for which they are designated
- the method necessary to accomplish these functions

Thus a cognitive area knows where to find the information to accomplish its tasks while it does not know who it must serve. It always stands at the disposition of the other cognitive areas that will call on it. These cognitive agents represent the interconnections between the diverse cognitive areas.

For example, the cognitive area triplet can call on the cognitive areas singlet, doublet or assemblage constraints. It possesses all the links with the objects it needs. On the contrary, it does not know that it can be called on by telescopic optical systems or other objects (see Fig. 3.5.3). By the same token, the cognitive area of the simple lens has the right to call for the cognitive area calculation of an optical area or assembly constraints but it does not know that it can be called upon by the objects such as doublets, triplets, or telescopic systems.

In summary, the interconnections within the KBOSD is as follows. Each object contains its own function, composition and path to look for what it needs. In exchange, the object is always readily available for all objects calling on it.

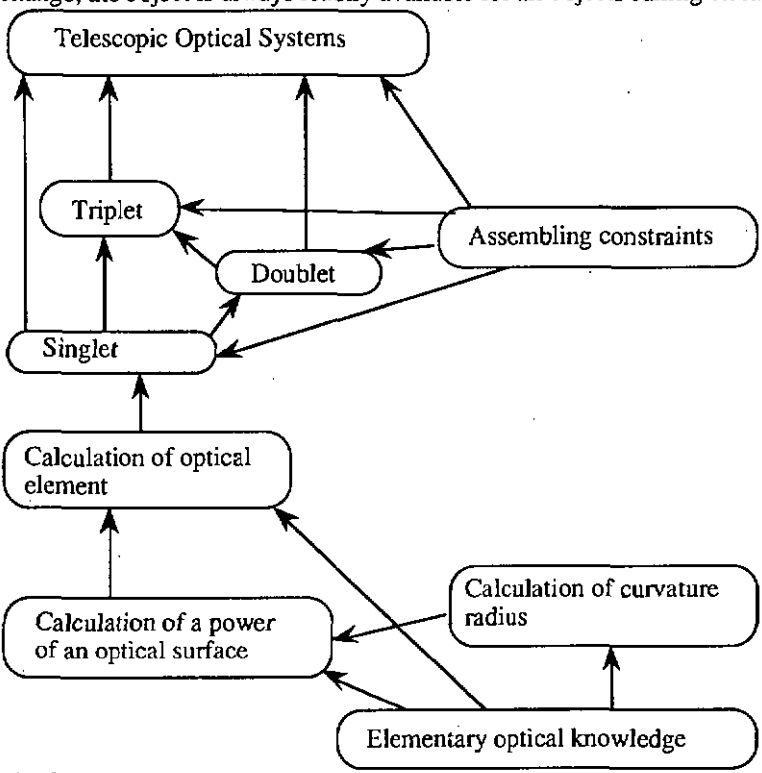


Fig. 3.5.3 Example of Simplified Interconnections Model

Example of Interconnections:

kepler_telescope (Transversal_Magnification, Longueur) :-
relation (Length, Optical_Power1, Optical_Power2),
three_different_wavelength(L1, L2, L3),
entrance_optical_system(Optical_Power1),
exit_optical_system(Optical_Power2).

This supposes that one is going to generate a Kepler type telescope used at three wavelengths. The Kepler telescope block calls on the doublet block and the triplet block, which has recourse in its turn to a doublet assembly block and a triplet assembly block.
This assembly block calls on a simple lens block, a singlet, that solicits the optical surface function. Thus, one sees that during the assembly of an optical system, it is repeatedly decomposed into simple elements that satisfy its physical and geometrical conditions.

3.6 The Reasoning of the Computer to Solve an Optical Problem

The KBOSD functions with a **backward-chaining inference engine** (for more details see chapter 2). Once a goal is selected, the rules concerning this goal will be activated. This goal is then divided into sub-goals introduced by the conditions of the rule and one attempts to prove them. One fixes the hypothesis used by the rule of the premise-conclusion to go back to the facts in accordance with this hypotheses.

The role of the inference engine in the KBOSD permits [Ref. 21]:

- the asking of questions
- relating the facts and the rules
- possibly unifying the clauses among themselves
- possibly determining values
- always finding a solution (sending "no solution available" is a solution).
- backtracking to find all solutions

To determine an optical system, our KBOSD begins by (see figure 3.6).

- calculating the opto-geometrical parameters of the optical system.
- (possibly) testing the compatibility of these parameters
- recognizing the conditions and the context of utilization of the optical system.
- identifying the optical system
- assembling the different components of the optical system

- outputing the data of the optical system

If one of these goals fails, the inference engine backtracks while identifying the cause of the failure and then looks for another alternative to satisfy this goal.

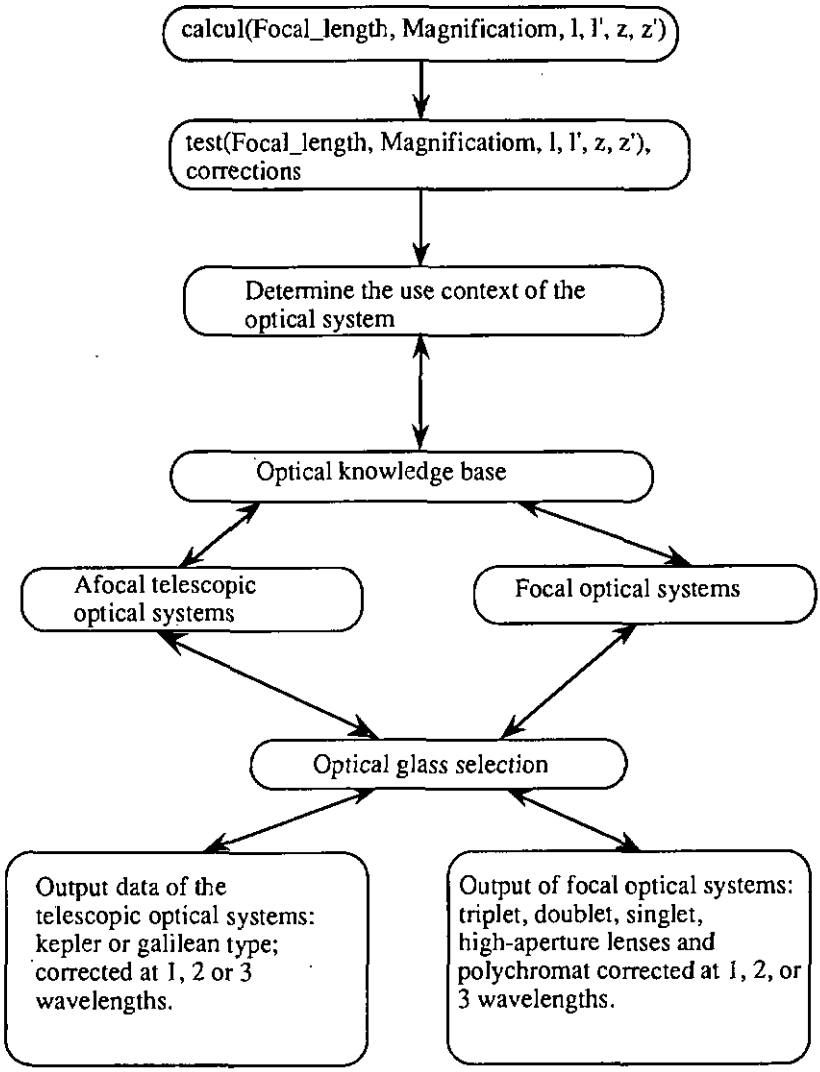


Fig. 3.6 Reasoning of the Computer to Determine an Optical System

Determining a goal to calculate an optical system calls upon a cognitive area. This cognitive area, in order to satisfy its needs, calls on these optical sub-systems until an elementary cognitive area, such as a cognitive cell is reached. The cognitive cell is an elementary knowledge area. Here is an example: The calculation of the curvature radius of an optical surface aside from its optical power as displayed in Fig. 3.5.2. First, we solve the sub-problem , then the problem, followed by the solution of the block, and finally the block is manipulated.

3.7 Conclusions

The representation and the organization of the optical knowledge structure of independent blocks equipped with interconnections to execute their functions has many advantages:

- each block is independent
- a block can be modified without touching the rest of the KBOSD
- new knowledge blocks can be added in case of the acquisition of additional knowledge
- blocks can be manipulated to compose new optical systems
- it tests the exactitude and the coherence of new clauses while inserting them into the knowledge base in the form of blocks
- it avoids useless calculations: it uses the power of reasoning, and not the power of calculation.

4 Optical Systems Generated by the Knowledge Based Optical Systems Design

4.0 Introduction

4.I Data and Analysis of a Doublet Lens

4.2 Data and Analysis of a Singlet Lens

4.0 Introduction

This chapter includes several examples generated by the Knowledge Based Optical Systems Design (KBOSD) that have been chosen to illustrate the capability of the KBOSD and some situations that arise in the optical design. The optical systems reported in this chapter are entirely produced by the KBOSD and optimized by the design and optimization program Sigma-PC [Ref. 17]. We have introduced some constraints on glass, opto-geometrical and physical properties of the optical system, then the KBOSD generated the optical start system.

Examples of some of the constraints on the glass:

- optical properties like dispersion, transmittance...
- chemical properties like dimming, staining, latent scratch, acid resistivity...
- mechanical properties like knoop hardness, abrasion factor...
- thermal properties like transformation temperature, thermal expansion...

The optical start systems generated by the KBOSD are optimized and analyzed with the Lens Design Program Sigma PC. Some of these optical systems are reported below as examples.

4.1 Data and Analysis of a Doublet Lens

This doublet lens is designed at the wavelength of 588 nm, it has a front focal length of 100.0 mm, a back focal length of 97.18 mm, a clear diameter of 20 mm and an entrance pupil diameter of 19 mm. The object is supposed to be at infinity. The numerical aperture is 0.095.

This doublet lens is obtained as the result of the optimization of the start system doublet produced by KBOSD and displayed at section 4.1.0.

It is useful to compare this doublet lens produced by the KBOSD with the doublet given in [Ref. 5 page 172].
This doublet lens consists of four spherical surfaces (TYPE S), the last surface is the image plane. As materials BAF3 and SF15 are used. The refractive indices (Index1, Index2, Index3) for the wavelengths 588 nm, 656 nm and 488 nm respectively, are given.

4.1.1.0 Data of the KBOSD Doublet Lens Start System

The doublet lens shown in Fig. 4.1.1.2 is proposed by the KBOSD as start system doublet lens.

If one enters the following predicate, calcul(100.0,Magnification, L, L', Z, Z'), to KBOSD, then, you will get the data of this doublet lens is obtained. The predicate calcul(100.0, Magnification, L, L', Z, Z') is explained in chaptres 2 and 3.

Displayed below, are the data of the doublet lens before optimization. It has a focal length of 100 mm and should be used at the wavelength 656.3 nm. It is supposed to be corrected for the chromatic aberrations at the wavelengths 656.3 nm and 587.6 nm. During the optimization process, the curvature radii, the thickness 1 and thickness 2 and the separation are variables. The type of optical glasses used for the singlet lens must be fixed during the optimisation.

EFL= 100.0000 [mm]
KBOSD Start System Doublet Lens File: KBOSD Start System Doublet Lens

WL1 = 0.000588 [mm]
WL2 = 0.000656 [mm]
WL3 = 0.000488 [mm]

#	Type	Curve	Sep.	Index1	Index2	Index3	Dispn.	Clr. Rad	Glass
1	S	0.024135	0.000	1.000000	1.000000	1.000000	0.000000	10.00	
2	S	-0.024135	4.143	1.582668	1.578932	1.591255	0.012323	10.00	BAF3
3	S	-0.025936	0.000	1.000000	1.000000	1.000000	0.000000	10.00	
4	S	0.000000	2.000	1.698946	1.692210	1.715041	0.022831	10.00	SF15

4.1.1.1 Data of the Doublet Lens

After five iterations by the optimization program, the merit function converges to zero, the data of the start system doublet lens displayed above (in section 4.1.0) are optimized to the data below.

EFL= 100.0000 [mm]
KBOSD Doublet Lens File: Doublet Lens

WL1 = 0.000588 [mm]
WL2 = 0.000656 [mm]
WL3 = 0.000488 [mm]

#	Type	Curve	Sep.	Index1	Index2	Index3	Dispn.	Clr. Rad	Glass
1	S	0.015399	0.000	1.000000	1.000000	1.000000	0.000000	10.00	
2	S	-0.028406	3.698	1.582668	1.578932	1.591255	0.012323	10.00	BAF3
3	S	-0.027862	0.000	1.000000	1.000000	1.000000	0.000000	10.00	
4	S	-0.005725	2.000	1.698946	1.692210	1.715041	0.022831	10.00	SF15
5	S	0.000000	97.184	1.000000	1.000000	1.000000	0.000000	1.77	

The table below contains the doublet lens information in term of radius, separation between surfaces, clear diameter and material. If a surface is plane, its radius is infinity. All the data are in mm.

EFL= 100.0000 [mm]
KBOSD Doublet Lens File: **Doublet Lens**

Radius[mm]	Sep. [mm]	Clear diameter	Material
64.939		20.00	
	3.698		BAF3
-35.204		20.00	
	0.000		Air
-35.892		20.00	
	2.000		SF15
-174.687		20.00	
	97.184		Air
Image Plane		3.54	

4.1.1.2 Drawing of the Doublet Lens

Displayed below is the drawing of the KBOSD doublet lens after optimization.

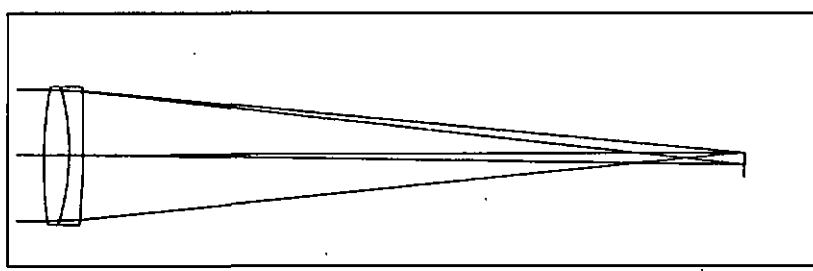


Fig. 4.1.1.2 Drawing of the KBOSD Doublet Lens after OPtimization

4.1.1.3 Doublet lens(Kingslake) from the Literaure

The example displayed below, is a doublet lens from the optical design literature, [Ref. 5, page 172]. It is optimized and analyzed according to M. Kidger [Ref. 17, page 1.3]. This doublet lens is reported in order to give some comparative ideas on the quality of the doublet lens produced by the KBOSD.
Note that, This doublet is analyzed at the same scale as the doublet lens produced by the KBOSD.

4.1.1.4 Data of the Kingslake Doublet

Displayed below is the data of the Kingslake doublet lens. This doublet is composed of two single lenses, an asymetric biconvex singlet and a plano-concave singlet.

EFL= 103.6652 [mm]
Kingslake Doublet Page. 172 File: Kingslake Doublet Page. 172

WL1 = 0.000588 [mm]
WL2 = 0.000656 [mm]
WL3 = 0.000488 [mm]

#	Type	Curve	Sep.	Index1	Index2	Index3	Dispn.	Clr. Rad	Glass
1	S	0.015090	0.000	1.000000	1.000000	1.000000	0.000000	10.00	
2	S	-0.022460	3.200	1.563838	1.561010	1.570287	0.009276	10.00	SK11
4	S	-0.005235	1.500	1.666796	1.660900	1.681110	0.020210	10.00	SF19
5	S	0.000000	101.279	1.000000	1.000000	1.000000	0.000000	10.00	

4.1.1.6 Analysis of the Kingslake Doublet

In this section one reports on the analysis of the Kingslake doublet.

4.1.1.6.1 Transverse Ray Aberrations of the Kingslake Doublet

Displayed below is the transverse ray aberrations of the Kinsglake doublet lens [Ref. 5, page 172].

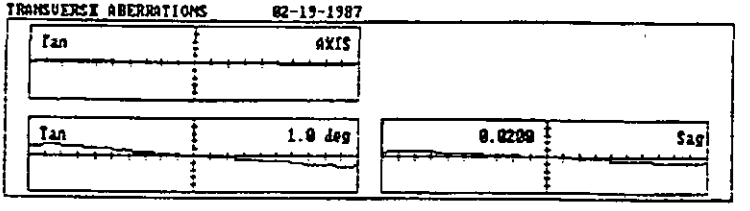


Fig. 4.1.1.6.1 Transverse Ray Aberrations of the Kingslake Doublet

4.1.1.6.2 Spot Diagrams of the Kingslake Doublet

Displayed below are the spot diagrams of the Kinsglake doublet lens [Ref. 5, page 172].

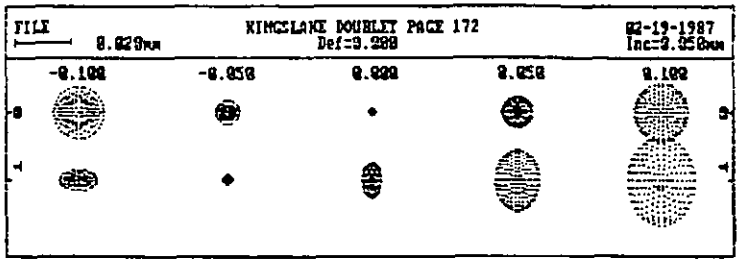


Fig. 4.1.1.6.2 Spot Diagrams of the Kingslake Doublet

4.1.1.6.3 Modulation Transfer Function Analysis of the Kingslake Doublet

Displayedd below is the modulated optical transfer function of this Kingslake doublet lens.

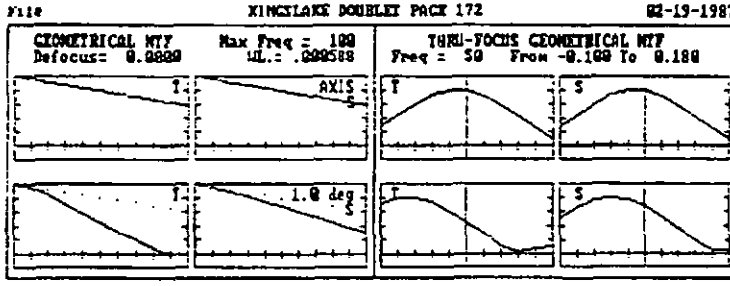


Fig. 4.1.1.6.3 Modulation Transfer Function Analysis of the Kingslake Doublet

4.1.2 Analysis of the Doublet Lens

This analysis contains the third-order aberrations or Seidel aberrations, the transverse ray aberrations, the optical path difference curves, the spot diagrams, the geometric transfer function and the diffraction transfer function [Ref. 4].

4.1.2.1 Third-Order Aberrations of the Doublet Lens

In this section we report the Seidel (third-order) aberrations [Ref. 4]. The Lagrange invariant is n.u.h = n'.u'.h'.

Where n, n' are the refractive indices in the object space and image space, u and u' are the paraxial angles, h and h ' are respectively object and image height.

The meanings of the variables used are explained in the table below:

H	paraxial marginal ray height
U	paraxial marginal ray angle
A	is n.i (i angle of incidence for the marginal ray)
ABAR	is n.ibar (ibar = angle of incidence for the chief ray)
HBAR	paraxial chief ray height
UBAR	paraxial chief ray angle
SpherAb.	Seidel spherical aberration
Coma	Seidel coma
Astig.	Seidel astigmatism
FCurv.	Seidel field curvature (Petzval sum = FCurv./H^2)
Distort.	Seidel distortion
LChroma.	Seidel longitudinal chromatic aberration
TChroma.	Seidel transverse (lateral) chromatic aberration or lateral color

A complete list of symbols is located at the begining of this text. The first table give the values of the parameters (H, U, HBAR, UBAR, D(U/N), A and ABAR) at each surfaces.

KBOSD Doublet Lens File: Doublet Lens

Lagrange Invariant = -0.1658

H	U	HBAR	UBAR	D(U/N)	A	ABAR
9.50000	0.00000	0.00000	0.01746	-0.03403	0.14629	0.01746
9.30083	-0.05386	0.04079	0.01103	-0.20515	-0.50338	0.01562
9.30083	-0.23918	0.04079	0.01678	0.21907	-0.49832	0.01564
9.23248	0.03417	0.06147	0.01034	-0.07489	-0.14785	0.01698
	-0.09500			0.01733		

SpherAb.	Coma	Astig.	FCurv.	Distort.	LChroma.	TChroma.
0.006919	0.000826	0.000098	0.000156	0.000030	0.010821	0.001291
0.483497	-0.015004	0.000466	0.000288	-0.000023	0.036454	-0.001131
-0.505952	0.015883	-0.000499	-0.000315	0.000026	-0.062282	0.001955
0.015114	-0.001735	0.000199	0.000065	-0.000030	0.018344	-0.002106
-0.000423	-0.000031	0.000265	0.000193	0.000002	0.003336	0.000009

The second table shows the surface contributions to the Seidel aberrations. The final line shows the sum of the individual surface contributions. Observe the values of SpherAb., Coma, Astig., FCurv., Distort. and TChroma. in the final line, they indicates that this doublet has very small third-order aberrations.

4.1.2.1.0 Third-Order Aberrations of the Doublet Lens Start System

Displayed below are the third-order aberrations of the doublet lens start system before the optimization.

Note the low values of the third-order aberrations of this doublet lens start system. The longitudinal and transversal chromatic aberrations displayed in the last line of this table as well as the distorsion are zero. It is a good start system for optimization.

KBOSD Doublet Lens Start System File doublet 00:39:39 06-12-1991

Lagrange invariant = -0.1658

H	U	HBAR	UBAR	D(U/N)	A	ABAR
9.50000	0.00000	0.00000	0.01746	-0.05334	0.22929	0.01746
9.16741	-0.08441	0.04345	0.01103	-0.20919	-0.48379	0.01580
9.16741	-0.26252	0.04345	0.01684	0.22915	-0.50030	0.01572
9.05400	-0.05670	0.06421	0.01038	-0.06296	-0.09633	0.01763
	-0.09633		0.01763			

SpherAb.	Coma	Astig.	FCurv.	Distort.	LChroma.	TChroma.
0.026639	0.002028	0.000154	0.000244	0.000030	0.010821	0.001291
0.448840	0.014654	0.000478	0.000244	-0.000024	0.036454	0.001131
-0.525802	0.016518	-0.000519	-0.000293	0.000026	-0.062282	0.001955
0.005290	-0.000968	0.000177	0.000000	-0.000032	0.018344	-0.002106
-0.045033	0.002924	0.000291	0.000195	-0.000000	0.000000	0.000009

4.1.2.2 Transverse Ray Aberrations of the Doublet Lens

In this section we report plots of transverse ray aberration as a function of aperture, as shown below.

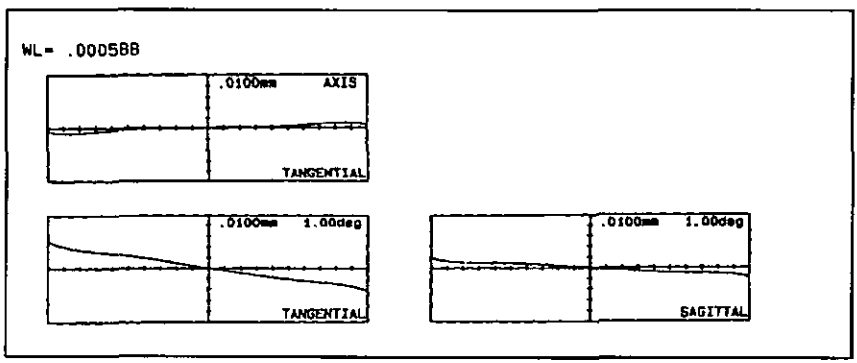


Fig.4.1.2.2 Transverse Ray Aberrations of the Doublet Lens

The top left curve shows the transverse aberrations of rays in the tangential plane (or y, z plane), on-axis.
In the bottom the axial plot, we see the aberrations at 1º off-axis. On the left we have the tangential aberrations; the sagittal aberrations are shown on the right. The scale for the aberrations is 0.01 mm. The plots can be repeated for three wavelengths, in this case the transverse ray aberrations is plotted at 588 nm.

4.1.2.3 Optical Path Differences of the Doublet Lens

Below the Optical Path Difference (OPD), otherwise known as wave front aberration, is plotted.

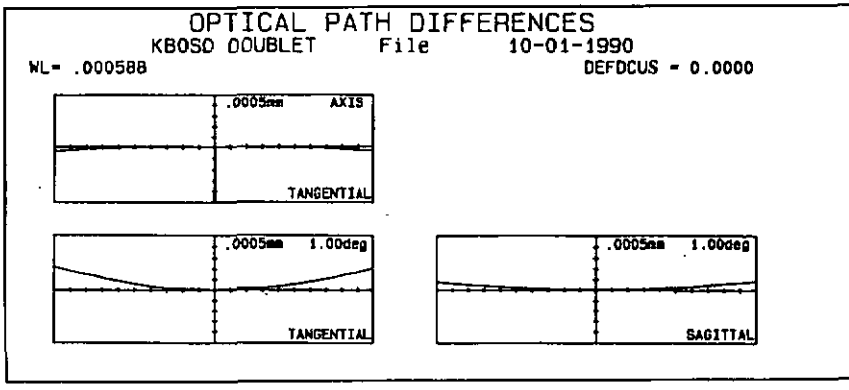


Fig. 4.1.2.3 Optical Path Differences of the Doublet Lens

This plot is for the wavelength 588 nm. The OPD scale is 0.0005 mm.

4.1.2.4 Spot Diagrams of the Doublet Lens

In this section we report a plot of the spot diagrams. The scale for the spot diagrams is indicated on the top left, it is the same as the scale for the transverse ray aberrations. The spot diagrams are plotted in five image positions, separated by a z-increment (Inc = 0.05 mm). The central set of spot diagram is in the focal position determined by a paraxial ray trace.

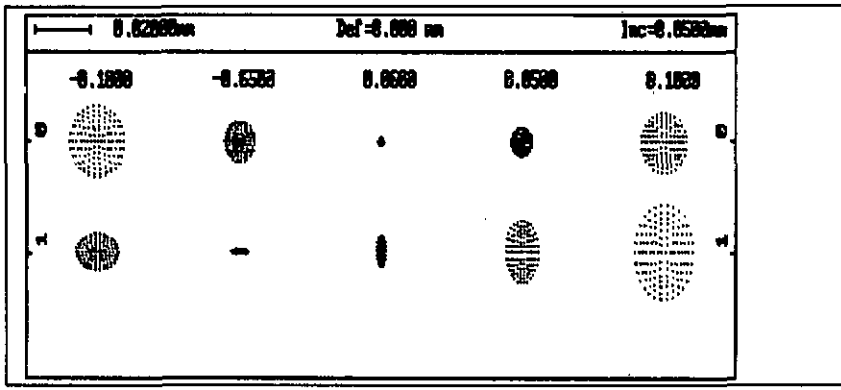


Fig. 4.1.2.4 Spot Diagrams of the Doublet Lens

The spot diagrams are shown at the field angele positions on axis and at 1º.

4.1.2.5 Modulation Transfer Function Analysis of the Doublet Lens

In this section one reports on the analysis of the KBOSD doublet lens.

4.1.2.5.1 Geometric MTF Table and Thru-focus Geometric MTF of the Doublet Lens

In this section one reports on the geometric MTF, the Thru-focus geometric MTF and the diffraction MTF.

Focal Plane GOTF Tables

KBOSD Doublet Lens File: Doublet Lens

Defocus = 0.0000 [mm]
Back focus = 97.184 [mm]
Wavelength = 0.0005876 [mm]

	Axis		1.00 Deg	
	T	S	T	S
Frequency				
10.00	0.962	0.962	0.954	0.961
20.00	0.924	0.924	0.892	0.921
30.00	0.885	0.885	0.817	0.879
40.00	0.845	0.845	0.732	0.835
50.00	0.805	0.805	0.641	0.790
60.00	0.765	0.765	0.546	0.745
70.00	0.725	0.725	0.453	0.699
80.00	0.685	0.685	0.362	0.652
90.00	0.645	0.645	0.278	0.606
100.00	0.606	0.606	0.202	0.561

If we compare this table with the diffraction MTF table, we observe that the geometric values differ by 0.02 of the diffraction values. Such differences indicate that the geometric result is inaccurate, the geometric MTF is an approximation of the diffraction MTF.

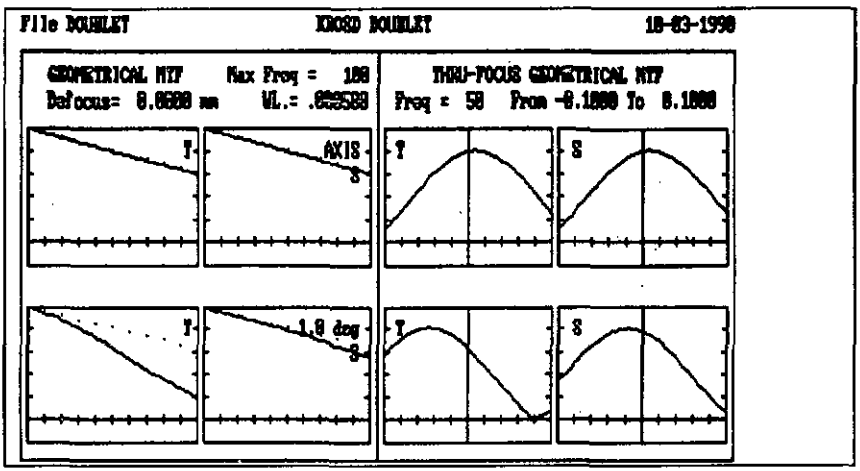


Fig.4.1.2.5.1 Geometric MTF and Thru-focus Geometric MTF of the Doublet Lens

On the left we have the conventional Geometric MTF curves, as a function of the spatial frequency, at each field position. In this case the diagrams are plotted at a spatial frequency of 100 lines/mm, at the wavelength 588 nm and at a defocus of zero mm.

On the right we have the thru-focus Geometric MTF, as a function of variation of the focal plane, at a fixed spatial frequency (Frequency = 50 lines/mm), over the same range of image positions as in the spot diagrams.

4.1.2.5.1.0 Geometric MTF and Thru-focus Geometric MTF of the Doublet Lens Start System

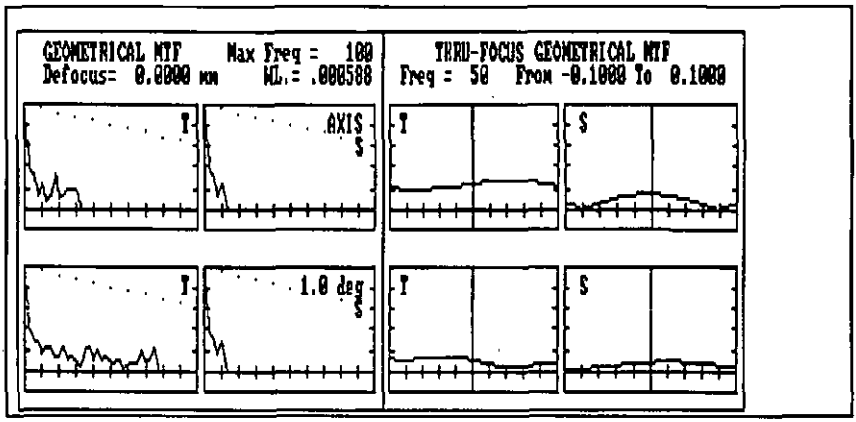


Fig.4.1.2.5.1.0 Geometric MTF and Thru-focus Geometric MTF of the Doublet Lens Start System before Optimization

4.1.2.5.2 Diffraction MTF of the Doublet Lens

KBOSD Doublet Lens File: Doublet Lens

Defocus = 0.000 [mm]
Back focus = 97.184 [mm]
Wavelength = 0.000588 [mm]

	Axis	1.00 deg	
Frequency [l/mm]		S	T
10.000	0.960	0.960	0.952
20.000	0.920	0.918	0.892
30.000	0.879	0.875	0.824
40.000	0.839	0.831	0.752
50.000	0.798	0.787	0.679
60.000	0.758	0.743	0.609
70.000	0.719	0.700	0.542
80.000	0.680	0.658	0.481
90.000	0.641	0.616	0.426
100.000	0.603	0.576	0.378

4.2 Data and Analysis of a Singlet Lens

This singlet lens is designed at a wavelength of 656 nm and has a front focal length of 110.0 mm and a back focal length of 108.12 mm. It has a clear di-

ameter of 20 mm and an entrance pupil diameter of 18 mm. The object is supposed to be at infinity: The numerical aperture is 0.082.

4.2.0 Data of the Singlet Lens Start System

The singlet lens shown below is produced by the KBOSD as a start system. It has a focal length of 110 mm and should be used at the wavelength 656 nm.
During the optimisation process, the curvature radii are variables. The type of optical glass used for the singlet lens must be fixed during the optimization.

EFL= 110.0001 [mm]
KBOSD Singlet Lens Start System File: KBOSD Singlet Lens Start System

WL1 = 0.000656 [mm]

#	Type	Curve	Sep.	Index1	Index2	Index3	Dispn.	Clr. Rad	Glass
1	S	0.013006	0.000	1.000000	1.000000	1.000000	0.000000	10.00	
2	S	0.000000	2.653	1.692210	1.692209	1.692211	0.000002	10.00	SF15

After 5 iterations, the merit function decreased, but not to zero. The data displayed in section 4.2.1.1 are results of the optimization.

4.2.1.1 Data of the Singlet Lens

Displayed below is the data(in terms of surfaces curvature) the optimized KBOSD singlet lens.

EFL= 110.000 [mm]
KBOSD Singlet Lens File: Singlet Lens

WL1 = 0.000656 [mm]

#	Type	Curve	Sep.	Index1	Index2	Index3	Dispn.	Clr. Rad	Glass
1	S	0.013227	0.000	1.000000	1.000000	1.000000	0.000000	10.00	
2	S	0.000095	2.600	1.692210	1.692209	1.692211	0.000002	10.00	SF15
3	S	0.000000	108.123	1.000000	1.000000	1.000000	0.000000	3.91	

Displayed below is the data(in terms of curvature radii) of the optimized KBOSD singlet lens.

EFL= 110.000 [mm]
KBOSD Singlet Lens File: Singlet Lens

Radius[mm]	Sep. [mm]	Clear diameter	Material
75.603		20.00	
	2.600		SF15
10510.539		20.00	
	108.123		Air
Image Plane		7.82	

4.2.1.2 Drawing of the Singlet Lens

Displayed below is the drawing of the optimized KBOSD singlet lens. Such as a singlet lens has the minimum of aberrations(heuristic).

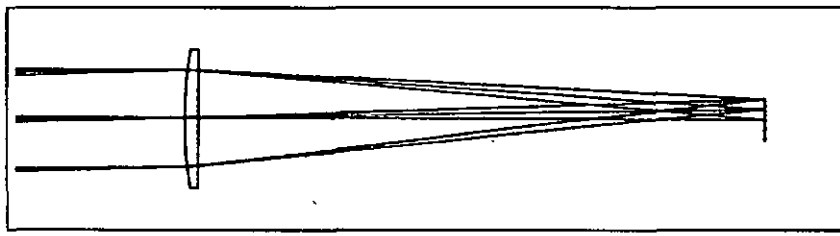


Fig. 4.2.1.2 Drawing of the KBOSD Singlet Lens after Optimization

4.2.2 Analysis of the Singlet Lens

In this section one reports on the analysis of the optimized KBOSD singlet lens.

4.2.2.1 Third-Order Aberrations of the Singlet Lens

Displayed below are the third order aberrations of the KBOSD singlet lens after optimization.

KBOSD Singlet Lens File: Singlet Lens

Lagrange Invariant = -0.3143

H	U	HBAR	UBAR	D(U/N)	A	ABAR
9.00000	0.00000	0.00000	0.03492	-0.02878	0.11904	0.03492
8.87339	-0.04870	0.05365	0.02064	-0.05304	-0.08097	0.03493
	-0.08182		0.03492			

SpherAb.	Coma	Astig.	PCurv.	Distort.	LChroma.	TChroma.
0.003670	0.001077	0.000316	0.000534	0.000249	0.000001	0.000000
0.003086	-0.001331	0.000574	-0.000004	-0.000246	0.000001	-0.000000
0.006756	-0.000255	0.000890	0.000531	0.000003	0.000002	0.000000

The summa of the third order aberrations is displayed at the last line.

4.2.2.2 Transverse Ray Aberrations of the Singlet Lens

Displayed below are the transverse ray aberrations of the KBOSD singlet lens after optimization. These transverse ray aberrations are plotted at the image plane under the following parameters:

- wavelength 656 nm
- field angle positions axis, 1º and 2º
- aberrations scale 50 µm
- sagittal and tangential planes.

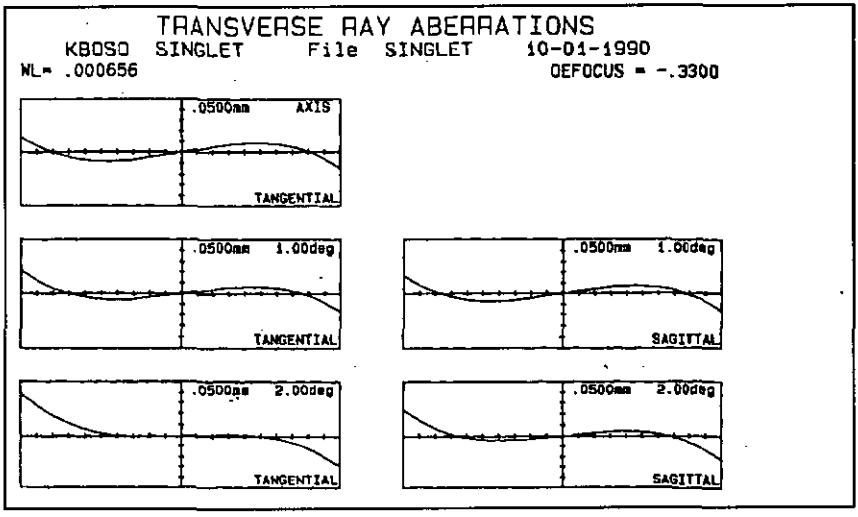


Fig.4.2.2.2 Transverse Ray Aberrations of the KBOSD Singlet Lens after Optimization

4.2.2.2.0 Transverse Ray Aberrations of the Singlet Lens Start System

Displayed below is the transverse ray aberrations of the KBOSD singlet lens before the optimization. This transverse ray aberrations are plotted under the same parameters as in section 4.2.2.2 above.

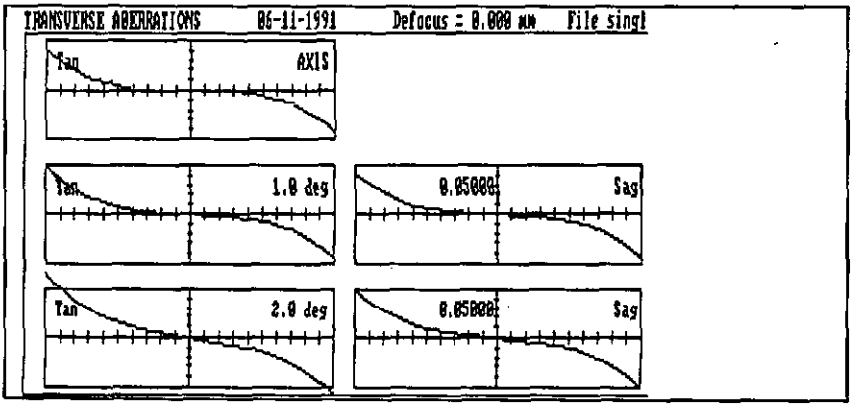


Fig.4.2.2.2.0 Transverse Ray Aberrations of the Singlet Lens Start System

4.2.2.3 Optical Path Differences of the Singlet Lens

Displayed below is the optical path differences of the KBOSD singlet lens after optimization. These optical path differences are plotted under the following parameters:

- wavelength 656 nm
- scale 500 nm
- field angle positions axis, 1 º and 2 º
- sagittal and tangential planes
- defocus of -0.33 mm.

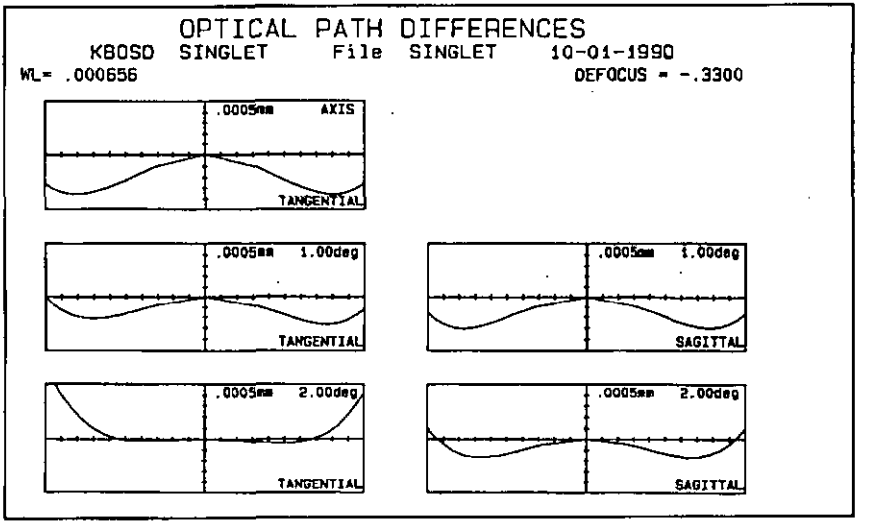


Fig. 4.2.2.3 Optical Path Differences of the Singlet Lens

4.2.2.4 Spot Diagrams of the Singlet Lens

Displayed below are the spot diagrams of the KBOSD singlet lens after optimization. These spot diagrams are plotted under the following parameters:

- wavelength 656 nm
- scale 50 μm
- field angle positions axis, 1º and 2 º.

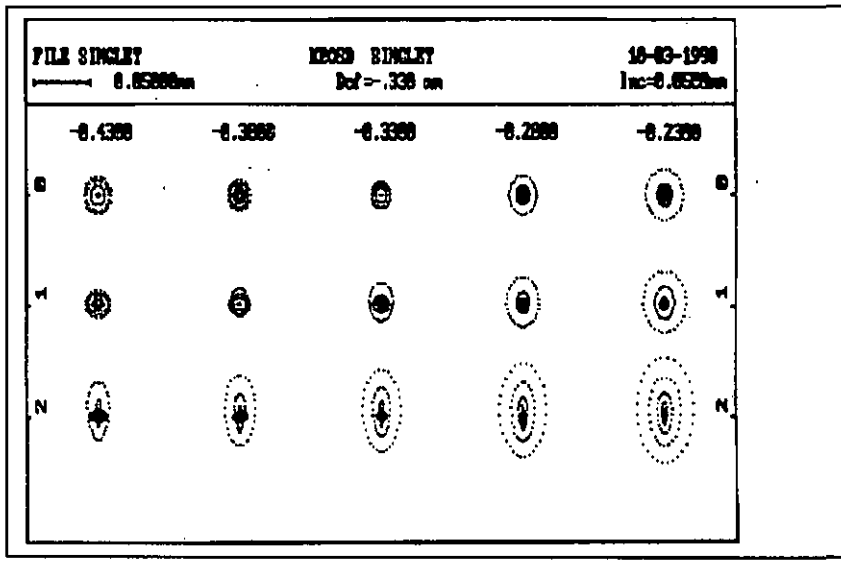


Fig. 4.2.2.4 Spot Diagrams of the Singlet Lens

4.2.2.4.0 Spot Dingrams of the Singlet Lens Start System

Displayed below are the spot diagrams of the KBOSD singlet lens before the optimization. These spot diagrams are plotted under the same parameters as in section 4.2.2.4.

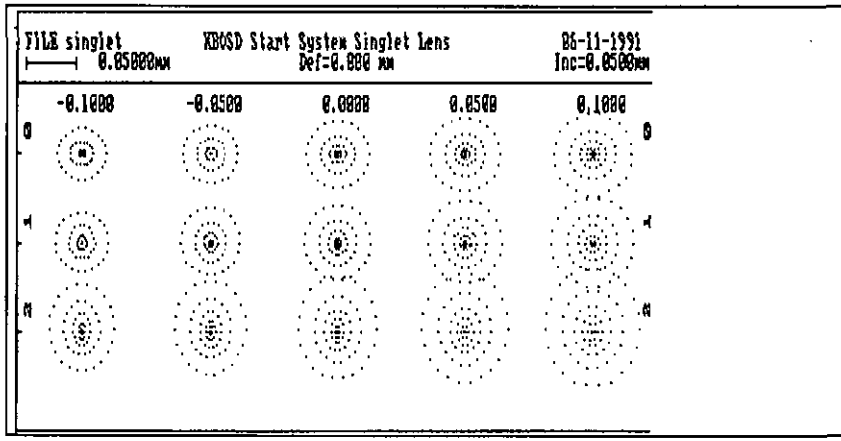


Fig. 4.2.2.4.0 Spot Diagrams of the Singlet Lens Start System before Optimization

4.2.2.5.0 Focal Plane GOTF Tables of the Singlet Lens Start System

Displayed below, is the result of GOTF analysis of the singlet lens start system before optimization. The result of the GOTF after the optimization are dispalayed in the section 4.2.2.5 below.

FOCAL PLANE GOTF TABLES
KBOSD Start System Singlet Lens File KBOSD Start System Singlet Lens 12.06.1991
defocus = 0.000 [mm]
back focus = 109.5 [mm]
WL = 0.0006563 [mm]

	Axis		1.00 Deg		2.00 Deg	
	T	S	T	S	T	S
Frequency [l/mm]						
5.00	0.889	0.889	0.858	0.876	0.743	0.832
10.00	0.656	0.656	0.570	0.618	0.314	0.501
15.00	0.421	0.421	0.319	0.373	0.127	0.246
20.00	0.278	0.278	0.211	0.243	0.125	0.172
25.00	0.243	0.243	0.209	0.226	0.098	0.189
30.00	0.258	0.258	0.211	0.241	0.050	0.166
35.00	0.252	0.252	0.162	0.216	0.042	0.091
40.00	0.200	0.200	0.100	0.148	0.064	0.046
45.00	0.132	0.132	.083	0.091	0.029	0.078
50.00	0.099	.0.099	0.106	0.092	0.064	0.124

4.2.2.5 Focal Plane GOTF Tables of the Singlet Lens

Displayed below are the geometric optical transfer function at the focal plane of the KBOSD singlet lens after optimization.

KBOSD Singlet Lens File: Singlet Lens

Defocus =-0.330 [mm]
Back focus = 108.123 [mm]
Wavelength = 0.000656 [mm]

	Axis		1.00 Deg		2.00 Deg	
	T	S	T	S	T	S
Frequency [l/mm]						
5.00	0.968	0.968	0.965	0.968	0.922	0.959
10.00	0.917	0.917	0.904	0.914	0.760	0.881
15.00	0.848	0.847	0.824	0.842	0.577	0.781
20.00	0.765	0.765	0.732	0.757	0.436	0.672
25.00	0.673	0.672	0.636	0.664	0.360	0.569
30.00	0.575	0.574	0.543	0.567	0.334	0.483
35.00	0.476	0.475	0.459	0.473	0.325	0.421
40.00	0.380	0.378	0.389	0.385	0.310	0.381
45.00	0.290	0.289	0.333	0.306	0.283	0.359
50.00	0.210	0.208	0.289	0.238	0.255	0.345

4.2.2.6 Geometrical MTF and Thru-focus Geometrical MTF of the Singlet Lens

Displayed below are the geometrical modulate transfer function and the thru-focus geometrical modulated transfer function of the KBOSD singlet after optimization. These diagrams are plotted under the following parameters:

- spatial frequency 50 lines/mm
- field angle positions axis, 1º and 2º
- sagittal and tangential planes
- thru-focus spatial frequency 25 lines/mm

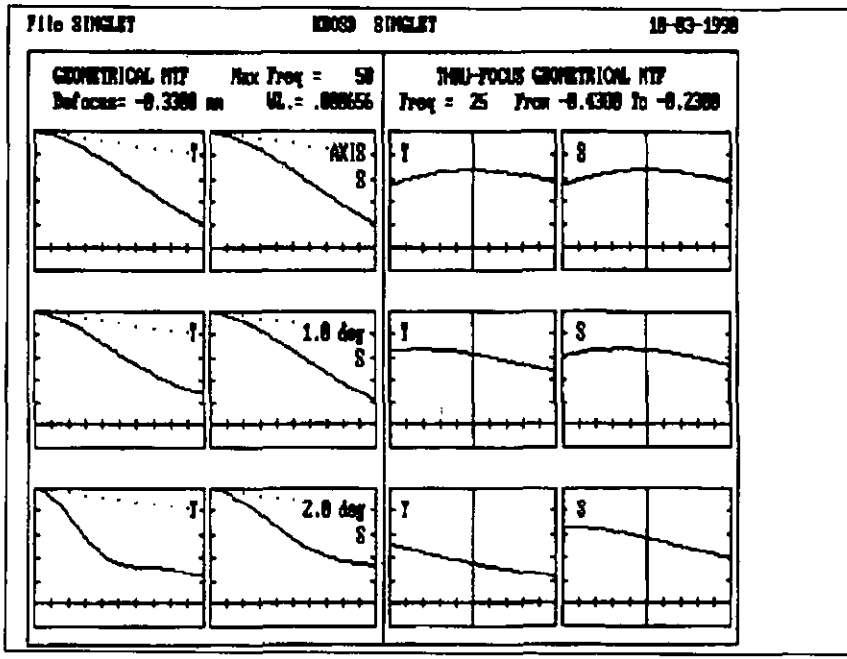


Fig. 4.2.2.6 Geometrical MTF and Thru-focus Geometrical MTF of the Singlet Lens

4.3 Data and Analysis of a Two Reversed Doublet

This two symmetrical reversed doublet is designed at a wavelength of 588 nm and is used as a high-aperture lens. It has a EFL = 97.93 mm, a

clear diameter of 36 mm and an entrance pupil diameter of 31 mm. It gives the image of an object of 6 mm height and situated 190 mm to the left of the first surface. The magnification is equal to unity. Such an optical system is powerful if it is used at a magnification near unity. Because of symmetry, many third order aberrations will be reduced, see for instance Coma, Distort., TChroma. in the third-order aberration table below.

4.3.1.0 Data of the Two Reversed Doublet Start System

Displayed below is the the data of the reversed doublet as proposed by the KBOSD.

EFL = 94.722 [mm]
KBOSD REVERSED Doublet Lens File: REVDOUB

Object Distance = 190.0005 [mm]

WL1 = 0.000588 [mm]
WL2 = 0.000656 [mm]
WL3 = 0.000656 [mm]

#	Type	Curve	Sep.	Index1	Index2	Index3	Dispn.	Clr. Rad	Glass
1	S	Plane	0.000	1.000000	1.000000	1.000000	0.000000	18.00	
2	S	0.020449	2.000	1.648306	1.642708	1.661527	0.018819	18.00	SF12
3	S	0.015666	0.000	1.000000	1.000000	1.000000	0.000000	18.00	
4	S	-0.015666	7.500	1.582668	1.578932	1.591255	0.012323	18.00	BAF3
5	S	0.015666	0.000	1.000000	1.000000	1.000000	0.000000	18.00	
6	S	-0.015666	7.500	1.582668	1.578932	1.591255	0.012323	18.00	BAF3
7	S	-0.020449	0.000	1.000000	1.000000	1.000000	0.000000	18.00	
8	S	Plane	2.000	1.648306	1.642708	1.661527	0.018819	18.00	SF12

4.3.1.1 Data of the Two Reversed Doublet System

Displayed below is the data of the KBOSD reversed doublet after the optimization, this data are displayed in terms of curvatures of the surfaces of the lens.

EFL = 97.9305 [mm]
KBOSD REVERSED Doublet Lens File: REVDOUB

Object Distance = 190.0005 [mm]

WL1 = 0.000588 [mm]
WL2 = 0.000656 [mm]
WL3 = 0.000656 [mm]

#	Type	Curve	Sep.	Index1	Index2	Index3	Dispn.	Clr. Rad	Glass
1	S	-0.002976	0.000	1.000000	1.000000	1.000000	0.000000	18.00	
2	S	0.018643	2.000	1.648306	1.642708	1.661527	0.018819	18.00	SF12
3	S	0.018852	0.000	1.000000	1.000000	1.000000	0.000000	18.00	

4	S	-0.013598	7.500	1.582668	1.578932	1.591255	0.012323	18.00	BAF3
5	S	0.013598	0.000	1.000000	1.000000	1.000000	0.000000	18.00	
6	S	-0.018852	7.500	1.582668	1.578932	1.591255	0.012323	18.00	BAF3
7	S	-0.018643	0.000	1.000000	1.000000	1.000000	0.000000	18.00	
8	S	0.002976	2.000	1.648306	1.642708	1.661527	0.018819	18.00	SF12
9	S	0.000000	190.000	1.000000	1.000000	1.000000	0.000000	121.38	

Displayed below is the data of the KBOSD reversed doublet after the optimization, this data are displayed in terms of curvature radii of the lens.

EFL = 97.9305 [mm]
KBOSD REVERSED Doublet Lens File: REVDOUB

Object Distance = 190.0005 [mm]

Radius[mm]	Sep. [mm]	Clear diameter	Material
-335.979		36.00	
	2.000		SF12
53.640		36.00	
	0.000		Air
53.044		36.00	
	7.500		BAF3
-73.540		36.00	
	0.000		Air
73.540		36.00	
	7.500		BAF3
-53.044		36.00	
	0.000		Air
-53.640		36.00	
	2.000		SF12
335.979		36.00	
	190.000		Air
Image Plane		10.76	

4.3.2.2 Drawing of the Two Reversed Doublet

Displayed below is the drawing of the KBOSD reversed doublet after optimization. The data of this reversed doublet are displayed in section 4.3.2.1 above.

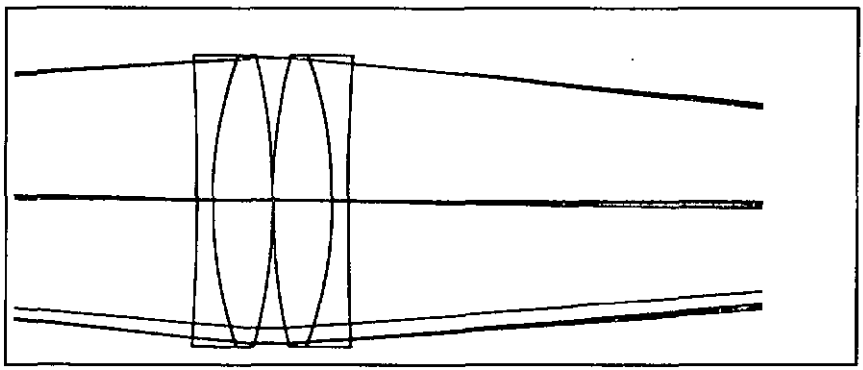


Fig. 4.3.2.2 Drawing of the Two Reversed Doublet after Optimization

4.3.2 Analysis of the Two Reversed Doublet

In this section one reports on the analysis of the two reversed doublet after optimization.

4.3.2.1 Third-Order Aberrations of the Two Reversed Doublet

Displayed below are the third order aberrations of the reversed doublet lens proposed by the KBOSD.

KBOSD REVERSED Doublet Lens File: REVDOUB

Object Distance = 190.0005 [mm]

Lagrange Invariant= -0.2400

H	U	HBAR	UBAR	D(U/N)	A	ABAR
-15.20004	-0.08000	0.08977	-0.01532	0.03976	-0.03476	-0.01558
-15.33270	-0.06633	0.07139	-0.00919	-0.25441	-0.58049	-0.01295
-15.33270	-0.29465	0.07139	-0.01428	0.24425	-0.58370	-0.01294
-15.93085	-0.07975	-0.00000	-0.00952	0.05039	0.21663	-0.01507
-15.93085	-0.00000	-0.00000	-0.01507	0.05039	-0.21663	-0.01507
-15.33270	0.07975	-0.07139	-0.00952	0.24425	0.58370	-0.01294
-15.33270	0.29465	-0.07139	-0.01428	-0.25441	0.58049	-0.01295
-15.20004	0.06633	-0.08977	-0.00919	0.03976	0.03476	-0.01558
	0.08000		-0.01532			

SpherAb.	Coma	Astig.	FCurv.	Distort.	LChroma.	TChroma.
0.000730	0.000327	0.000147	-0.000067	0.000036	0.006032	0.002705
-1.314437	-0.029323	-0.000654	-0.000422	-0.000024	-0.101620	-0.002267
1.275979	0.028276	0.000627	0.000400	0.000023	0.069684	0.001544
0.037673	-0.002620	0.000182	0.000288	-0.000033	0.026871	-0.001869
0.037673	0.002620	0.000182	0.000288	0.000033	0.026871	0.001869
1.275979	-0.028276	0.000627	0.000400	-0.000023	0.069684	-0.001544
1.314437	0.029323	-0.000654	-0.000422	0.000024	-0.101620	0.002267

0.000730 -0.000327 0.000147 -0.000067 -0.000036 0.006032 -0.002705

-0.000110 0.000000 0.000603 0.000397 0.000000 0.001933 -0.000000

Note the low values of the summa(last line) of these third order aberrations.

4.3.2.2 Transverse Ray Aberrations of the Two Reversed Doublet

Displayed below are the transverse ray aberrations at the image plane of the optimized KBOSD two reversed doublet at the wavelength 588 nm. These transverse ray aberrations are displayed at the objet positions axis, 1 mm, 2 mm and 3 mm respectivelly at the tangential and sagittal planes.

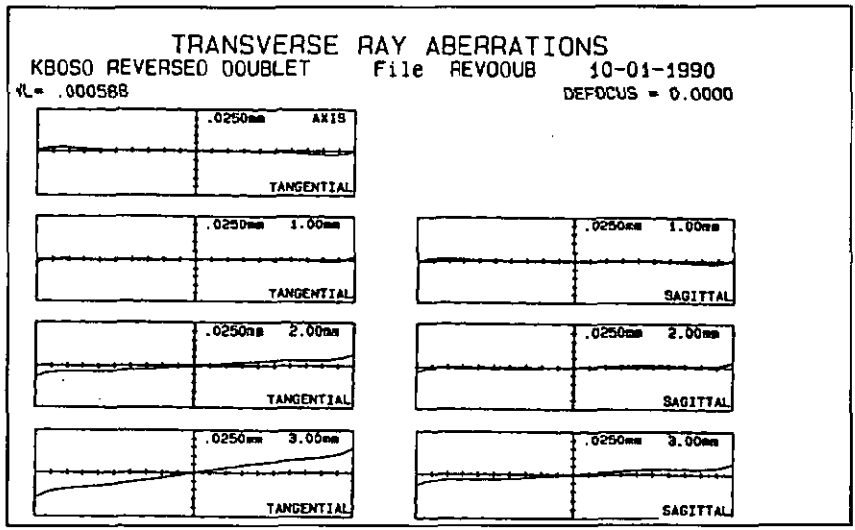


Fig. 4.3.2.2 Transverse Ray Aberrations of the Two Reversed Doublet

4.3.2.3 Optical Path Differences of the Two Reversed Doublet

Displayed below are the optical path differences aberrations, this aberrations are displayed at the wavelength 588 nm and at the scale 700 nm at the following object positions axis, 1 mm 2 mm and 3mm.

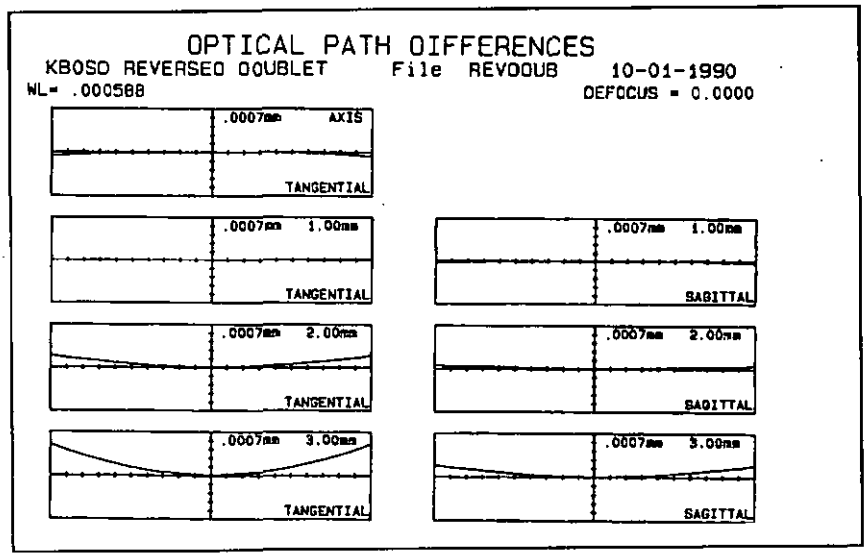


Fig. 4.3.2.3 Optical Path Differences of the Two Reversed Doublet

4.3.2.4 Spot Diagrams of the Two Reversed Doublet

Displayed below are the spot diagrams of the optimized KBOSD reversed doublet. This spot diagrams are plotted in the following conditions:

- scale 25 μm
- object positions axis, 1 mm, 2 mm and 3 mm
- wavelength 588 nm.

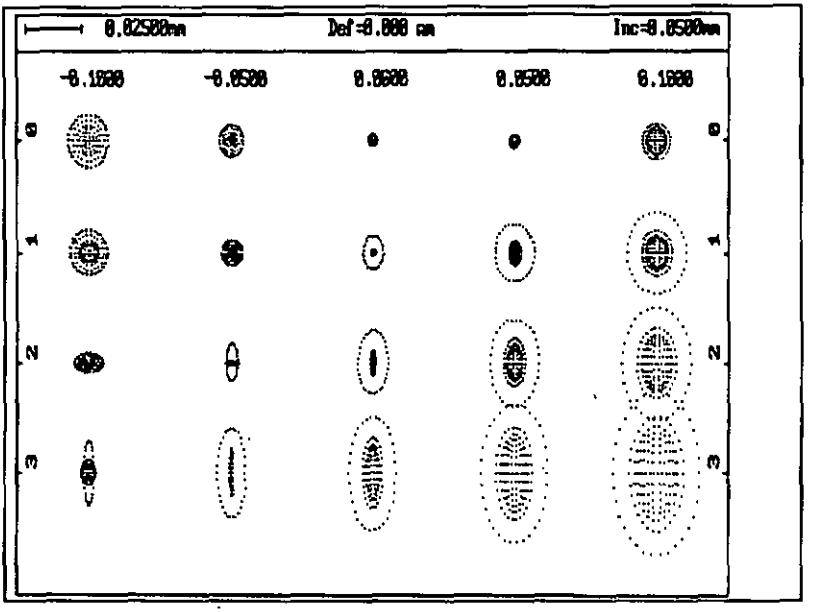


Fig. 4.3.2.4 Spot Diagrams of the Two Reversed Doublet

4.3.2.5 Focal Plane GOTF Tables of the Two Reversed Doublet

Displayed below are the geometric optical transfert function of the optimized KBOSD reversed doublet at the focal plane.

Focal Plane GOTF Tables
KBOSD REVERSED Doublet Lens File: REVDOUB

Defocus = 0.0000 [mm]
Back focus = 190.000 [mm]
Wavelength = 0.0005876 [mm]

	Axis		1.00 mm		2.00 mm		3.00 mm	
	T	S	T	S	T	S	T	S
Frequency [l/mm]								
10.00	0.952	0.952	0.947	0.949	0.924	0.944	0.831	0.923
20.00	0.900	0.900	0.884	0.889	0.802	0.871	0.520	0.800
30.00	0.847	0.847	0.814	0.824	0.659	0.788	0.240	0.655
40.00	0.792	0.792	0.740	0.755	0.521	0.701	0.083	0.515
50.00	0.736	0.736	0.666	0.685	0.408	0.618	0.016	0.399
60.00	0.679	0.679	0.595	0.615	0.325	0.542	0.030	0.315
70.00	0.622	0.622	0.530	0.548	0.269	0.478	0.072	0.258
80.00	0.566	0.566	0.472	0.485	0.228	0.426	0.080	0.218
90.00	0.511	0.511	0.422	0.428	0.190	0.387	0.039	0.180
100.00	0.458	0.458	0.380	0.376	0.146	0.357	0.016	0.137

4.3.2.6 Geometrical MTF and Thru-focus Geometrical MTF of the Two Reversed Doublet

Displayed below are the geometrical modulated transfert function and the thru-focus geometrical modulated transfert function of the optimized KBOSD reversed doublet. The data of the functions plotted below is displayed in section 4.3.2.5 above.

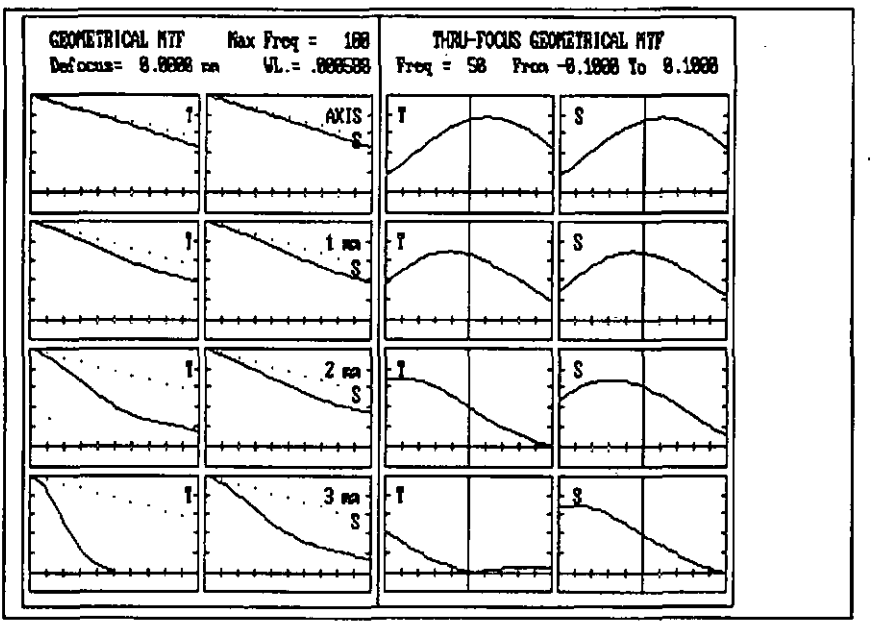


Fig. 4.3.2.6 Geometrical MTF and Thru-focus Geometrical MTF of the Two Reversed Doublet

4.4 Data and Analysis of the Triplet Lens

In this section we report the data of a triplet lens which is designed at a wavelength of 546 nm and assumed to be corrected for the three wavelength 546 nm, 656 nm and 486 nm. It has a front focal length of 100.0 mm, a back focal length of 92.47 mm, a diameter of 26 mm and an entrance pupil diameter of 25 mm. The numerical aperture is 0.125. The object is supposed to be at infinity.

4.4.1.0 Data of the Start System Triplet Lens

Reported below are the data of the KBOSD start system triplet lens. During optimization, the merit function converges rapidly to zero. The results of this triplet lens after optimization are displayed in section 4.4.1.1.

EFL = 100.0000 [mm]
KBOSD Start System Triplet Lens F=100 [mm] File: KBOSD Start System Triplet

WL1 = 0.000546 [mm]
WL2 = 0.000656 [mm]
WL3 = 0.000486 [mm]

#	Type	Curve	Sep.	Index1	Index2	Index3	Dispn.	Clr. Rad	Glass
1	S	0.000000	0.000	1.000000	1.000000	1.000000	0.000000	12.50	
2	S	0.021049	0.000	1.000000	1.000000	1.000000	0.000000	12.50	
3	S	-0.021049	4.840	1.558977	1.553827	1.563323	0.009496	12.50	BAK5
4	S	-0.039761	0.000	1.000000	1.000000	1.000000	0.000000	12.50	
5	S	0.039761	1.500	1.727353	1.714359	1.739040	0.024681	12.50	SF18
6	S	0.027197	0.000	1.000000	1.000000	1.000000	0.000000	12.50	
7	S	-0.027197	5.888	1.768606	1.753572	1.782302	0.028730	12.50	SF14

4.4.1.1 Data of the Triplet Lens

Displayed below, are the data of the KBOSD triplet lens after optimization.

EFL = 100.0000 [mm]
KBOSD Triplet Lens F=100 [mm] File: Triplet Lens

WL1 = 0.000546 [mm]
WL2 = 0.000656 [mm]
WL3 = 0.000486 [mm]

#	Type	Curve	Sep.	Index1	Index2	Index3	Dispn.	Clr. Rad	Glass
1	S	0.000000	0.000	1.000000	1.000000	1.000000	0.000000	12.50	
2	S	0.015087	0.000	1.000000	1.000000	1.000000	0.000000	12.50	
3	S	-0.020926	4.840	1.558977	1.553827	1.563323	0.009496	12.50	BAK5
4	S	-0.020413	0.000	1.000000	1.000000	1.000000	0.000000	12.50	
5	S	-0.004280	1.500	1.727353	1.714359	1.739040	0.024681	12.50	SF18
6	S	0.006932	0.000	1.000000	1.000000	1.000000	0.000000	12.50	
7	S	0.004989	5.804	1.768606	1.753572	1.782302	0.028730	12.50	SF14
8	S	0.00000092.471		1.000000	1.000000	1.000000	0.000000	1.75	

Displayed below is the data(in terms of radii) of the KBOSD triplet lens after optimization.

EFL = 100.0000 [mm]
KBOSD Triplet Lens F=100 [mm] File: Triplet Lens

Radius[mm]	Sep. [mm]	Clear diameter	Material
Plane		25.00	
	0.000		Air
66.282		25.00	

	4.840		BAK5
-47.787		25.00	
	0.000		Air
-48.988		25.00	
	1.500		SF18
-233.656		25.00	
	0.000		Air
144.267		25.00	
	5.804		SF14
200.440		25.00	
	92.471		Air
Image Plane		3.50	

4.4.1.2 Drawing of the Triplet Lens

Displayed below are the drawing of the KBOSD lens after optimization.

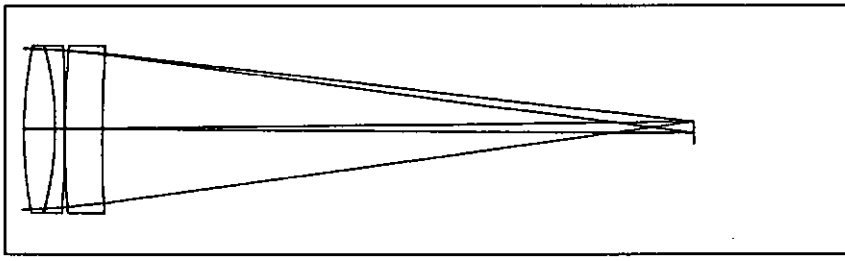


Fig. 4.4.1.2 Drawing of the Triplet Lens after Optimization

4.4.2 Analysis of the KBOSD Triplet Lens

In this section one reports on the analysis of the KBOSD triplet lens before and after optimization.

4.4.2.1.0 Third-Order Aberrations of the Triplet Lens Start System

Displayed below, are the third-order aberrations of the triplet lens start system before optimization.

KBOSD Start System Triplet Lens File triplet

Lagrange Invariant = -0.2182

H	U	HBAR	UBAR	D(U/N)	A	ABAR
12.50000	0.00000	0.00000	0.01746	0.00000	0.00000	0.01746
12.50000	0.00000	0.00000	0.01746	-0.06051	0.26311	0.01746

12.04339	-0.09434	0.05419	0.01120	-0.22826	-0.54228	0.01568
12.04339	-0.28878	0.05419	0.01682	-0.30872	-0.76763	0.01466
12.09508	0.03446	0.07016	0.01064	0.38937	0.89023	0.02320
12.09508	0.40932	0.07016	0.02041	-0.35929	0.73827	0.02232
12.60861	0.08848	0.13233	0.01071	-0.15711	-0.45000	0.01258
	-0.10708	0.01618				

SpherAb.	Coma	Astig.	FCurv.	Distort.	LChroma.	TChroma.
0.000000	0.000000	0.000000	0.000000	0.000000	0 . 000000	0 . 000000
0.052366	0.003474	0.000230	0.000359	0.000039	0.020033	0.001329
0.808395	-0.023370	0.000676	0.000359	-0.000030	0.039780	-0.001150
-2.190923	0.041849	-0.000799	-0.000797	0.000030	-0.132094	0.002523
-3.732261	-0.09727	-0.002535	-0.000797	-0.000087	-0.153848	-0.004010
2.368532	0.071613	0.002165	0.000563	0.000082	0.145055	0.004386
0.401143	-0.011216	0.000314	0.000563	-0.000024	0.092170	-0.002577
-2.292748	-0.014928	0.000050	0.000250	0.000011	0.011096	0.000501

4.4.2.1 Third-Order Aberrations of the Triplet Lens

Displayed below, are the third-order aberrations of the triplet lens after the optimization.

KBOSD Triplet Lens F = 100 [mm] File: Triplet Lens

Lagrange Invariant= -0.2182

H	U	HBAR	UBAR	D(U/N)	A	ABAR
12.50000	0.00000	0.00000	0.01746	0.00000	0.00000	0.01746
12.50000	0.00000	0.00000	0.01746	-0.04337	0.18859	0.01746
12.17272	-0.06762	0.05419	0.01120	-0.20443	-0.50253	0.01569
12.17272	-0.24780	0.05419	0.01682	0.22533	-0.49629	0.01571
12.11448	-0.03883	0.06950	0.01020	-0.08230	-0.15663	0.01711
12.11448	-0.10478	0.06950	0.01741	0.05065	-0.02081	0.01789
11.55882	-0.09574	0.12541	0.00963	-0.07087	-0.06733	0.01815
	-0.12500		0.01752			

SpherAb.	Coma	Astig.	FCurv.	Distort.	LChroma.	TChroma.
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.019283	0.001785	0.000165	0.000258	0.000039	0.014359	0.001329
0.628444	-0.019618	0.000612	0.000357	-0.000030	0.037260	-0.001163
-0.675562	0.021392	-0.000677	-0.000409	0.000034	-0.086318	0.002733
0.024461	-0.002672	0.000292	0.000086	-0.000041	0.027112	-0.002962
-0.000266	0.000228	-0.000196	0.000143	0.000046	-0.004095	0.003521
0.003714	-0.001001	0.000270	-0.000103	-0.000045	0.012643	-0.003407
0.000073	0.000114	0.000465	0.000332	0.000003	0.000960	0.000051

4.4.2.2 Transverse Ray Aberrations of the Triplet Lens

Displayed below are the transverse ray aberrations of the KBOSD triplet lens after optimization. This diagram is plotted under the following parameters:

- wavelength 546 nm, 656 nm and 486 nm.
- scale 25 μm
- field angle positions axis(0º) and 1º
- sagittal and tangential planes.

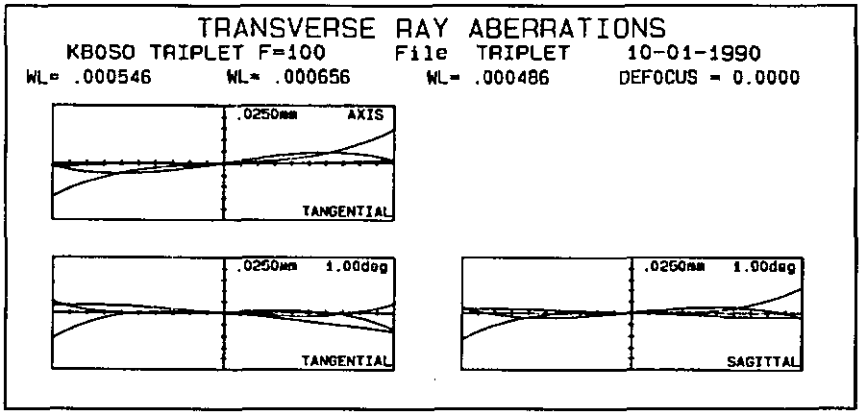


Fig. 4.4.2.2 Transverse Ray Aberrations of the Triplet Lens

4.4.2.3 Spot Diagrams of the Triplet Lens

Displayed below are the spot diagrams of the KBOSD triplet lens after optimization. These spot diagrams are displayed at the following parameters:

- field angle positions axis and 1º
- scale 25 μm
- wavelength 546 nm.

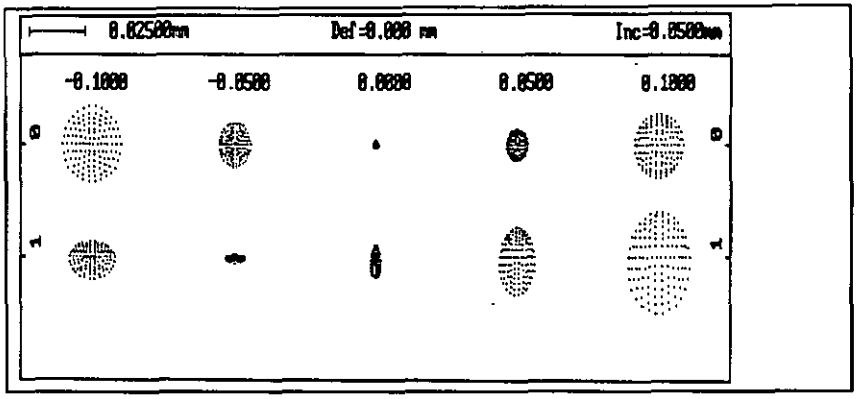


Fig. 4.4.2.3 Spot Diagrams of the Triplet Lens

4.4.2.4 Focal Plane GOTF Tables of the Triplet Lens

Displayed below are geometric optical transfer function at the focal plane of the KBOSD triplet lens after optimization.

Focal Plane GOTF Tables
KBOSD Triplet Lens F=100 [mm] File: Triplet Lens

Defocus = 0.0000 [mm]
Back focus = 92.471 [mm]
Wavelength = 0.0005460 [mm]

	Axis		1.00 Deg	
	T	S	T	S
Frequency [l/mm]				
10.00	0.965	0.965	0.947	0.962
20.00	0.929	0.929	0.861	0.919
30.00	0.893	0.893	0.751	0.871
40.00	0.856	0.856	0.625	0.819
50.00	0.819	0.819	0.494	0.764
60.00	0.781	0.781	0.367	0.707
70.00	0.744	0.744	0.256	0.649
80.00	0.706	0.706	0.172	0.590
90.00	0.669	0.669	0.129	0.531
100.00	0.632	0.632	0.125	0.473

4.4.2.5 Geometrical MTF and Thru-focus Geometrical MTF of the Triplet Lens

Displayed below are the geometrical modulated transfer function and the the thru-focus geometrical modulated transfer function of the KBOSD triplet lens after optimization. The data of the functions plotted below is displayed in section 4.4.2.5 above. The functions reported below are plotted under the following parameters:

- spatial frequency 100 lines/mm
- wavelength 546 nm
- field angle position axis and 1^{o}
- tangential and sagittal planes.

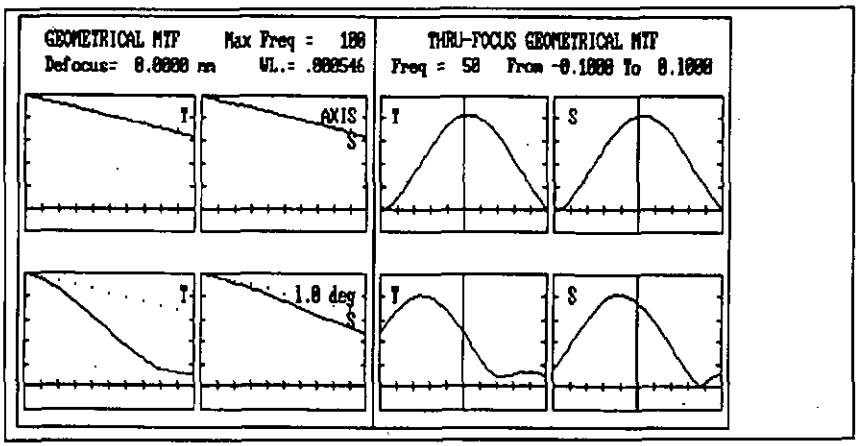


Fig. 4.4.2.5 Geometrical MTF and Thru-focus Geometrical MTF of the Triplet Lens

4.5 Data and Analysis of a High-Aperture Lens

In this section one reports on a high-aperture lens produced by the KBOSD and optimized by the SIGMA-PC [Ref. 17].

4.5.1.1.0 Data of the Start System of the High-Aperture Lens

Displayed below, are the data of the high-aperture lens start system before optimization. This high-aperture lens is composed of two symmetric triplet lenses mounted face to face.

EFL = 55.2786 [mm]
KBOSD Start System of the High Apert. Lens F=55.2786 [mm] File: Pair Reversed Triplet

Object Distance = 100.0000 [mm]
WL1 = 0.000546 [mm]
WL2 = 0.000656 [mm]
WL3 = 0.000486 [mm]

#Type		Curve	Sep.	Index1	Index2	Index3	Dispn.	Clr. Rad	GLASS
1	S	0.015231	0.000	1.000000	1.000000	1.000000	0.000000	21.00	
2	S	-0.015231	8.159	1.768606	1.753572	1.782302	0.028730	21.00	SF14
3	S	-0.022649	0.000	1.000000	1.000000	1.000000	0.000000	21.00	
4	S	0.022649	2.500	1.727353	1.714359	1.739040	0.024681	21.00	SF18
5	S	0.021049	0.000	1.000000	1.000000	1.000000	0.000000	21.00	
6	S	-0.013367	8.050	1.558977	1.553827	1.563323	0.009496	21.00	BAK5
7	S	0.013367	0.000	1.000000	1.000000	1.000000	0.000000	21.00	
8	S	-0.021049	8.050	1.558977	1.553827	1.563323	0.009496	21.00	BAK5
9	S	-0.022649	0.000	1.000000	1.000000	1.000000	0.000000	21.00	
10	S	0.022649	2.500	1.727353	1.714359	1.739040	0.024681	21.00	SF18
11	S	0.015231	0.000	1.000000	1.000000	1.000000	0.000000	21.00	
12	S	-0.015231	8.159	1.768606	1.753572	1.782302	0.028730	21.00	SF14
13	S	0.000000	100.037	1.000000	1.000000	1.000000	0.000000	1.02	

4.5.1.1 Data of the High-Aperture Lens after Optimization

Displayed below are the data(in terms of curvature) of the high-aperture triplet lens after optimization. During the optimization the merit function converges rapidly to zero.

EFL = 55.2786 [mm]
KBOSD High Apert. Lens F=55 [mm] File: Pair Reversed Triplet

Object Distance = 100.0000 [mm]
WL1 = 0.000546 [mm]
WL2 = 0.000656 [mm]
WL3 = 0.000486 [mm]

#Type		Curve	Sep.	Index1	Index2	Index3	Dispn.	Clr. Rad	GLASS
1	S	-0.006759	0.000	1.000000	1.000000	1.000000	0.000000	21.00	
2	S	-0.021425	6.159	1.768606	1.753572	1.782302	0.028730	21.00	SF14
3	S	-0.008088	0.000	1.000000	1.000000	1.000000	0.000000	21.00	
4	S	0.023860	1.500	1.727353	1.714359	1.739040	0.024681	21.00	SF18
5	S	0.023867	0.000	1.000000	1.000000	1.000000	0.000000	21.00	
6	S	-0.013367	9.950	1.558977	1.553827	1.563323	0.009496	21.00	BAK5
7	S	0.013367	0.000	1.000000	1.000000	1.000000	0.000000	21.00	
8	S	-0.023867	9.950	1.558977	1.553827	1.563323	0.009496	21.00	BAK5
9	S	-0.023860	0.000	1.000000	1.000000	1.000000	0.000000	21.00	
10	S	0.008088	1.500	1.727353	1.714359	1.739040	0.024681	21.00	SF18
11	S	0.021425	0.000	1.000000	1.000000	1.000000	0.000000	21.00	
12	S	0.006759	6.159	1.768606	1.753572	1.782302	0.028730	21.00	SF14
13	S	0.000000	100.037	1.000000	1.000000	1.000000	0.000000	1.02	

Displayed below is the data(in terms of curvature radii) of the KBOSD high-aperture lens after optimization.

EFL = 55.2786 [mm]
KBOSD High Apert.Lens F = 55 [mm] File: Pair Reversed Triplet

Object Distance = 100.000 [mm]0

Radius[mm]	Sep. [mm]	Clear diameter	Material
-147.956		42.00	
	6.159		SF14
-46.674		42.00	
	0.000		Air
-123.639		42.00	
	1.500		SF18
41.910		42.00	
	0.000		Air
41.899		42.00	
	9.950		BAK5
-74.812		42.00	
	0.000		Air
74.812		42.00	
	9.950		BAK5
-41.899		42.00	
	0.000		Air
-41.910		42.00	
	1.500		SF18
123.639		42.00	
	0.000		Air
46.674		42.00	
	6.159		SF14
147.956		42.00	
	100.037		Air
Image Plane		2.04	

4.5.1.2 Drawing of the High-Aperture Lens

Plotted below is the drawing of the KBOSD high-aperture lens after optimization.

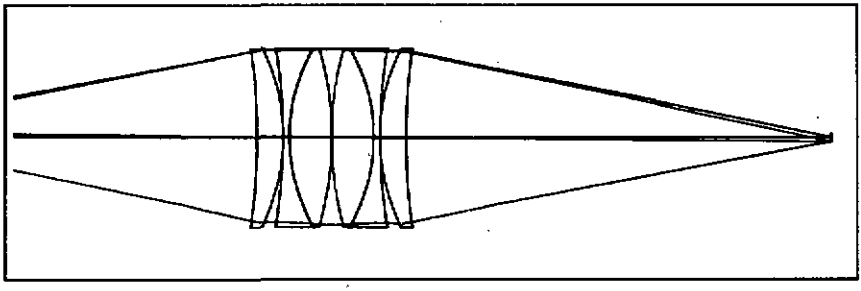


Fig. 4.5.1.2 Drawing of the High-Aperture Lens

4.5.2 Analysis of the High-Aperture Lens after Optimization

In this section one reports on the analysis of the optimized KBOSD high-aperture lens.

4.5.2.1 Third-Order Aberrations of the High-Aperture Lens after Optimization

Displayed below are the third order aberrations of the optimized KBOSD high-aperture lens.

KBOSD High Apert. Triplet Lens F = 55 [mm] File: Pair Reversed Triplet

Object Distance = 100.0000 [mm]

Lagrange Invariant= 0.2000

H	U	HBAR	UBAR	D(U/N)	A	ABAR
20.00000	0.20000	0.08344	-0.00917	-0.10285	0.06482	-0.00973
21.05837	0.17183	0.05302	-0.00494	-0.14004	-0.49407	-0.01074
21.05837	-0.04289	0.05302	-0.00961	0.07003	-0.21321	-0.01003
21.12871	0.04689	0.04495	-0.00538	0.42054	0.95183	-0.00744
21.12871	0.44769	0.04495	-0.00851	-0.37946	0.95196	-0.00744
22.18694	0.10636	-0.01321	-0.00585	-0.06819	-0.29654	-0.00884
22.18694	0.00003	-0.01321	-0.00901	-0.06823	0.29660	-0.00919
21.12913	-0.10632	-0.07011	-0.00572	-0.37943	-0.95191	-0.00631
21.12913	-0.44763	-0.07011	-0.00798	0.42050	-0.95178	-0.00631
21.05885	-0.04685	-0.07809	-0.00532	0.07008	0.21328	-0.01029
21.05885	0.04296	-0.07809	-0.00966	-0.14009	0.49415	-0.01133
20.00070	-0.17179	-0.10725	-0.00473	-0.10280	-0.06475	-0.00965
	-0.19993		-0.00893			

SpherAb.	Coma	Astig.	FCurv.	Distort.	LChroma.	TChroma.
0.008644	-0.001297	0.000195	-0.000117	-0.000012	0.021061	-0.003161
0.719874	0.015651	0.000340	0.000372	0.000015	0.169015	0.003675
-0.067039	-0.003155	-0.000148	-0.000136	-0.000013	-0.064152	-0.003019
-8.049987	0.062928	-0.000492	-0.000402	0.000007	-0.287351	0.002246
7.265705	-0.056788	0.000444	0.000342	-0.000006	0.122513	-0.000958
0.133039	0.003965	0.000118	0.000192	0.000009	0.040075	0.001194
0.133174	-0.004127	0.000128	0.000192	-0.000010	0.040084	-0.001242
7.264449	0.048133	0.000319	0.000342	0.000004	0.122509	0.000812
-8.048578	-0.053339	-0.000353	-0.000402	-0.000005	-0.287341	-0.001904
-0.067133	0.003238	-0.000156	-0.000136	0.000014	-0.064175	0.003096
0.720384	-0.016517	0.000379	0.000372	-0.000017	0.169046	-0.003876
0.008621	0.001285	0.000192	-0.000117	0.000011	0.021039	0.003136
0.021154	-0.000022	0.000964	0.000502	-0.000002	0.002322	-0.000001

Note the low values of the summa(last line) of the order aberrations.

4.5.2.2 Transverse Ray Aberrations of the High-Aperture Lens after Optimization

Displayed below are the transverse ray aberrations of the optimized KBOSD high-aperture lens. These transverse ray aberrations are plotted at the image plane under the following parameters:

- wavelength 546 nm, 486 nm and 656 nm
- aberrations scale 40 µm
- field angle positions axis and 1º
- sagittal and tangential planes.

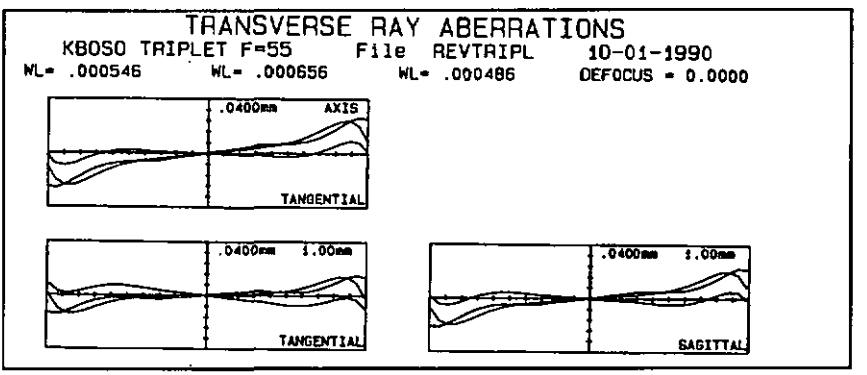


Fig. 4.5.2.2 Transverse Ray Aberrations of the High-Aperture Lens after Optimization

4.5.2.3 Optical Path Differences of the High-Aperture Lens

Displayed below are the optical path differences of the optimized KBOSD high-aperture lens. This optical path differences are plotted under the following parameters:

- wavelength 546 nm
- aberrations scale 1 µm
- field angle positions axis and 1º
- sagittal and tangential plane

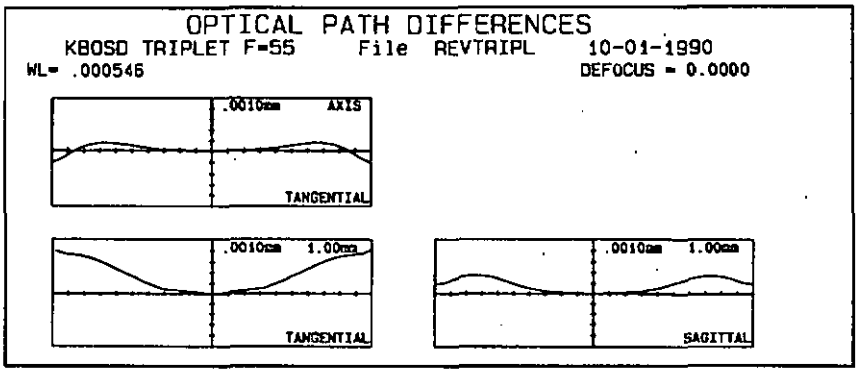


Fig. 4.5.2.3 Optical Path Differences of the High-Aperture Lens

4.5.2.4 Spot Diagrams of the High-Aperture Lens

Displayed below are the spot diagrams of the optimized KBOSD high-aperture lens. This spot diagrams are plotted at the image plane under the following parameters:

- wave length 546 nm
- field angle positions axis and t^{o}
- scale 40 µm

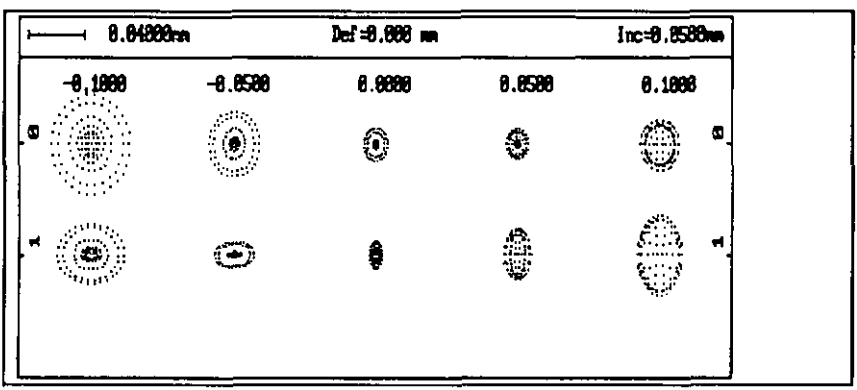


Fig. 4.5.2.4 Spot Diagrams of the High-Aperture Lens

4.5.2.5 Focal Plane GOTF Tables of the High-Aperture Lens after Optimization

Displayed below are the focal plane geometric optical transfert function of the optimized KBOSD high-aperture lens.

Focal Plane GOTF Tables
KBOSD High Apert. Lens F=55 [mm] File: Pair Reversed Triplet

Defocus = 0.0000 [mm]
Back focus = 100.037 [mm]
Wavelength = 0.0005460 [mm]

	Axis		1.00 mm	
	T	S	T	S
Frequency [l/mm]				
8.00	0.970	0.970	0.972	0.979
16.00	0.910	0.910	0.917	0.945
24.00	0.826	0.826	0.839	0.898
32.00	0.727	0.727	0.744	0.841
40.00	0.622	0.622	0.640	0.775
48.00	0.521	0.521	0.532	0.701
56.00	0.432	0.432	0.427	0.623
64.00	0.360	0.360	0.331	0.542
72.00	0.309	0.309	0.249	0.461
80.00	0.279	0.279	0.183	0.381

4.5.2.6 Geometrical MTF and Thru-focus Geometrical MTF of the High-Aperture Lens

Displayed below are the geometric modulated transfert function and the thru-focus geometrical modulated transfert function. These diagrams are plotted under the following parameters:

- spatial frequency 80 lines/mm
- wavelength 546 nm
- thru-focus spatial frequency 80 lines/mm

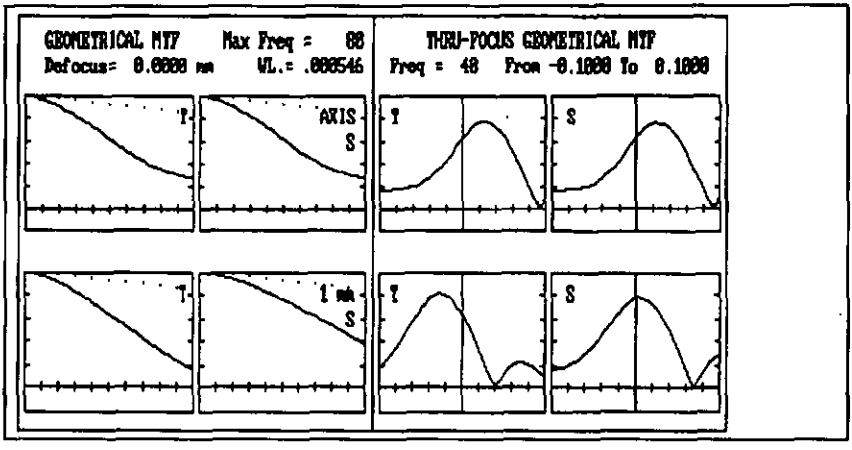


Fig. 4.5.2.6 Geometrical MTF and Thru-focus Geometrical MTF of the High-Aperture Lens

4.6 Data and Analysis of an Afocal Optical System: The Galilean Telescope

A Galilean telescope is a telescope with a negative, divergent lens and a positive, convergent lens. A system is assumed to be afocal if its numerical aperture is zero. The following optical systems are reported as afocal and function as laser beam expanders or as telescopes. In this case the magnification is also the angular magnification or the ratio of the beam sizes in the object and image spaces. This beam expander has a magnification of four and can be used at the following three wavelengths WL1 = 588 nm, WL2 = 656 nm and WL3 = 488 nm. Its entrance pupil has a diameter of 6 mm and the exit pupil has a diameter of 40 mm, the whole optical system galilean telescope has a length of around 300 mm.

4.6.1.1.0 Data of the Start System Galilean Telescope

Dispalyed below is the data of a optical start system (before optimization) for Galilean telescope proposed by the KBOSD. This optical start system is composed of plano-concave singlet and a plano-convex doublet. The plano-concave singlet represents the entrance part of the telescope and the doublet represents its sortance part. The plane surface of the entrance part is mounted toward the plane surface of the doublet (sortance part); because of this assembling this start system Galilean telescope leads to a good optical quality telescope after the optimization.
This Galilean telescope has an angular magnification of 4.

KBOSD Start System Galilean Telescope 4* File: Galilean Telescope

Radius[mm]	Sep. [mm]	Clear diameter	Material
	0.000		Air
-58.900		10.00	
	2.000		BAF3
Plane		10.00	
	292.922		Air
Plane		40.00	
	2.000		SF55
223.000		40.00	
	0.000		Air
197.100		40.00	
	4.033		BAF3
-197.100		40.00	
	0.000		Air
Plane		50.35	

4.6.1.1 Data of the Optimized Galilean Telescope

Displayed below is the data(in terms of surfaces curvature) of the KBOSD Galilean telescope after optimization.

KBOSD Galilean Telescope 4* File: Galilean Telescope

WL1 = 0.000588 [mm]
WL2 = 0.000656 [mm]
WL3 = 0.000488 [mm]

#	Type	Curve	Sep.	Index1	Index2	Index3	Dispn.	Clr. Rad	Glass
1	S	0.000000	0.000	1.000000	1.000000	1.000000	0.000000	3.00	
2	S	-0.016949	0.000	1.000000	1.000000	1.000000	0.000000	5.00	
3	S	0.000000	2.000	1.582668	1.578932	1.591255	0.012323	5.00	BAF3
4	S	0.000000	292.922	1.000000	1.000000	1.000000	0.000000	20.00	
5	S	0.004484	2.000	1.761794	1.753657	1.781415	0.027758	20.00	SF55
6	S	0.005076	0.000	1.000000	1.000000	1.000000	0.000000	20.00	
7	S	-0.005076	4.000	1.582668	1.578932	1.591255	0.012323	20.00	BAF3
8	S	0.000000	0.000	1.000000	1.000000	1.000000	0.000000	25.18	

Displayed below is the data (in terms of curvature radii) of the KBOSD Galilean telescope after optimization.

KBOSD Galilean Telescope 4* File: Galilean Telescope

Radius[mm]	Sep. [mm]	Clear diameter	Material
Plane		6.00	
	0.000		Air
-59.000		10.00	
	2.000		BAF3
Plane		10.00	
	292.922		Air
Plane		40.00	
	2.000		SF55
223.000		40.00	
	0.000		Air
197.000		40.00	
	4.000		BAF3
-197.000		40.00	
	0.000		Air
Plane		50.35	

4.6.1.2 Drawing of the Galilean Telescope

Displayed below is the drawing of the KBOSD Galilean telescope after optimization. Heuristically the geometrical form of this KBOSD Galilean telescope has a minimum of aberrations.

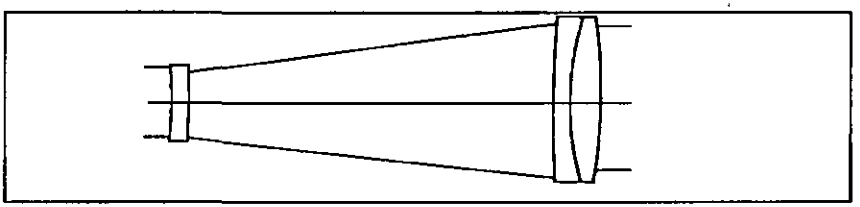


Fig. 4.6.1.2 Drawing of the Galilean Telescope after Optimization

4.6.2 Analysis of Optimized the Galilean Telescope

In this section one reports on the analysis of KBOSD Galilean telescope after optimization.

4.6.2.1 Angular Ray Aberrations

The angular ray aberrations of the KBOSD Galilean telescope after optimization are plotted under the following parameters:

- wavelength 588 nm
- scale 0.0003 radian
- field angle positions axis and 1º
- tangential and sagittal plane.

According to the diagrams plotted below the KBOSD Galilean telescope has a low aberrations.

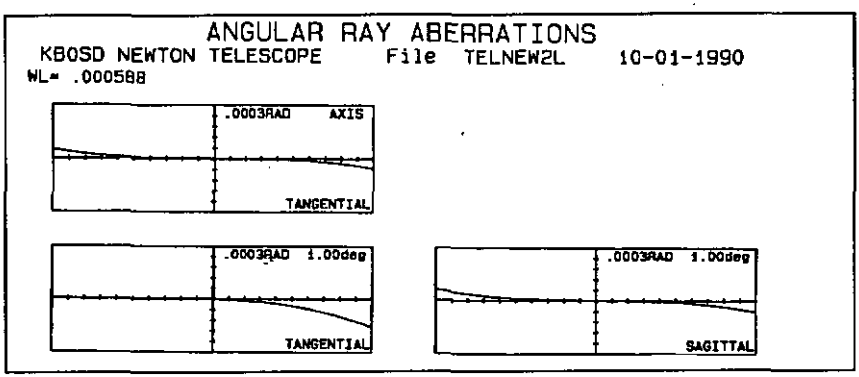


Fig.4.6.2.1 Angular Ray Aberrations of the KBOSD Galilean Telescope

4.6.2.2 Focal Plane GOTF Tables

Geometrical MTF

For the geometrical MTF angular aberrations are used, and MTF is computed in terms of the angular frequency, in cycles/radian in the image space. The MTF is computed in the image space. The geometric MTF is multiplied by the MTF of a diffraction-limited system. In doing this, it assumed that the exit pupil radius = entrance pupil radius/angular magnification. The aberration is defined as an angle, in radians.

Through-Focus Geometric MTF

The Through-focus geometric MTF is computed in the image space. The frequency is measured in cycles/radian. It is necessary to compute the angular aberrations produced by defocusing and this is done by assuming that a weak perfect lens is placed in the exit pupil plane. The power of this lens is changed from -2.Zinc to +2.Zinc (Zinc is the defocusing increment measured in dioptries, as used in the spot diagrams). The modified angular aberrations are computed, plotted in spot diagrams, and used to compute through-focus MTF.
According to the diagrams plotted below the KBOSD Galilean telescope has a good optical quality. This diagrams is plotted under the following parameters:

- spatial frequency is 10000 cycles/rad
- wavelength 588 nm
- field angle positions axis and 1º
- sagittal and tangential plane.

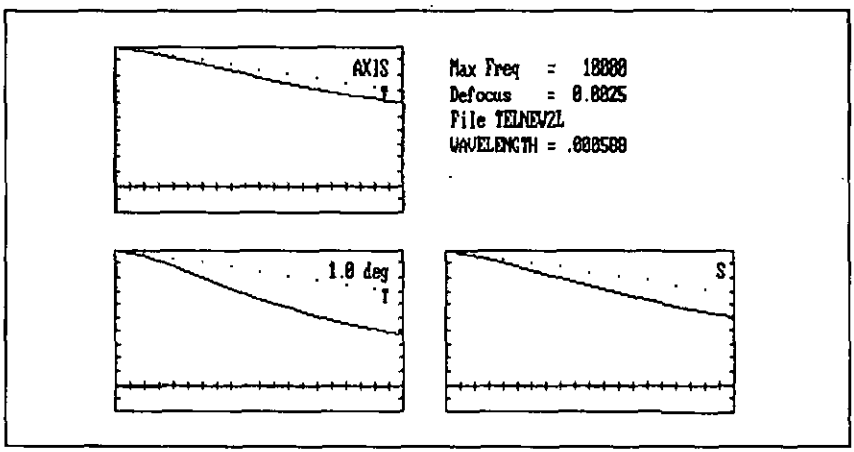


Fig. 4.6.2.2 Diffraction MTF of the KBOSD Galilean Telescope after Optimization

Displayed below is the diffraction MTF table of the KBOSD galilean telescope after optimization. At the frequency of 10000 cycles/rad the KBOSD Galilean telescope has a spatial resolution 60% on axis, it is considered as a good resolution.

KBOSD Galilean Telescope 4* File: Galilean Telescope

Defocus = 0.003 D
Back focus = 0.000 [mm]
Wave length = 0.000588 [mm]

	Axis	1.00 deg	
Frequency[c/rad]		S	T
1000.000	0.964	0.961	0.951
2000.000	0.923	0.910	0.879
3000.000	0.879	0.854	0.797
4000.000	0.835	0.797	0.715
5000.000	0.791	0.741	0.638
6000.000	0.750	0.688	0.570
7000.000	0.709	0.638	0.511
8000.000	0.671	0.592	0.461
9000.000	0.635	0.550	0.417
10000.000	0.600	0.511	0.378

4.7 Data and Analysis of a Kepler Telescope

A Kepler telescope is a telescope with two positive convergent lens.

4.7.1.1.0 Data of the Start System Kepler Telescope

Displayed below is the data of a Kepler telescope start system proposed by the KBOSD. This KBOSD Kepler telescope is composed of two parts. A plano-convex singlet lens representing the entrance part and a plano-convex doublet representing the sortance part. The plane surface of this entrance part is mounted toward the the plane surface of the sortance part. Heuristically, he geometrical form and the assembling of the KBOSD telescope lead to a telescope with good optical quality after optimization. This KBOSD Kepler telescope has a magnification of -3.

KBOSD Start System kepler Telescope 3* File: TELKEP2L

Radius [mm]	Sep. [mm]	Clear diameter	Material
	0.00		Air
51.537		10.00	
	2.240		BAF3
Plane		10.00	
	345.741		Air
Plane		30.00	
	2.000		SF13
134.586		30.00	
	0.000		Air
125.130		30.00	
	3.800		BAF3
125.130		30.00	
	0.000		Air

4.7.1.1 Data of the Optimized Kepler Telescope

Displayed below is the data of the KBOSD Kepler telescope after optimization. This data is displayed in terms of surface curvature.

KBOSD KEPLER Telescope 3* File: TELKEP2L

WL1 = 0.000588 [mm]
WL2 = 0.000656 [mm]
WL3 = 0.000489 [mm]

#	Type	Curve	Sep.	Index1	Index2	Index3	Dispn.	Clr. Rad	Glass
1	S	0.000000	0.000	1.000000	1.000000	1.000000	0.000000	3.50	
2	S	0.019417	0.000	1.000000	1.000000	1.000000	0.000000	5.00	
3	S	0.000000	2.240	1.582668	1.578932	1.591163	0.012231	5.00	BAF3
4	S	0.000000	345.741	1.000000	1.000000	1.000000	0.000000	15.00	
5	S	0.007463	2.000	1.740765	1.733038	1.759190	0.026151	15.00	SF13
6	S	0.008000	0.000	1.000000	1.000000	1.000000	0.000000	15.00	
7	S	-0.008000	3.800	1.582668	1.578932	1.591163	0.012231	15.00	BAF3
8	S	0.000000	0.000	1.000000	1.000000	1.000000	0.000000	21.18	

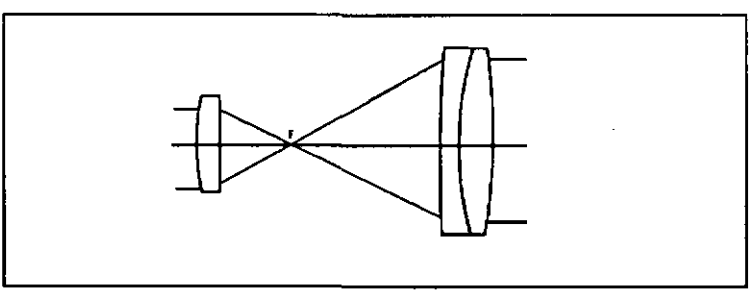
Displayed below is the data (in terms of of curvature radii) of the KBOSD Kepler telescope after optimization.

KBOSD KEPLER Telescope 3* File: TELKEP2L

Radius [mm]	Sep. [mm]	Clear diameter	Material
Plane		7.00	
	0.00		Air
51.500		10.00	
	2.240		BAF3
Plane		10.00	
	345.741		Air
Plane		30.00	
	2.000		SF13
134.000		30.00	
	0.000		Air
125.000		30.00	
	3.800		BAF3
125.000		30.00	
	0.000		Air
Plane		42.35	

4.7.1.2 Drawing of the Kepler Telescope

Displayed below is a scheme of the KBOSD Kepler telescope after optimization.



Scheme of the kepler Telescopic Optical System

4.7.2 Analysis of the Kepler Telescope

In this section one reports on the analysis of the KBOSD kepler telescope after optimization.

4.7.2.1 Angular Ray Aberrations of the Kepler Telescope

Displayed below are the angular ray aberrations of the KBOSD Kepler telescope after optimization. This angular ray aberrations are plotted under the following parameters:

- wavelength 588 mm
- scale 0.0003 rad
- angle field positions axis and 1º
- tangential and sagittal plane.

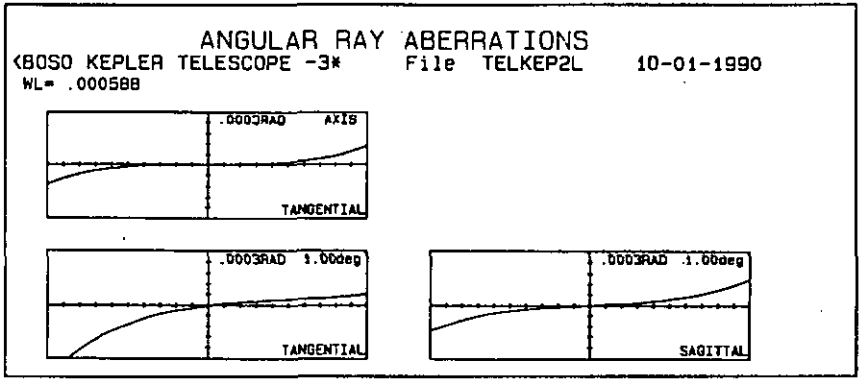


Fig. 4.7.2.1 Angular Ray Aberrations of the KBOSD Kepler Telescope

4.7.2.2 Optical Path Differences of the Kepler Telescope

Displayed below are the optical path differences of the KBOSD Kepler telescope after optimization. This diagrams are plotted under the following parameters:

- wavelength 588 nm
- scale 0.001 mm
- tangential and sagittal planes

- field angle positions axis and 1º

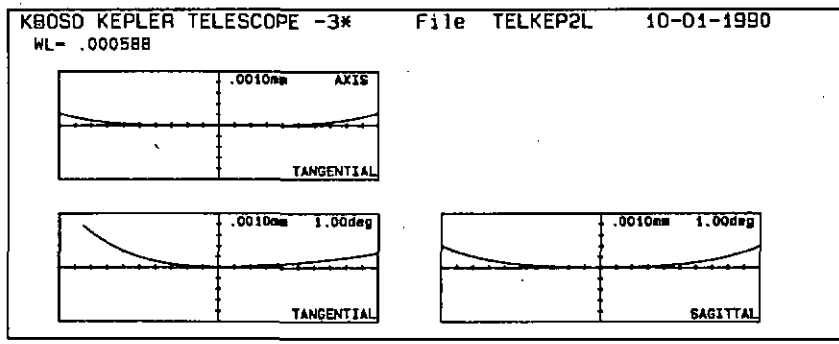


Fig. 4.7.2.2 Optical Path Differences of the KBOSD Kepler Telescope after Optimization.

4.7.2.3 Diffraction MTF Tables

Displayed below are the diffraction modulated transfer function tables of the optimized KBOSD Kepler telescope.

KBOSD KEPLER Telescope 3* File: Telkep21

Defocus = 0.000 D
Back focus = 0.000 [mm]
Wave length = 0.000588 [mm]

	Axis	1.00 deg	
Frequency [c/rad]		S	T
800.000	0.961	0.949	0.923
1600.000	0.908	0.867	0.787
2400.000	0.849	0.773	0.639
3200.000	0.789	0.678	0.509
4000.000	0.731	0.589	0.412
4800.000	0.676	0.511	0.348
5600.000	0.626	0.445	0.307
6400.000	0.580	0.388	0.279
7200.000	0.538	0.340	0.256
8000.000	0.499	0.300	0.235

At the spatial frequency of 8000 cycles/rad the KBOSD Kepler telescope has a spatial resolution of 49.9 %; such as resolution may considered enough for many application.

4.7.2.4 Geometrical MTF and Thru-focus Geometric MTF of the Kepler Telescope

Displayed below are the geometrical modulated transfer function and the thru-focus geometric modulated transfer function of the KBOSD Kepler

telescope after optimization, the data of these diagrams are displayed in section 4.7.2.3 above. THis diagrams are plotted under the following parameters:

- spatial frequency 8000 cycles/rad
- wavelength 588 nm
- field angle positions axis and 1º
- sagittal and tangential planes.

The thru-focus modulated transfer function is plotted at the frequency of 4000 cycles/rad.

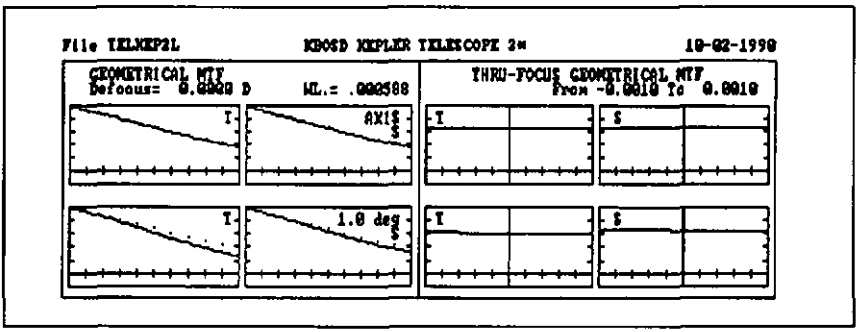


Fig. 4.7.2.4 Geometrical MTF and Thru-focus Geometric MTF of the Kepler Telescope

4.8 Conclusion:

The KBOSD proposes and produces a start optical system in a few seconds (less than 50 sec. with a Macintosh II fx). This start optical system is optimized subsequently by a lens design program and after a several iterations a good optical design system is obtained. The KBOSD start systems are precalculated, predesigned and have a low third order aberrations, because of this reasons they allows the merit function to converge rapidly during the optimization.
To attempt the same goal, an optical design expert will normally spend much more time than the computer.

Thus the KBOSD is a powerful tool for assisting an optical design expert in saving time and reducing the probability of error. It is also a valuable tool for a novice in the optical design field.

The analysis of optical systems at 100 lines/mm and high ratios of the focal length to diameter demonstrate that these optical systems are of good quality [see appendix 1]. Such optical systems are commonly used at

lower spatial frequency. The results obtained by this KBOSD are very encouraging and confirm the validity and usefulness of such a program.

5.0 Introduction

In this chapter several optical systems optimized from parallel plane plates as start optical systems are reported and compared with optical systems generated at the same conditions by the KBOSD. The parallel plane plates start systems are optimized and analyzed with the Lens Design Program Sigma PC [Ref. 17].

5.1 Discusions and Comparisons of Singlet Lenses

In this section two singlet lenses are compared and discussed. One singlet lens is produced by the KBOSD and optimized using the Sigma PC [Ref. 17]. This singlet lens is reported in more detail in section 4.2 of this thesis. The second singlet lens is produced from parallel plane plates and is also optimized using the Sigma PC [17].

Both singlet lenses are designed at a wavelength of 656 nm and have a front focal length of 110.0 mm and a back focal length of 108.12 mm. They have a clear diameter of 20 mm and an entrance pupil diameter of 18 mm. The object is now at infinity. The numerical aperture is 0.082.

5.1.1 Parallel Plane Plate Start System Singlet Lens

Starting from a parallel plane plate the optimization program was incapable of starting. In the case of parallel plane plate, all surfaces are plane and the outgoing rays are parallel to the optical axis, this fact leads to a numerical aperture equal to zero. While the numerical aperture is zero the optimization program displays a message error "overflow" and stops. It's the first incapacity of the optimization program if one starts from a parallel plane plate and tries to attempt a singlet lens of good quality.

To avoid this incapacity of the optimization program one uses the "Angle Solve" function to alternate the last curvature radius of the parallel plane plate according to the following relation:

$$R = F*(N-1)$$

Where:

R: Last curvature radius
F: Focal length of the plano-convex singlet lens.
N: Refractive index of the plano-convex singlet lens.

Because of this fact the numerical aperture NA becomes:

$$NA = H/F_b$$

Where:

H: High of the marginal ray at the last surface
F_b: Back focal length of the singlet.

The second assistance to the optimization program is that we input the optical glass and the separation of the two surfaces. The optical glass(sf15) proposed by the KBOSD was selected. One assumes that the thickness is 3 mm which is a reasonable thickness of a plano-convex singlet. The operations described above lead to the plane parallel start system singlet lens displayed in Fig. 5.1.1

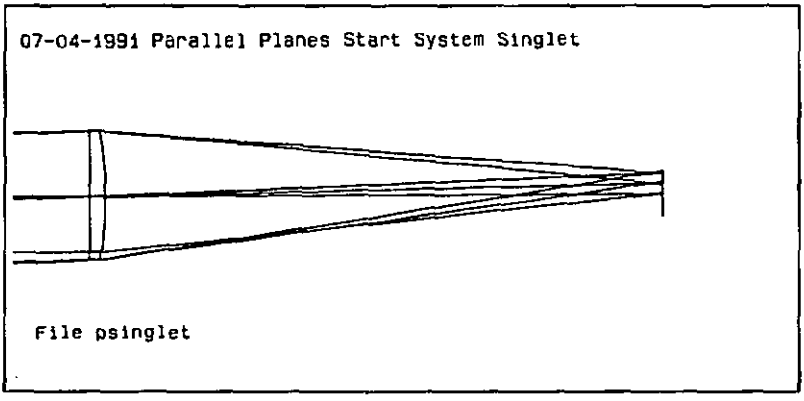


Fig. 5.1.1 Drawing of the Parallel Plane Plate Start System Singlet Lens before Optimization.

Note that the convex surface is toward the outgoing rays. The data of the parallel plane plate start system singlet lens before the optimization are displayed below.

EFL= 110.000 [mm]
Parallel Planes Start System Singlet File: PSinglet

Radius[mm]	Sep. [mm]	Clear diameter	Material
Pupil(Plane)		20.00	
	3		SF15
-76.143		20.00	
	109.67		Air
Image Plane		8.55	

5.1.2 KBOSD Start System Singlet Lens

In the same start conditions: the clear diameter = 20 mm, the focal length is equal to 110 mm and the wavelength = 656 nm, the KBOSD proposes the plano-convex singlet lens shown below in Fig. 5.1.2.

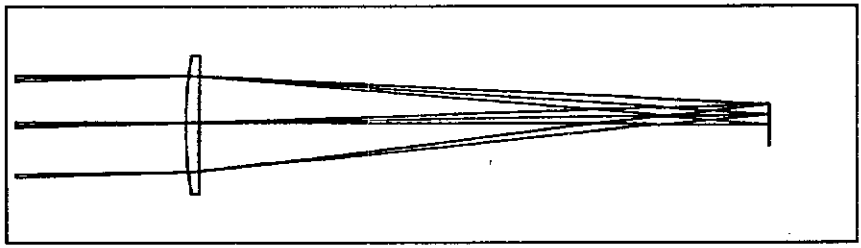


Fig. 5.1.2 Drawing of the KBOSD Start System Singlet Lens Before Optimization

Note that the convex surface is toward the incoming rays.

Calling the predicate *calcul(110.0, Magnification, l, 108.0, z, z')* the KBOSD answered with the data of the start system singlet lens as displayed below.

EFL= 110.000 [mm]
KBOSD Singlet Lens Start System

Radius[mm]	Sep. [mm]	Clear diameter	Material
Pupil: 76.88		20.00	
	2.653		SF15
Plane		20.00	
	108.00		Air
Image Plane		8.55	

This plano-convex start system singlet lens proposed by the KBOSD is composed of two surfaces and an optical glass. First surface has a radius of 76.88 mm and the second surface is plane. The singlet lens has a thickness of 2.653 mm, a clear diameter of 20.00 mm and a sf15 optical glass.

5.1.3 Optimization of the Parallel Plane Plate Singlet Lens

Displayed below Fig. 5.1.3.1 are the KBOSD singlet lens start system and the parallel plane plate start system before the optimization.

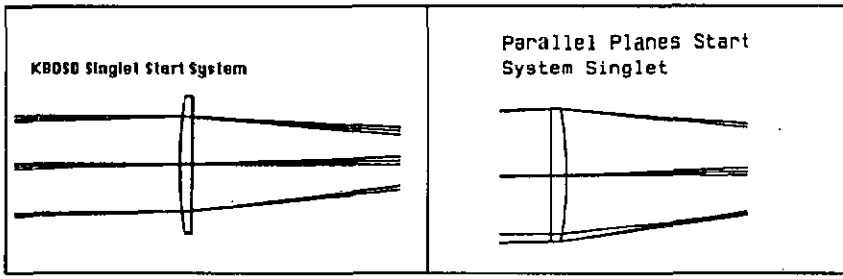


Fig. 5.1.3.1 KBOSD and Parallel Plane Plate Start System.

During the optimization process of the KBOSD start system and of the parallel plane plate start system the curvature radii and the thickness are variable. The optical glass is fixed.

After twenty iterations the merit function converges and the optimization program stops. The Data of the parallel plane plate start system of the singlet lens after Optimization are displayed below.

EFL= 110.000 [mm]
Parallel Plane Plate Start System Singlet File: PSinglet

Radius[mm]	Sep. [mm]	Clear diameter	Material
75.358		20.00	
	3		SF15
7168.605		20.00	
	1007.88		Air
Image Plane		7.55	

During the optimization, the parallel plane plate start system singlet lens is reversed. Its first radius is alternated from infinity to 75.358 mm. Its second radius is changed from 76.143 to 7168.605 mm. The thickness remains fixed at 3 mm. These radical changes are provoked by the fact that there is no predesign of the parallel plane plate start system singlet.

Illustrated below, Fig. 5.1.3.2 is the parallel plane plate start system of the singlet lens after the optimization. Note that the geometrical form of the parallel plane plate start system after the optimization (Fig. 5.1.3.2) converged to the geometrical form of the KBOSD start system(Fig. 5.1.2) before the optimization.

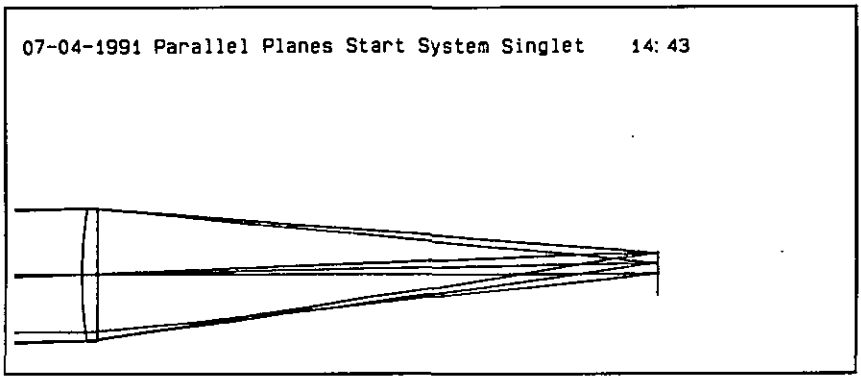


Fig. 5.1.3.2 Parallel Plane Plate Start System of the Singlet Lens after Optimization

5.1.4 Optimization of the KBOSD Start System Singlet Lens

By the optimization, the merit function of the KBOSD singlet start system converges after five iterations and the optimization program stops. The KBOSD singlet start system keeps the same geometrical form before and after the optimization. Only a few changes in the curvature radii are produced during the optimization.

Displayed below is the drawing of the KBOSD start system singlet lens after the optimization.

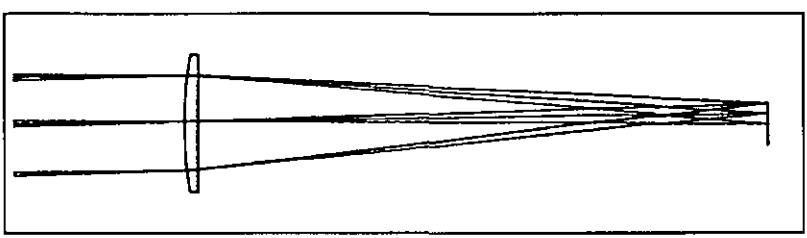


Fig. 5.1.4 KBOSD Singlet Start System after Optimization

Displayed below are the data of KBOSD start system singlet lens after the optimization.

EFL= 110.000 [mm]
KBOSD Singlet Lens Start System after Optimization

Radius[mm]	Sep. [mm]	Clear diameter	Material

Pupil: 75.603		20.00	
	2.600		SF15
10510.539		20.00	
	108.123		Air
Image Plane		7.82	

Heuristically in this particular situation, the geometrical form of the singlet, the convex surface toward the incoming parallel rays and the plane surface toward the convergent rays, leads to an optical system with a minimum of aberrations. Note that the geometrical form of the start system proposed by the KBOSD is the same as the geometrical form of the optimization result of the parallel plane start system.

The start system proposed by the KBOSD kept relatively the same parameters before and after the optimization. Its first curvature radius is changed from 76.88 mm to 75.603 mm. The second radius is alternate from infinity to 10510.539 mm. The thickness moved from 2.653 mm to 2.6 mm. The fact that the KBOSD pre-designed heuristically the start systems explains the small changes during the optimization and allows the convergence of the merit function to a minimum in a small number of iterations.

5.1.5 Comparisons of the Parallel Plane Plate Start System Singlet Lens with the Singlet Lens Generated by the KBOSD

The optimized parallel plane plate singlet lens and the KBOSD start system after optimization are analyzed in the same conditions: same clear aperture, same scale, same field angles and the same spatial frequency. The results of the analysis are reported in this section in order to compare the quality of each singlet lens.

5.1.5.1 Comparisons of the Transverse Ray Aberrations

Figure 5.1.5.1.1 represents the transverse ray aberrations of the KBOSD singlet lens at the field angle axis(0º degree), 1º and 2º respectively in the sagittal and tangential image plane. The scale of the transverse ray aberrations is 0.05 mm.

Figure 5.1.5.1.2 represents the transverse ray aberrations of the parallel plane plate start system singlet at the field angle axis(0º degree), 1º and 2º respectively in the sagittal and tangential image plane. The scale of the transverse ray aberrations is 0.05 mm.
Displayed below are the transverse ray aberrations of the parallel plane plate start system singlet lens at the wavelength 656 nm after optimization.

Comparing the transverse ray aberrations of the KBOSD singlet displayed in Fig. 5.1.5.1.1 with the transverse ray aberrations of the parallel plane plate displayed in Fig. 5.1.5.1.2 one observes:

- within the zonal area(60% of the clear diameter) the transverse ray aberrations are comparable and are closed to zero.
- outside the zonal area, the KBOSD start system has less transverse ray aberrations than the parallel plane plate start system.

According to the diagrams displayed in Fig. 5.1.5.1.1 and in 5.1.5.1.2 one may conclude that the KBOSD singlet is of a better quality (in terms of transverse ray aberrations) than the parallel plane plate start system.

Displayed below are the transverse ray aberrations of the KBOSD singlet lens at the wavelength 656 nm after optimization.

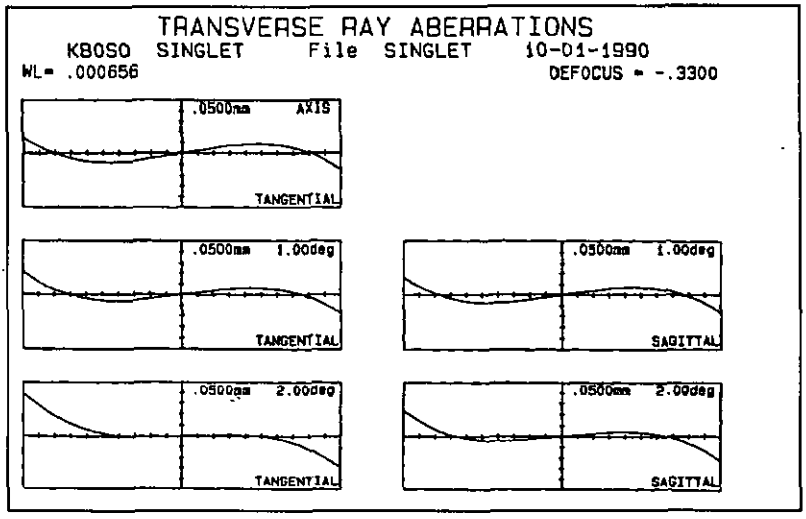


Fig. 5.1.5.1.1 Transverse Ray Aberrations of the KBOSD Start System Singlet Lens after Optimization

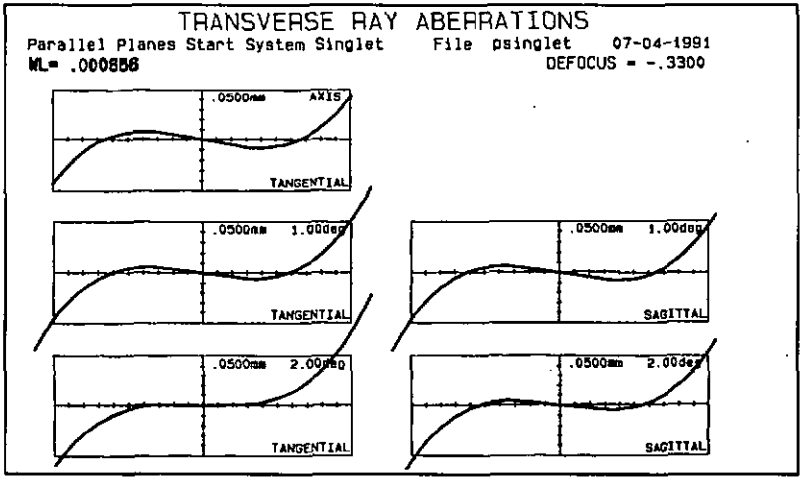


Fig. 5.1.5.1.2 Transverse Ray Aberrations of the Parallel Plane Plate Start System Singlet Lens after Optimization

5.1.5.2 Comparisons of the Spot Diagrams

In this section an analysis of the spot diagrams of the KBOSD singlet start system and the parallel plane plate singlet start system is reported. The spot diagrams are reported under the same conditions: same wavelengths, the same scale and the same field angles.

According to the KBOSD singlet lens start system spot diagrams displayed in Fig. 5.1.5.2.1 and the spot diagrams of the plan parallel plate start system displayed in Fig. 5.1.5.2.2 one may conclude that the KBOSD singlet lens start system has greater performance than the parallel plane plate singlet lens start system.

Figure 5.1.5.2.1 represents the spot diagrams of the KBOSD singlet lens after optimization. This spot diagrams is plotted at the following parameters:

- wavelength 656 nm
- field angle positions 0^{o}(axis), 1^{o} and 2^{o}
- scale 0.05 mm.

Figure Fig. 5.1.5.2.2 represents the spot diagrams of the parallel plane plate start system singlet lens after optimization. These spot diagrams are plotted at the:

- wavelength 656 nm
- field angle positions 0^{o}(axis), 1^{o} and 2^{o}
- scale 0.05 mm.

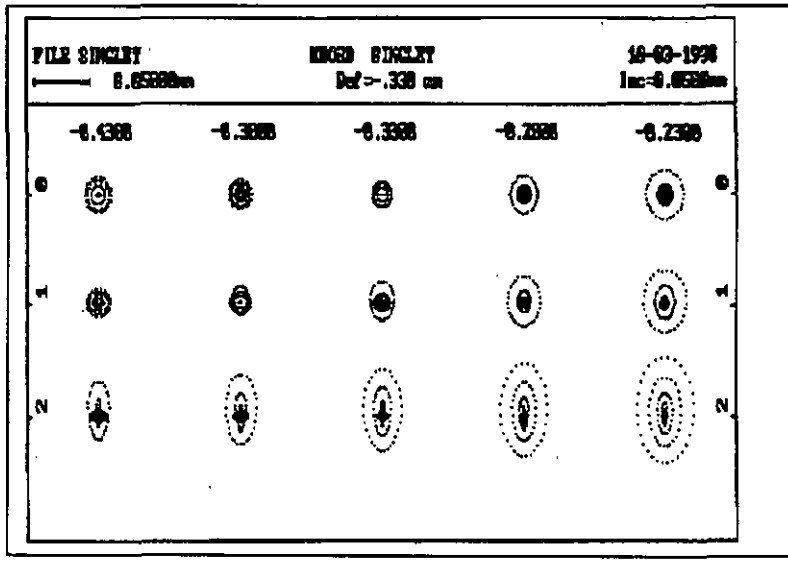


Fig. 5.1.5.2.1 Spot Diagrams of the KBOSD Singlet Lens Start System after Optimization.

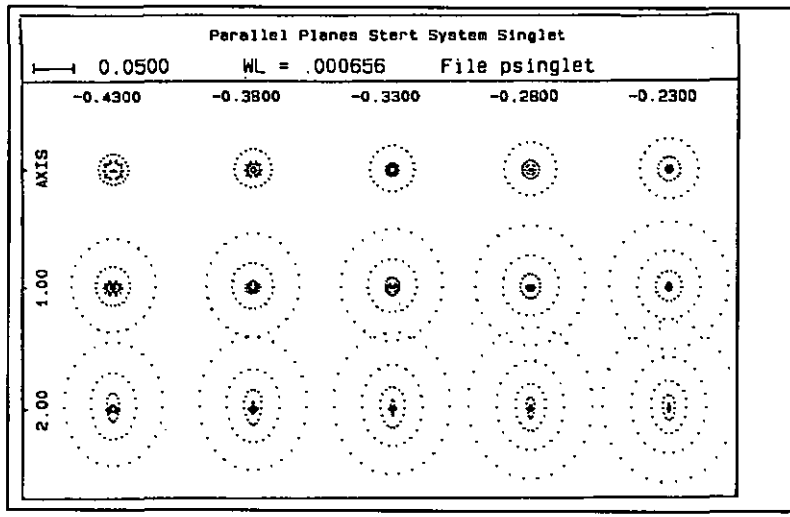


Fig. 5.1.5.2.2 Spot Diagrams of the Parallel Plane Plate Start System Singlet Lens

5.1.5.3 Comparisons of the Geometrical MTF

In this section one reports upon the geometrical MTF of the KBOSD start system singlet lens and the parallel plane plate start system after optimization. Both geometrical MTF diagrams displayed in figure 5.1.5.3 are plotted at the same parameters:

- field angles 0º(axis), 1º and 2º
- spatial frequency 50 lines/mm
- full aperture

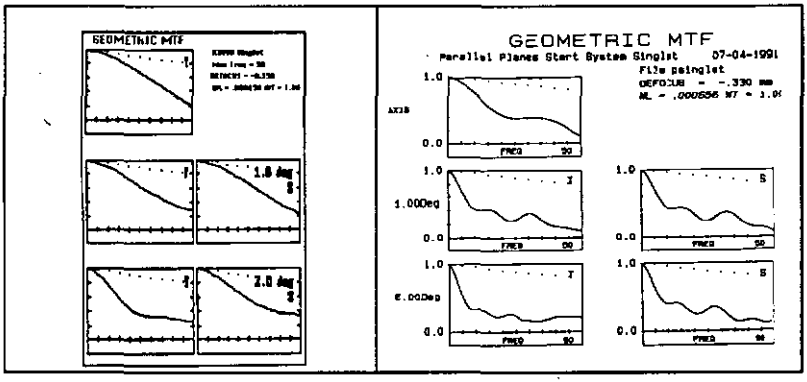


Fig. 5.1.5.3 Geometrical MTF of the KBOSD Singlet Lens(left) and Geometrical MTF of the parallel plane plate start system(right)

According to the diagrams shown in figure 5.1.5.3 it can be said that the KBOSD singlet lens has a greater resolution than the singlet lens obtained from a parallel plane plate start system.

5.2 Discusions and Comparison of Doublet Lenses

In this section, two doublet lenses are compared and discussed. One doublet lens is obtained from the optimization of the start system doublet produced by the KBOSD and displayed in section 4.1.0. The other doublet lens is obtained from the optimization of a parallel plane plates. Both start system doublet lenses are optimized with the PC Sigma [Ref. 17]. These two doublet lenses are designed and analyzed under the following conditions:

- wavelength = 588 nm and 656 nm
- focal length = 100.0 mm
- clear diameter = 20 mm
- entrance pupil diameter = 19 mm
- object is supposed to be at infinity
- numerical aperture = 0.095.
- spatial frequency = 100 lines/mm
- field angles 0º(axis) and 1º.

5.2.1 Parallel Plane Plates Start System Doublet Lens

Starting from a parallel plane plates the optimization program was incapable of starting. In the case of parallel plane plate, all surfaces are plane and the outgoing rays are parallel to the optical axis, this fact leads to a numerical aperture equal to zero. While the numerical aperture is zero the optimization program displays a message error "overflow" and stops. It's the first incapacity of the optimization program if one starts from a parallel plane plate and tries to attempt a doublet lens of good quality.

To avoid this incapacity of the optimization program one uses the "Angle Solve" function to alternate the last curvature radius of the second parallel plane plate according to the following relation:

$$R = F*(Nsf15 - 1)$$

Where:

R : Curvature radius of the last surface
F : Focal length of the final plano-convex singlet lens.
Nsf15: Refractive index of the second parallel plane plate(sf15).

Due to this fact the numerical aperture NA is calculate according to the following relation:

$$NA = H/F_b$$

Where:

H : High of the marginal ray at the last surface(aperture radius)
F_b : Back focal length of the singlet.

The second assistance to the optimization program is that we input two optical glasses. These optical glasses are baf3 and sf15. The baf3 and sf15 optical glasses are appropriate to chromatic aberrations corrections.The optical glasses baf3 and sf15 are proposed by the KBOSD and used in the pre-design of the KBOSD doublet start system. The optimization program is incapable of calculating the thickness of the parallel plane plates doublet lens. A thickness of 5 mm is introduced for each parallel plane plate. This thickness of 5 mm is considered to be a reasonable thickness of a doublet start system. The separation between the two plane plates is zero. The operations described above lead to the plane parallel start system doublet lens displayed in Fig. 5.2.1

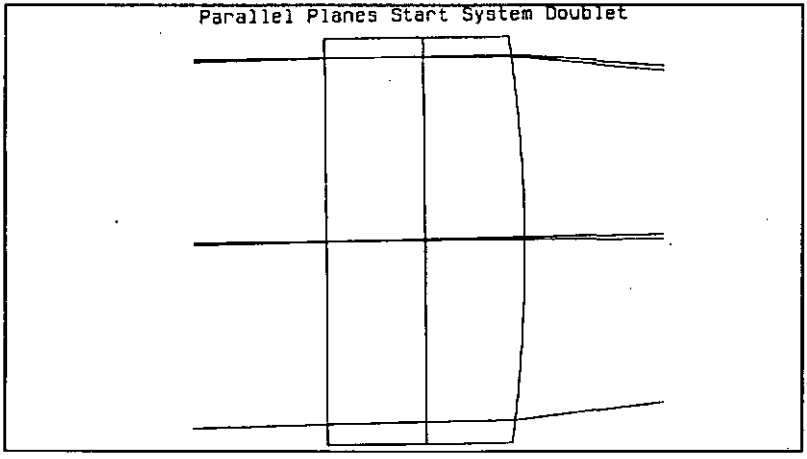


Fig. 5.2.1 Drawing of the Parallel Plane Plates Start System Doublet Lens before Optimization.

The parallel plane plates start system doublet lens is composed of two pieces: a parallel plane plate with a thickness of 5 mm (left in Fig. 5.2.1) made of baf3 glass and a plano-convex singlet make of sf15 glass; the plano-convex singlet has a curvature radius of 69.895 mm, a focal length of 100 mm and a thickness of 5 mm. This doublet has a clear diameter of 20 mm. The image plane is located at the distance of 100 mm from the last surface of the doublet.

The data of the parallel plane plate start system doublet lens before the optimization are displayed below:

EFL= 100.000 [mm]
Parallel Planes Start System Doublet before Optimization File: PDoublet

Radius[mm]	Sep. [mm]	Clear diameter	Material
Pupil(Plane)		20.00	
	5.00		BAF3
Plane		20.00	
	0.00		Air
Plane		20.00	
	5.0		SF15
-69.895		20.00	
	100.00		Air
Image Plane		3.57	

5.2.2 KBOSD Start System Doublet Lens

Under the same conditions as the parallel plane plates start system doublet the KBOSD proposes the following start system doublet lens.

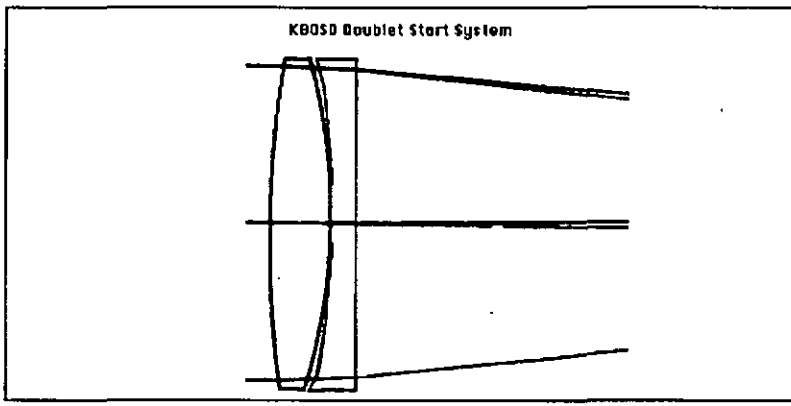


Fig. 5.2.2 Drawing of the KBOSD Start System Doublet Lens before Optimization.

The KBOSD start system doublet lens has a back-focal length of 97.184 mm and is composed of a bi-convex singlet lens and a plano-concave singlet lens. The bi-convex singlet is made of baf3 glass, it has curvature radii of 41.435 mm and -41.435 mm and a center thickness of 4.143 mm. The plano-concave singlet is made of sf15 glass, it has curvature radii of -38.556 mm and infinity and a center thickness of 2 mm. This KBOSD has a clear diameter of

20 mm. The data of the KBOSD start system doublet lens before the optimization is displayed below:

EFL= 100.000 [mm]
KBOSD Start System Doublet before Optimization File: KBOSD Doublet

Radius[mm]	Sep. [mm]	Clear diameter	Material
41.435		20.00	
	4.143		BAF3
-41.435		20.00	
	0.00		Air
-38.556		20.00	
	2.0		SF15
Plane		20.00	
	100.00		Air
Image Plane		3.57	

5.2.3 Optimization of the Parallel Plane Plates Start System Doublet Lens

Displayed below Fig. 5.2.3.1 are the KBOSD and the parallel plane plates start system doublets.

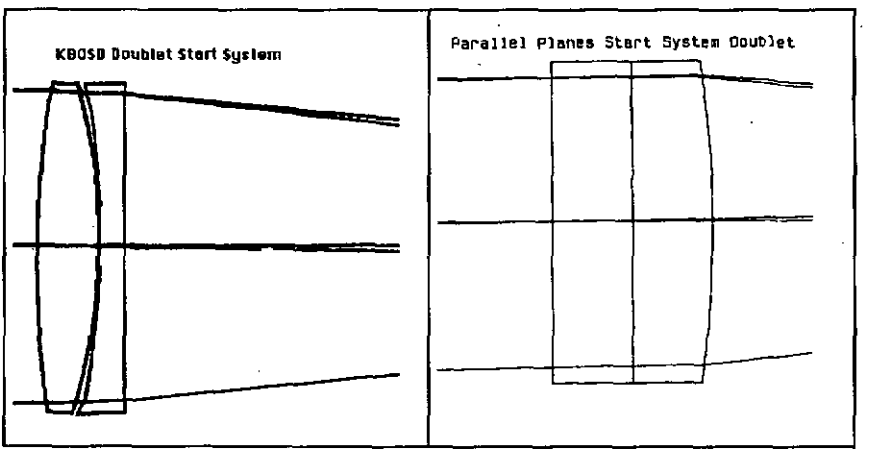


Fig. 5.2.3.1 Drawing of the KBOSD (left) and the Parallel Plane Plates(right) Start System Doublet before the Optimization.

During the optimizations of the parallel plane plates start system the curvature radii and the separations(thicknesses) are variable. The optical glasses are fixed.

After twenty eight iterations the merit functions converges. Displayed below is the data of the parallel plane plates start system after optimization

EFL= 100.000 [mm]
Parallel Planes Start system Doublet after optimization

Radius[mm]	Sep. [mm]	Clear diameter	Material
92.127		20.00	
	1.60		BAF3
-922.083		20.00	
	45.20		Air
569.555		20.00	
	96.39		SF15
-74.999		20.00	
	23.76		Air
Image Plane		3.57	

The parallel plane plates start system doublet lens is optimized to two bi-convex singlet lenses separated with a distance of 45.2 mm. The radius of the first bi-convex singlet is 92.127 mm and the second radius is -922.083 mm. The thickness of the first bi-convex singlet is 1.6 mm and its optical glass still baf3. The second bi-convex singlet lens has a thickness of 96.39 mm, it is made of the sf15 optical glass. The first curvature radius of the second bi-convex doublet is 569.555 mm and the second radius is -74.999 mm. The second bi-convex singlet is not appropriated in this situation because of the high value of its thickness. Using a singlet lens with a thickness of 96.39 mm one losses optical glass causing the cost to rise. The second problem of this optimized doublet is the shallow thickness which is 1.60 mm of the first bi-convex lens comparatively with its diameter which is 20 mm. This example demonstrates the incapacity of the optimization program to generate a doublet lens from a parallel plane plates start system, even in helping him by introducing the corrected optical glass, a reasonable thicknesses and the use of the angle solve function. The last inconvenient of this doublet is separation of 45.20 mm between the two bi-convex singlet. There is no reason to introduce such as separation.

Reported below in Fig. 5.2.3.2 is the parallel plane plates start system doublet lens after optimization using the PC-Sigma [Ref. 17].

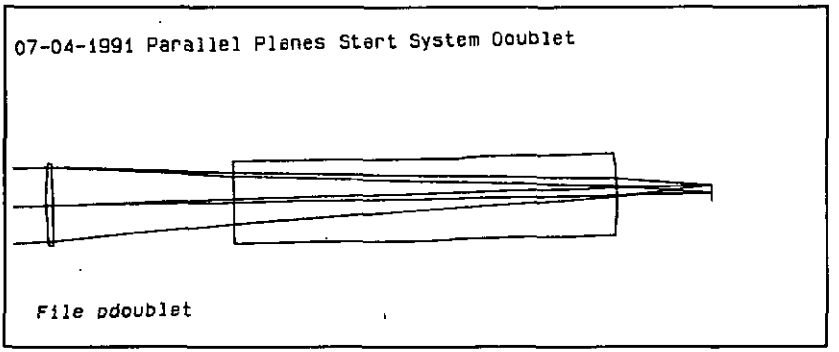


Fig. 5.2.3.2 Parallel Plane Plates Start System Doublet Lens after Optimization.

5.2.4 Optimization of the KBOSD Start System Doublet Lens

The optimization of the KBOSD start system doublet lens is done under the same conditions as the parallel plane plates start system doublet lens. The curvature radii and the thickness are variable and the optical glasses are fixed. Note that the thickness is heuristically pre-designed with the KBOSD.

After five iterations by the optimization program, the merit function converges to zero, the data of the start system doublet lens displayed above, in section 5.2.2 are optimized to the data displayed below.

EFL= 100.000 [mm]
KBOSD Start System Doublet Lens after Optimization File: Doublet Lens

Radius[mm]	Sep. [mm]	Clear diameter	Material
64.939		20.00	
	3.698		BAF3
-35.204		20.00	
	0.000		Air
-35.892		20.00	
	2.000		SF15
-174.687		20.00	
	97.184		Air
Plane		3.54	

Reported below is the drawing of the KBOSD start system doublet lens after optimization, Fig. 5.2.4.

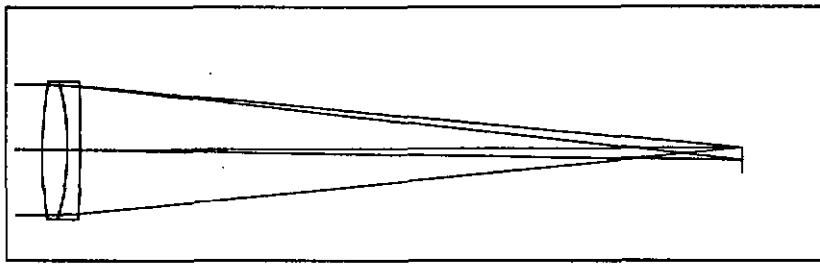


Fig. 5.2.4 Drawing of the KBOSD Start System Doublet Lens after Optimization.

The results of the optimization of KBOSD start system doublet lens is similar to the KBOSD start system before the optimization. During the optimization process of the KBOSD doublet, only a small adjustments of the curvature radii are introduced. After the optimization the KBOSD doublet kept the same geometrical form as before the optimization. The KBOSD doublet still composed of a bi-convex and plano-concave singlet lenses. The thicknesses of the doublet still constant. The fact that the KBOSD pre-designed heuristically the start system doublet lens explains the small changes during the optimization. The heuristically pre-designeddoublet lens is near a minimum of the merit function, because of these reasons that the merit function converges rapidly. In this particular situation the KBOSD start system doublet lens remember us the Fraunhofer doublet or the Steinheil doublet [Ref. 20].

5.2.5 Comparison of the Parallel Plane Plates Start System Doublet Lens with the Doublet Lens Generated by the KBOSD

In spite of the unusable design of the parallel plan plates doublet after optimization one analyses and compares it with the KBOSD doublet after optimization. The analysis is the same for the KBOSD doublet and the parallel plane plates doublet. These analytic conditions are:

- clear aperture is 19 mm
- field angles are 0º(axis) and 1º
- aberrations scale is 20 μm for the parallel plane plates doublet.
- aberrations scale is 10 μm for KBOSD doublet
- spatial frequency is 100 lines/mm
- wavelength is 588 nm.

The results of this analysis are reported and commented on below.

5.2.5.1 Comparisons of the Transverse Ray Aberrations

In this section one reports the plots of the transverse ray aberrations as a function of aperture, as shown below. Figure 5.2.5.1.1 represents the transverse ray aberrations of the KBOSD doublet after optimization. This transverse ray aberrations are at the field angle 0º(axis) and 1º respectively in the sagittal and tangential image plane. The scale of the transverse ray aberrations is 0.01 mm.

Figure 5.1.5.1.2 represents the transverse ray aberrations of the parallel plane plates start system doublet lens after optimization. The transverse ray aberrations is plotted at the field angle 0º(axis) and 1º respectively in the sagittal and tangential image plane. Note that the scale of the transverse ray aberrations is 0.02 mm.

According to the transverse ray aberrations analysis displayed in Fig. 5.1.5.1.1 and the other one displayed in Fig. 5.1.5.1.2 and taking into account the scale factor one may conclude that the KBOSD generated doublet lenses of better quality than the parallel plane plates start system doublet lens. This judgement is done in terms of transverse ray aberrations.

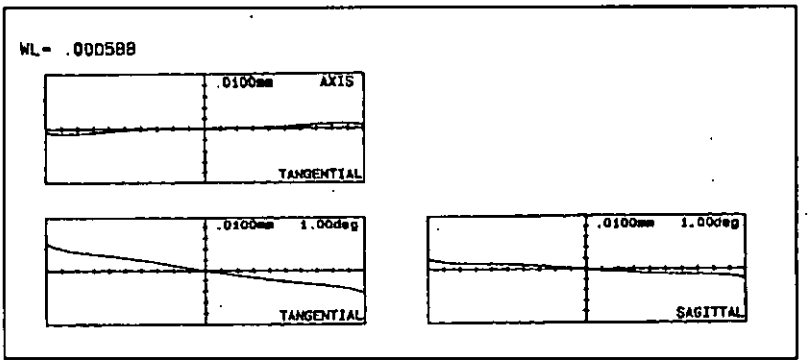


Fig.5.2.5.1.1 Transverse Ray aberrations of the KBOSD Start System Doublet Lens after Optimization.

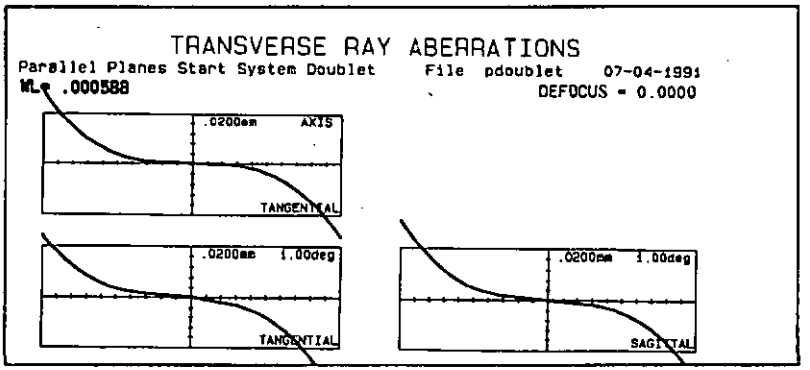


Fig. 5.2.5.1.2 Transverse Ray Aberrations of the Parallel Plane Plates Start System Doublet Lens after Optimization.

5.2.5.2 Comparisons of the Spot Diagrams

In this section an analysis is being made of the spot diagrams of the KBOSD start system doublet lens and the parallel plane plates doublet start system. The KBOSD doublet spot diagrams and the parallel plane plates start system doublet has been done under the same following conditions:

- wavelength is 588 nm
- field angles positions 0º(axis) and 1º
- scale 20 μm.

Displayed below in Fig. 5.2.5.2.1 are the spot diagrams of the KBOSD doublet after optimization.

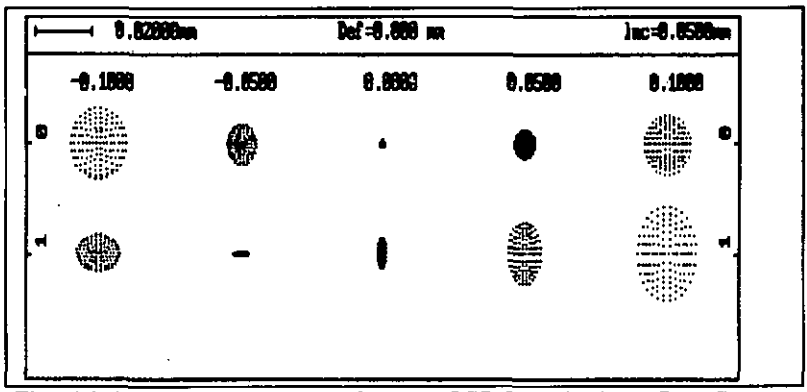


Fig. 5.2.5.2.1 Spot Diagrams of the KBOSD Doublet Lens Start System after Optimization.

Displayed below Fig. 5.2.5.2.2 are the spot diagrams of the parallel plane plates start system doublet after optimization. This spot diagrams is done under the following parameters:

- wavelength is 588 nm
- field angles positions 0º(axis) and 1º
- scale 20 μm.

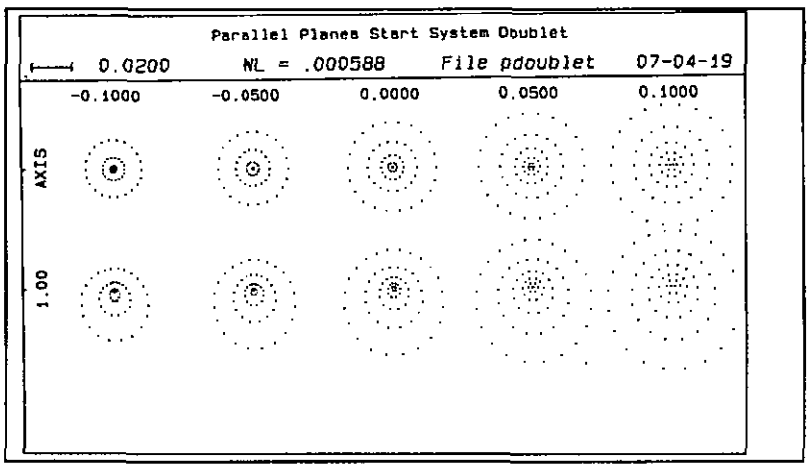


Fig. 5.2.5.2.2 Spot Diagrams of the KBOSD Parallel Plane Plates Start System Doublet Lens after Optimization.

Comparing the spot diagrams of the KBOSD doublet displayed in Fig. 5.1.5.2.1 with the spot diagrams of the parallel plane plates start system doublet displayed in Fig. 5.1.5.2.2 one concludes that the KBOSD doublet has a better quality.

5.2.5.3 Comparisons of the Geometrical MTF

In this section one reports on the geometrical MTF of the KBOSD start system doublet and the parallel plane plates start system doublet after optimization. Both geometrical MTF diagrams displayed in figure 5.2.5.3 are plotted at the same following parameters:

- field angles 0º(axis) and 1º
- spatial frequency 100 lines/mm
- full aperture.

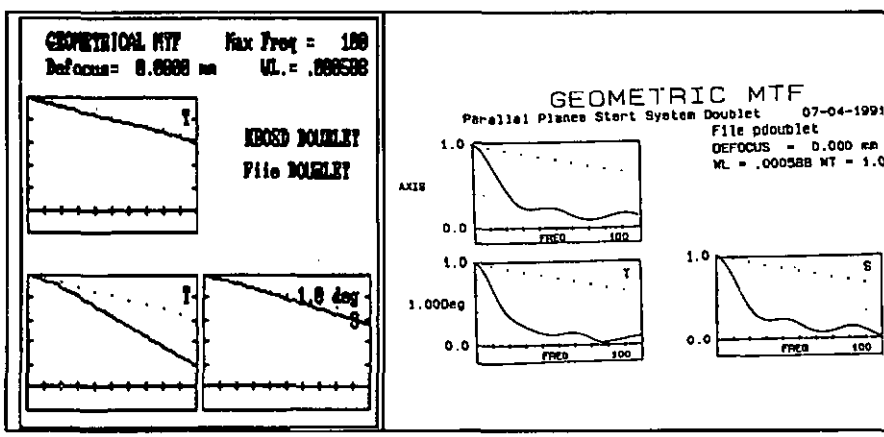


Fig.5.2.5.3 Geometrical MTF of the KBOSD Doublet Lens(left) and Geometrical MTF of the Parallel Plane Plates Start System(right) after optimization.

According to the spot diagrams plotted in figure 5.2.5.3 it can be said that the KBOSD doublet lens has a greater resolution than the parallel plane plates start system doublet lens after optimization.

5.3 Discusions and Comparisons of Triplets Lens

In this section one discusses data and comparisons of two triplet lenses. The first triplet is proposed by the KBOSD as a start system. The second triplet is optimized from a parallel plane plates start system. Both triplets are designed under the following parameters:

- wavelength of 546 nm
- correction for the wavelength 546 nm, 656 nm and 486 nm.
- focal length of 100.0 mm
- diameter of 26 mm
- an entrance pupil diameter of 25 mm.
- numerical aperture is 0.125
- object is supposed to be at infinity.

5.3.1 Parallel Plane Plates Start System Triplet Lenses

As explained in sections 5.1.1 and 5.2.1 the optimization program is incapable of starting the optimization of a triplet lens from parallel plane plates. One has to help the optimization program by using the angle solve function. In order to get the expected numerical aperture and focal length, one uses the angle solve function of the optimization program to change the curvature radius of the last surface. The second help to the optimization program one introduces the optical glasses, the thicknesses and the separations between the plane plates. The optical glasses are bak5, sf18 and sf14. These optical glasses are proposed by the KBOSD and lead to a chromatically corrected triplet lens. Each plan plate has a thickness of 5 mm and a clear diameter of 25 mm. The separations between the plane plates are zero.

Displayed below, are the data of the parallel planes plates triplet lens before optimization.

EFL = 100.000 [mm]
Parallel Plane Plates Start System Triplet Lens before Optimization

Radius[mm]	Sep. [mm]	Clear diameter	Material
Pupil		25.00	
	0.000		Air
Plane		25.00	
	5.00		BAK5
Plane		25.00	
	0.000		Air
Plane		25.00	
	5.00		SF18
Plane		25.00	
	0.000		Air
Plane		25.00	

	5.00		SF14
-76.861		25.00	
	100.00		Air
Image Plane		11.05	

Figure 5.3.1 represents the drawing of the parallel plane plates start system triplet lens before optimization as described above.

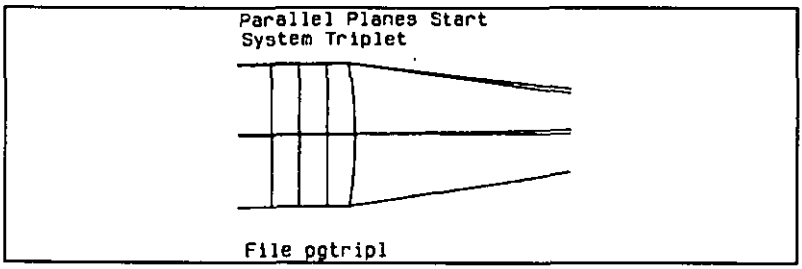


Fig. 5.3.1 Drawing of the Parallel Plane Plates Start System Triplet Lens before Optimization.

5.3.2 KBOSD Start System Triplet Lens

Under the same starting conditions: design wavelength is 546 nm, aberrations corrections at the wavelength 546 nm, 656 nm and 486 nm, focal length equal 100.0 mm, diameter of 26 mm, entrance pupil diameter of 25 mm, numerical aperture is 0.125 and the object is said to be at infinity the KBOSD proposed the triplet lens displayed below. Note that these starting conditions are the same for the parallel plane plates triplet lens.
Displayed below, are the data of the KBOSD start system triplet lens before optimization.

EFL = 100.000 [mm]
KBOSD Triplet Lens before Optimization F=100 [mm]

Radius[mm]	Sep. [mm]	Clear diameter	Material
Plane		25.00	
	0.000		Air
47.50		25.00	
	4.840		BAK5
-47.50		25.00	
	0.000		Air
-26.15		25.00	
	1.500		SF18
26.15		25.00	
	0.000		Air
36.76		25.00	
	5.804		SF14
-36.76		25.00	
	93.000		Air
Image Plane		3.50	

To get this KBOSD triplet one has to type the following predicate: *calcul(100.0, Magnification, l, 93.0, z, z')* then the KBOSD answers by displaying the data reported above.

The triplet proposed by the KBOSD is composed of two bi-convex singlet lenses and a bi-concave singlet lens. The first bi-convex singlet lens is made of bak5 glass, it has two curvature radii of 47.5 mm and -47.5 mm, a thickness of 4.84 mm and a clear diameter of 25 mm. The bi-concave singlet is made of sf18 optical glass; it has two curvature radii of -26.15 mm and 26.15 mm, a center thickness of 1.5 mm and clear diameter of 25 mm. The last bi-convex singlet lens is made of sf14 glass, it has two radii of 36.76 mm and -36.76 mm, a thickness of 5.804 mm and a clear diameter of 25 mm.

This KBOSD start system triplet should be perfectly corrected for the chromatic aberrations.

Plotted below is the drawing of KBOSD start system triplet lens. The design of this triplet emphasizes the geometrical form of the Tessar triplet.

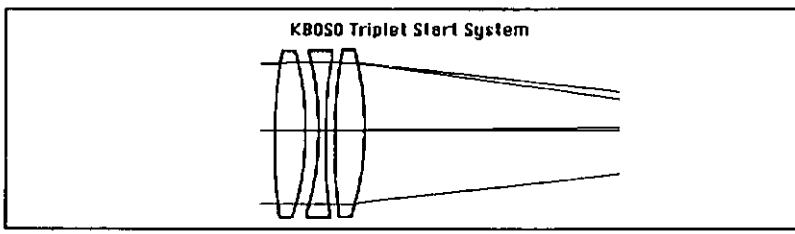


Fig. 5.3.2 Drawing of the KBOSD Start System Triplet Lens before Optimization.

5.3.3 Optimization of the Parallel Plane Plate Start System Triplet Lens

In order to point out the differences between the KBOSD start system triplet and the parallel plane plates start system triplet we display their designs on the same table.

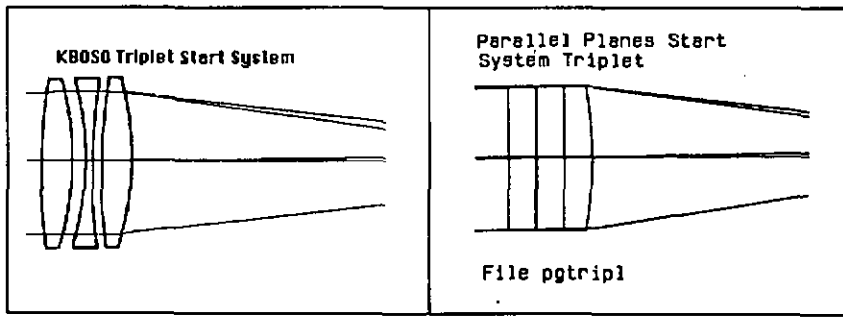


Fig. 5.3.3.1 Drawing of the KBOSD (left) and the Parallel Plane Plates(right) Start System Triplet before the Optimization.

During the optimization of the parallel plane plates one leave the curvature radii and the separations variables and the optical glasses fixed.

After twenty five iterations the merit function of the optimization converges and the optimization program stops. Displayed below, are the data of the parallel plane plates triplet lens after optimization.

EFL = 100.000 [mm]
Parallel Plane Plates Start System after Optimization File: PGTriplet Lens

Radius[mm]	Sep. [mm]	Clear diameter	Material
45.875		25.00	
	30.80		BAK5
-46.438		25.00	
	0.000		Air
-203.056		25.00	
	1.500		SF18
-986.100		25.00	
	3.51		Air
-34.180		25.00	
	21.76		SF14
-79.732		25.00	
	62.73		Air
Image Plane		3.51	

The parallel plane plates triplet lens after optimization is composed of a bi-convex singlet and two meniscus singlet lenses. The bi-convex singlet lens has two radii of 45.875 mm and -46.438 mm and a thickness of 30.80 mm !!. Unfortunately, this singlet is so thick that is unusable. With the exception of the inconvenience of the thickness, this bi-convex singlet is similar to the bi-convex singlet of the KBOSD triplet start system(displayed in section 5.3.2) before the optimization. The second meniscus singlet of this optimized triplet lens is made of sf14 glass, it has two curvature radii of -34.180 mm and -

79.732 mm and a thickness of 21.76 mm. This thickness of 21.76 mm is excessive and make the meniscus also unusable.

Displayed below Fig. 5.3.3.2 is the parallel plane plates start system triplet lens after optimization.

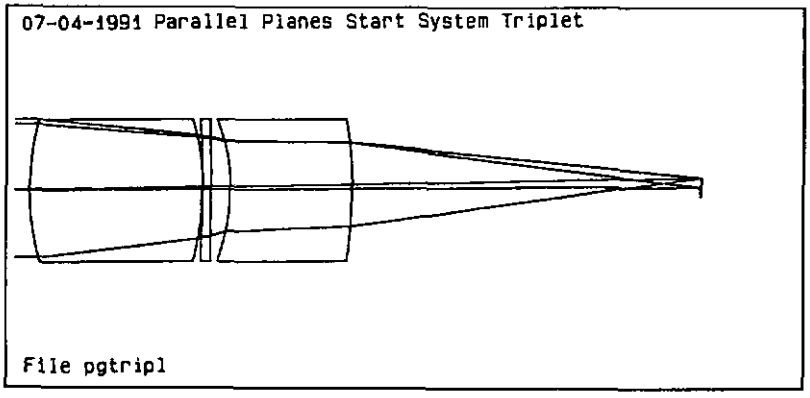


Fig. 5.3.3.2 Parallel Plane Plates Start System triplet Lens after Optimization.

5.3.4 Optimization of the KBOSD Start System Triplet Lens

Displayed below, are the data of the KBOSD start system triplet lens after optimization.

EFL = 100.000 [mm]
KBOSD Triplet Lens after Optimization File: Triplet Lens

Radius[mm]	Sep. [mm]	Clear diameter	Material
Plane		25.00	
	0.000		Air
66.282		25.00	
	4.840		BAK5
-47.787		25.00	
	0.000		Air
-48.988		25.00	
	1.500		SF18
-233.656		25.00	
	0.000		Air
144.267		25.00	
	5.804		SF14
200.440		25.00	
	92.471		Air
Plane		3.50	

Drawing of the KBOSD Triplet Lens after Optimization

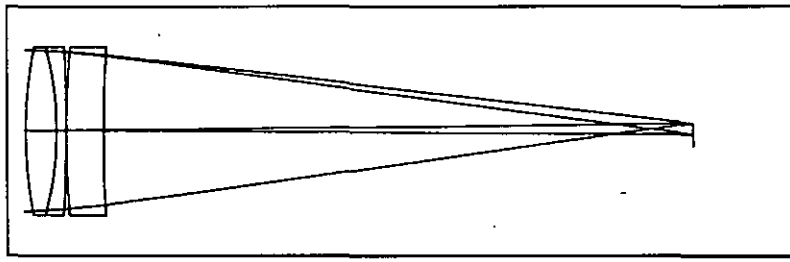


Fig. 5.3.3.2 KBOSD Start System triplet Lens after Optimization.

5.3.5 Comparisons of the Parallel Plane Plates Start System Triplet Lens with the Triplet Lens Generated by the KBOSD

The parallel plane plates start system triplet lens after optimization is analyzed in the same conditions as the triplet lens generated by the KBOSD after optimization. The analysis conditions are displayed below:

- spatial frequency is 100 lines/mm
- focal length is 100 mm
- wavelengths are 546 nm, 656 nm and 486 nm
- aberrations scale is 0.025 mm
- field angle 0º(on axis) and 1º.

The results of this analysis are reported in this section in order to compare the optical quality of the triplet optimized from a parallel plane plates with the KBOSD triplet after optimization.

5.3.5.1 Comparisons of the Transverse Ray Aberrations

Figure 5.3.5.1.1 reported below represents the transverse ray aberrations of the KBOSD triplet lens after optimization. The KBOSD triplet lens transverse ray aberrations are plotted at the following parameters:

- wavelength are 486 nm, 546 nm and 656 nm
- field angle 0º and 1º respectively in the sagittal and tangential planes.
- aberrations scale is 0.025 mm.

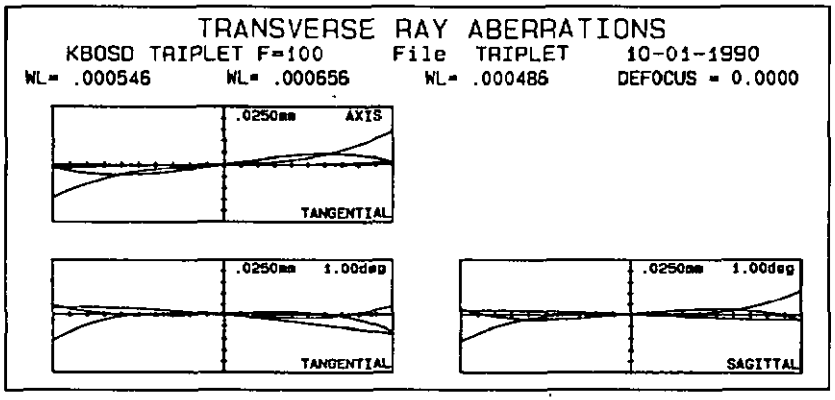


Fig.5.3.5.1.1 Transverse Ray aberrations of the KBOSD Start System Triplet Lens after Optimization.

Figure 5.3.5.1.2 reported below represents the transverse ray aberrations of the parallel plane plates triplet lens after optimization. The parallel plane plates triplet transverse ray aberrations are plotted at the wavelength 486 nm, 546 nm and 656 nm at the field angle 0º and 1º respectively in the sagittal and tangential planes. The aberrations scale is 0.025 mm.

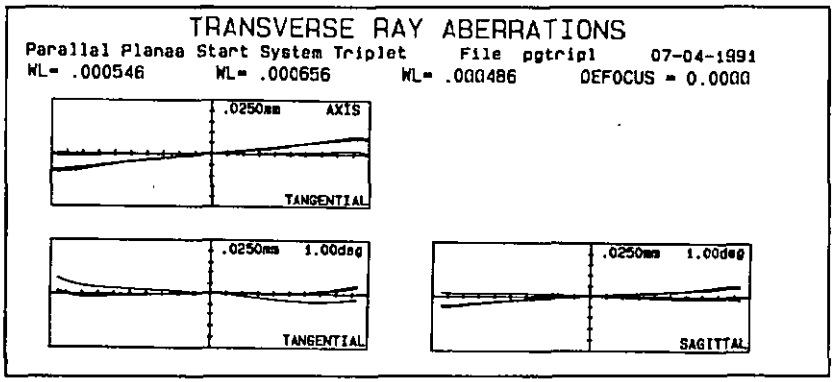


Fig. 5.3.5.1.2 Transverse Ray Aberrations of the Parallel Plane Plates Start System Triplet Lens after Optimization.

According to the transverse ray aberrations curves plotted in Fig. 5.3.5.1.2 above one observes a vignetting effects at the field angle position 1º in the sagittal and tangential diagrams. the fact that the aberrations curves are don't attempt the border of the aperture is typically the sign of the vignetting effect. This vignetting aberrations is to avoid. With the exception of the

vignetting effect the quality of the plane parallel plates optimized triplet lens is comparable to the quality to the KBOSD start system triplet lens.

5.3.5.2 Comparisons of the Spot Diagrams

Figure 5.3.5.2.1 below represents the spot diagrams of the KBOSD Triplet lens start system after optimization and figure 5.3.5.2.2 represents the spot diagrams of the parallel plane plates start system after optimization. These spot diagrams have been made within the following parameters:

- wavelength 546 nm
- scale 0.025 mm
- field angle 0º and 1º.

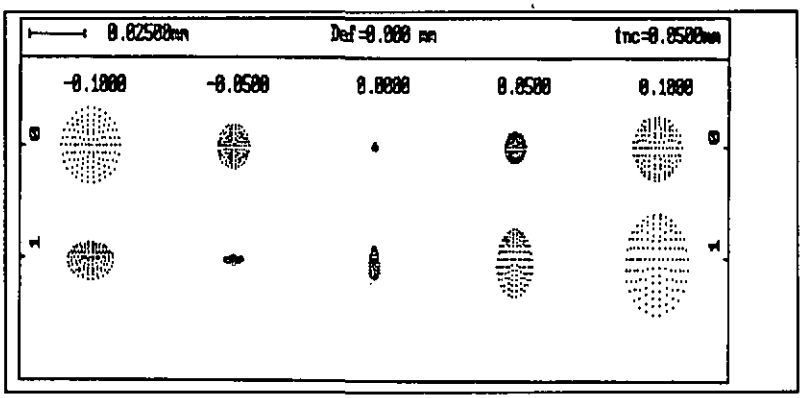


Fig. 5.3.5.2.1 Spot Diagrams of the KBOSD Triplet Lens Start System after Optimization.

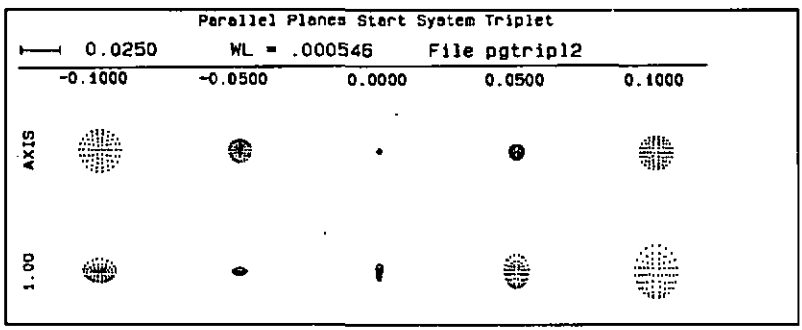


Fig. 5.3.5.2.2 Spot Diagrams of the KBOSD Parallel Plane Plates Start System Triplet Lens after Optimization.

According to the spot diagrams plotted in figure 5.3.5.2.1 and figure 5.3.5.2.2 one may conclude that the optimized parallel plane plates start system triplet lens has an optical quality(in terms of spot diagrams) comparable to the KBOSD start system triplet lens.

5.3.5.3 Comparisons of the Geometrical MTF

In this section one reports the geometrical MTF of the KBOSD start system triplet lens after optimization and also the parallel plane plates start system triplet lens after optimization. The geometric MTF diagrams displayed in figure 5.3.5.3 are plotted at the following parameters:

- field angle 0º and 1º
- spatial frequency 100 lines/mm
- full aperture.

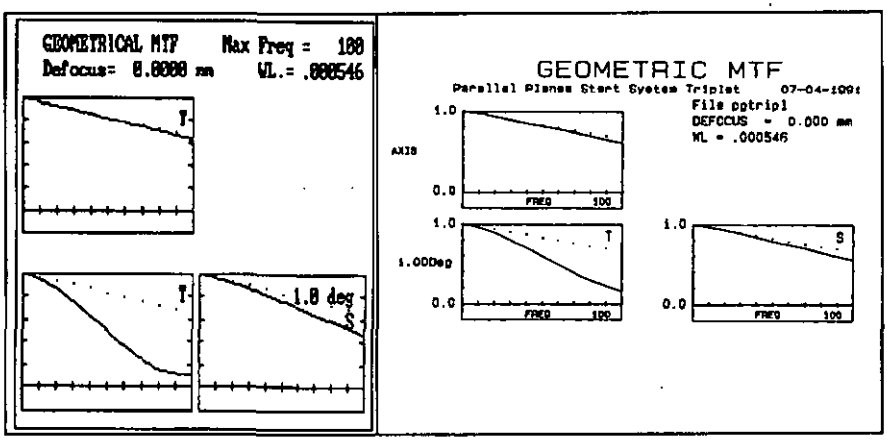


Fig.5.3.5.3 Geometrical MTF of the KBOSD Triplet Lens(left) and Geometrical MTF of the Parallel Plane Plates Start System Triplet Lens(right) after optimization.

According to the diagrams plotted in figure 5.3.5.3 above one concludes that the parallel plane plates start system triplet lens after optimization appears to have a better quality than the triplet optimized from the KBOSD start system triplet lens. One has to remember that the separation between the parallel plane plates triplet lens after optimization and the image plane is only 62.75 mm(see section 5.3.3 above); the same separation is 92.471 mm by the KBOSD start system triplet lens after optimization(see 5.3.4 above). Because of the separation of 62.75 mm, comparatively to the separation of 92.471

mm that the optimization of the parallel plane plates start system has a better quality than the KBOSD start system triplet lens. In any case the triplet obtained from the optimization of the parallel plane plates start system is unusable because of the large value of its thickness.

5.4 Discusions of a Triplet Lens Obtained from Parallel Plane Plates and Good Mastering of the Optimization Programs

As mentioned above in section 5.3, the major inconvenience of the parallel plane plates triplet lens after optimization is the high value of its thickness. In this section one demonstrates the possibility of obtaining a good quality lens from plan parallel plates, using the existing commercial optimization program. In fact, the lens design optimization programs allows the same parametric control during the optimization, such as the lens thickness, the lens length(distance from the object plane to the image plane) and the numerical aperture....
The triplet lens treated in this section is obtained from parallel plane plates respecting the following steps:

- start from parallel plane plates as displayed above in section 5.3.1
- input three optical glasses chromatically corrected, bak5, sf18 and sf14 in our case
- use the angle solve function to change the last surface of the parallel plane plates to get the expected focal length, 100 mm in our case
- input the total length(distance from the first surface to the image plane) of the triplet lens, 106 mm in our case
- input the distance from the last surface to the image plane
- the curvature radii are variable
- thicknesses and separations are variable
- optical glasses are fixed.

After forty iterations we stoped(intuitively) the optimization program. The results obtained from the optimization of the parallel plane plates triplet are comparable to the triplet obtained from the KBOSD triplet start system after optimization.

Starting from parallel plane plates and trying to attempt the same results as the KBOSD start system, one has to execute a preliminary work in searching the corrected optical glasses, in calculating the lens thickness... and then one has to wait a high number of iterations. In the other side, the KBOSD proposes the corrected optical glasses and calculates the optimal thickness, the expected focal length as well as the other parameters of the lens; these facts lead rapidly to an optimized lens. Introducing the chromatic corrected optical

glasses, lens thickness... a special knowledge of the optical design is needed. In our case this optical design knowledge is memorized in the KBOSD.

5.4.1 Data of the Parallel Plane Plates Triplet Lens

Displayed below are the data of a triplet lens obtained from parallel plane plates and good control of the optimization program. These data are obtained after optimization.

EFL = 100.000 [mm]
KBOSD Triplet Lens after Optimization File: Triplet Lens

Radius[mm]	Sep. [mm]	Clear diameter	Material
Plane		25.00	
	0.00		Air
53.737		25.00	
	6.34		BAK5
-60.000		25.00	
	0.00		Air
-71.158		25.00	
	6.12		SF18
-93.094		25.00	
	4.19		Air
-54.706		25.00	
	5.66		SF14
-121.458		25.00	
	84		Air
Plane		3.50	

This triplet lens is composed of bi-convex singlet lens and two meniscus singlet lenses. Displayed below is the drawing of triplet obtained from the parallel plane plates start system and good control of the optimization program.

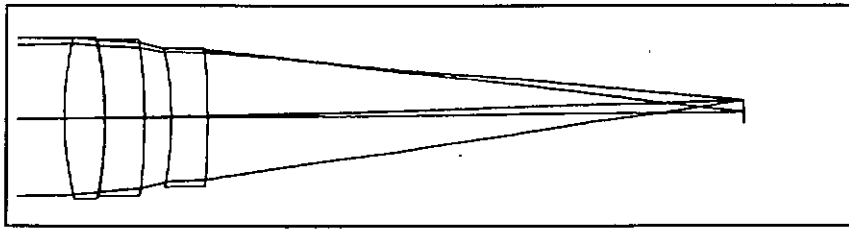


Fig. 5.4.1.1 Drawing of Triplet Obtained from Parallel Plane Plates Start System and Good Control of the Optimization Program

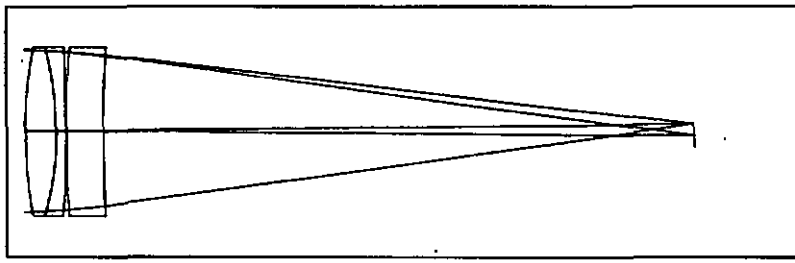


Fig. 5.4.1..2 KBOSD Start System triplet Lens after Optimization.

The triplet lens displayed in Fig. 5.4.1.2 above is described in detail in section 5.3.2.

5.4.2 Transverse Ray aberrations of the Parallel Plane Plates Triplet Lens

In this section one displays the transverse ray aberrations of the triplet lens obtained from parallel plane plates and good control of the optimization program.

The transverse ray aberrations displayed in Fig. 4.4.2.1 and Fig. 4.4.2.2 are plotted at the following parameters:

- wavelength are 486 nm, 546 nm and 656 nm
- field angle 0° and 1° respectively in the sagittal and tangential planes.
- aberrations scale is 0.025 mm.

According to the transverse ray aberrations displayed in Fig. 4.4.2.1 and the transverse ray aberrations plotted in Fig. 4.4.2.2 one concludes that the transverse ray aberrations of the KBOSD start system triplet lens is comparable to the transverse ray aberrations of the optimized parallel plane plates start system triplet lens.

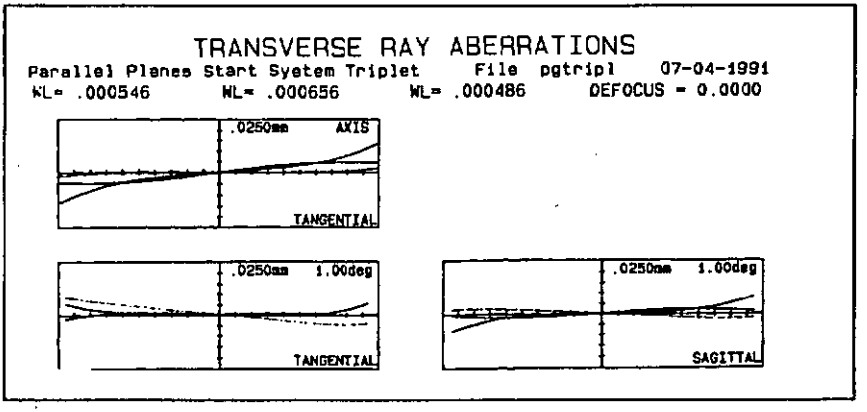


Fig. 4.4.2.1 Transverse Ray Aberrations of the Triplet Lens Obtained from Parallel Plane Plates and Good Mastering of the Optimization Program.

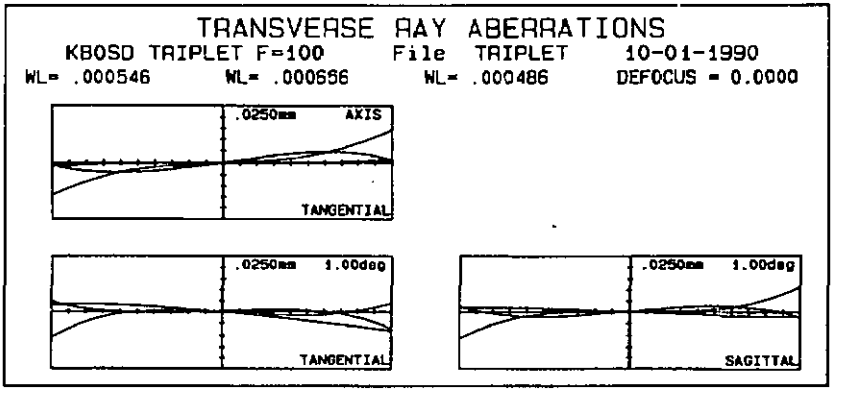


Fig.5.4.2.2 Transverse Ray aberrations of the KBOSD Start System Triplet Lens after Optimization.

5.4.3 Spot Diagrams of the Parallel Plane Plates Triplet Lens

In this section one displays the spot diagrams of the triplet lens obtained from parallel plane plates and good control of the optimization program.
Figure 5.4.3.2 below represents the spot diagrams of the KBOSD Triplet lens start system after optimization and figure 5.4.3.1 represents the spot diagrams of the triplet lens obtained from parallel plane plates start system and good control of the optimization program after optimization. These spot diagrams have been done within the following parameters:

- wavelength 546 nm
- scale 0.025 mm
- field angle 0º and 1º.

One notes the similarity between the spot diagrams displayed in Fig. 5.4.3.1 and the spot diagrams displayed in Fig. 5.4.3.2.

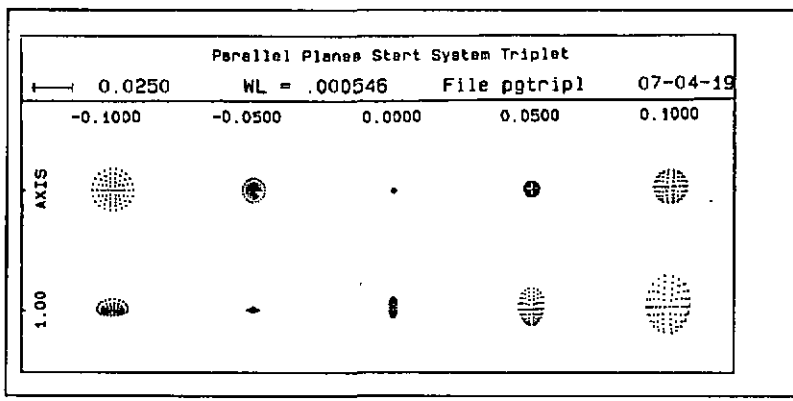


Fig. 5.4.3.1 Spot Diagram of the Triplet Lens Obtained from parallel plane plates and Good Mastering of the Optimization program after Optimization.

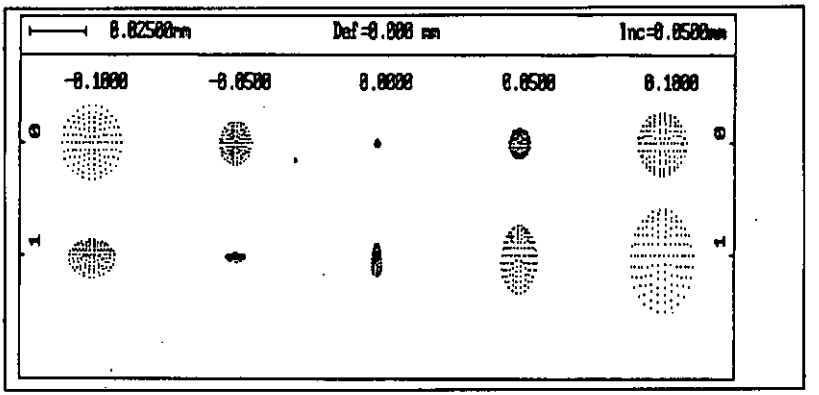


Fig. 5.3.4.2 Spot Diagrams of the KBOSD Parallel Plane Plates Start System Triplet Lens after Optimization.

5.4.4 Geometrical MTF of the Parallel Plane Plates Triplet Lens

In this section one displays the geometrical MTF of the triplet lens obtained from parallel plane plates and good control of the optimization program. Fig. 5.4.4(left) represents the geometric MTF of the KBOSD start system triplet lens after optimization. Fig. 5.4.4(right) represents the triplet lens obtained from parallel plane plates start system and good control of the optimization program. The geometric MTF diagrams displayed in figure 5..4.4 are plotted at the following parameters:

- field angle 0º and 1º
- spatial frequency 100 lines/mm
- full aperture.

From the comparison of the geometric MTF of the KBOSD triplet lens after optimization (left) with the geometric MTF of the parallel plane plates after optimization one may conclude that both optical system have comparative resolution.

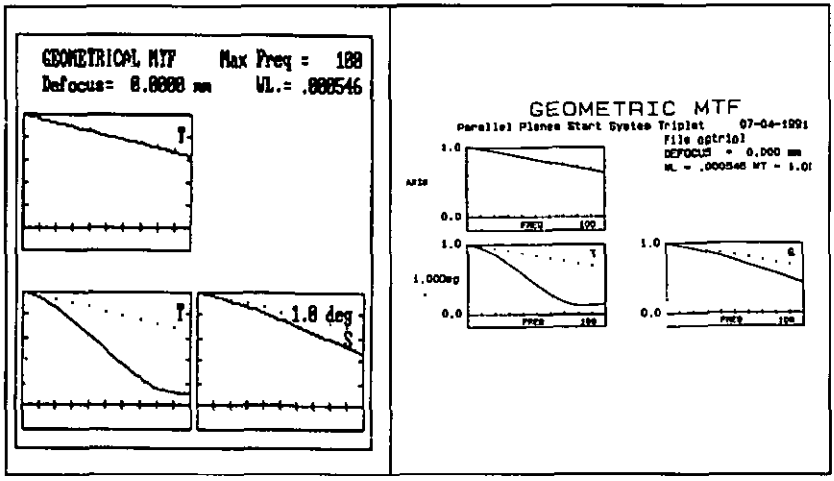


Fig.5.4.4 Geometrical MTF of the KBOSD Triplet Lens(left) and Geometrical MTF of the Triplet Lens Obtained from Parallel Plane Plates Start System(right) and Good Mastering of the Optimization Program.

5.5 Conclusion

In this chapter one optimized and compared several optical systems proposed by the KBOSD with optical systems generated from parallel plane plates. Starting from parallel plane plates one demonstrates the incapacity of the used optimization program of starting the optimization. Also one has to help the optimization program by using the angle solve function and by introducing the appropriated optical glass. On the other hand the KBOSD proposes start optical systems ready to be optimized using an optimization program. Contrarily to parallel plane plates start systems, the start systems proposed by the KBOSD leads rapidly (less number of iterations) to optimal optical system and contain the appropriated optical parameters such as optical glasses, lens thickness....
From this comparative study one concludes that the optical start systems proposed by the KBOSD and optimized are of a better quality than the optical start system obtained from the optimization of parallel plane plates.

However it is possible to obtain optical systems from the optimization of parallel plane plates with an optical quality comparable to the optical systems obtained from the optimization of the KBOSD start systems. This statement is illustrated by the triplet lens example displayed in section 5.4. To attempt the same results as the KBOSD start systems after optimization from parallel plane plates one has to master the optimization program and to execute a

preliminary work. This preliminary work is dedicated for the start system predesign such as the calculation of the thickness and the selection of the optical glasses.... By executing this preliminary work one has to be equipped with optical design knowledge. In our case the KBOSD representes the optical knowledge.

6. Quality of the Knowledge Based Optical System Design Starting Systems and their Influence on the Final Optimization Results

6.0 Introduction

6.1 Comparative Table and Discussions of the Knowledge Based Optical System Design Doublet

6.2 Comparative Table and Discussions of the Knowledge Based Optical System Design Singlet

6.3 Comparative Table and Discussions of the Knowledge Based Optical System Design Triplet

6.0 Introduction

In this section one explains why the KBOSD start systems lead to a good quality optical systems after optimization. This explanations are illustrated by three KBOSD optical start systems, a KBOSD doublet, a KBOSD singlet and a KBOSD triplet. The quality of the starting systems are illustrated in this chapter in terms of the third order aberrations. Comparing the third order aberrations of the KBOSD starting systems before and after the optimization, one may conclude the influence of the KBOSD start system on final optimization results. High order aberrations may have influences on the final results of the KBOSD optical systems but in this discussion one take only the order aberrations into account.

6.1 Comparative Table and Discussions of the KBOSD Doublet

In this section one reports the summa of the third order aberrations of a doublet lens proposed by the KBOSD before optimization and after optimization. The third order aberrations of the KBOSD doublet lens are compared with the third order aberrations of the Kingslake doublet lens. The KBOSD doublet start system before and after optimization and the Kingslake doublet are treated in detail in chapter 4, section 4.1 of this work. The values of the third order aberrations displayed below are expressed in mm.

Lens	KBOSD doublet start system	Kingslake doublet	KBOSD doublet start system after optimization
Spherical aberrations	-0.045033	0.001048	-0.000423
Coma	0.002924	-0.000031	-0.000031
Astigmatism	0.000291	0.000295	0.000265
Field curvature	0.000195	0.000210	0.000193
Distortion	-0.000000	0.000002	0.000002
Longitudinal chromatic aberrations	0.003335	0.000178	0.003336
Transverse chromatic aberrations	0.000009	-0.000004	0.000009

Table 6.1 Comparative Table of the Third Order Aberrations of the KBOSD Doublet and Kingslake Doublet.

According to the second column (KBOSD doublet start system) of the comparative table above one may conclude that the KBOSD start system is potentially a good doublet. This statement is based on the low values of the

third order aberrations of this start doublet. The values of the third order aberrations of the KBOSD start system before the optimization are comparable to the values of the Kingslake doublet.
After optimization the KBOSD doublet has a better quality than the Kingslake doublet. With the exception of the chromatic aberrations, the values of the aberrations of the KBOSD doublet, after optimization are less or equal to the values of the aberrations of the Kingslake doublet.

In the other on the other hand it is necessary to point out that many third order aberrations such as astigmatism, field curvature, distortion and chromatic aberrations of KBOSD doublet start system remain the same before and after the optimization. This fact resulted from the rules used by the KBOSD to predesign optical design start systems.

Empirically, a doublet lens is composed of two singlet lenses and two types of optical glass. One singlet lens has a positive focal length and the second one has a negative focal length. The optical glass is made of crown type and flint type. The geometrical form of the two singlet lenses is an important parameter as well as their thickness and their curvature radii. The facts and rules mentioned above are used by the KBOSD in order to generate a potentially good optical design start systems. Because of these reasons the KBOSD start system leads to a good quality optical system after optimization.

6.2 Comparative Table and Discussions of the KBOSD Singlet

The KBOSD singlet lenses discussed in this chapter are treated in more detail in chapter 4.2.
According to the comparative table displayed below, one may conclude that there is no larges differences in the third order aberrations before the optimization and after the optimization. This fact demonstrates that the KBOSD start systems are good pre-designed and will lead to good quality optical systems after optimization. To prove this statement in one note, simply by proposing a singlet lens, the KBOSD calculated the curvature radii, selected the adequated optical glass and calculated the singlet thickness according to physical laws. The KBOSD used some heuristics concerning the geometrical form and the conditions of use of the singlet lenses to propose the appropriate singlet lenses. The geometrical form of the singlet lens proposed by the KBOSD leads empirically to a good lens element after optimization. The values of the third order aberrations displayed below are expressed in mm.

Lens	KBOSD singlet before optimization	KBOSD singlet after optimization
Spherical aberrations	0.006562	0.006756
Coma	-0.000283	-0.000255
Astigmatism	0.000881	0.000890
Field curvature	0.000526	0.000531
Distortion	0.000004	0.000003
Longitudinal chromatic aberrations	0.000002	0.000002
Transverse chromatic aberrations	0.00000	0.00000

Table 6.2 Comparative Table of the Third Order Aberrations of the KBOSD Singlet Lens before and after Optimization.

6.3 Comparative Table and Discussions of the KBOSD Triplet

In this section one demonstrates why a KBOSD optical start system triplet lens leads to a good optical system triplet after optimization. The data and the analysis of the triplet discussed in this section are treated in detail in chapter 4, section 4.4.

In table 6.3 displayed below one observes that the third order aberrations of the KBOSD triplet before and after optimization does not change significantly , excepting the spherical aberrations. The fact that the third order aberrations of the KBOSD triplet does not significantly change during the optimization demonstrates that the KBOSD start system triplet lens leads to a good triplet lens after optimization. In response to the question: Why does the KBOSD triplet have a low third order aberrations before the optimization? In response to this question, in proposing this triplet, the KBOSD:

- calculated the thicknesses of the three singlet lenses making the triplet
- selected three optical glasses chromatically corrected
- used heuristics according to the final use of the triplet
- selected the optimal geometrical form for the triplet lens.
- assembled the different optical elements(singlet lenses) to compose the triplet lens.

This triplet lens is composed of three singlet lenses: A bi-convex singlet with a positive focal length, a bi-concave singlet with negative focal length and a bi-convex singlet with positive focal length. This triplet configuration (focal length positive, focal length negative and focal length positive) recalls to us

the Tessar triplet. Empirically, a triplet with appropriated optical glasses and with a geometrical form like the Tessar triplet leads to a good triplet after optimization. The value of the third order aberrations displayed in table 6.3 below are expressed in mm. The high value of the spherical aberrations of the triplet before the optimization is provoked by the relatively high numerical aperture. The numerical aperture of this triplet is equal to 0.125. After optimization the spherical aberrations changed from -2.292748 to 0.000073.

Lens	KBOSD triplet lens before optimization	KBOSD triplet lens after optimization
Spherical aberrations	-2.292748	0.000073
Coma	-0.014928	-0.000114
Astigmatism	0.00005	0.000465
Field curvature	0.000250	0.000332
Distortion	0.000011	0.000003
Longitudinal chromatic aberrations	0.011096	0.000960
Transverse chromatic aberrations	0.000501	0.000051

Table 6.3 Comparative Table of the Third Order Aberrations of the KBOSD Triplet Lens before and after Optimization.

6.4 Conclusion

In this chapter one discusses the quality of the KBOSD starting systems and demonstrates their influence on the final optimization results. Note that the KBOSD starting systems have a low third order aberrations. In proposing a start system the KBOSD:

- calculates the thickness of the lens
- selectes the appropriate optical glass (correction of the chromatic aberrations)
- uses heuristics leading to the best geometrical form of the lens(correction of the geometrical aberrations such as coma, spherical aberrations, astigmatism)
- utilizes heuristics and rules to find the appropriate lens design according to the final use of the lens
- most lenses proposed by the KBOSD leads empirically to good optical systems after optimization.

The facts mentioned above explain why the KBOSD optical starting systems lead to good quality optical systems after optimization.

7. Conclusions

In this work a Knowledge Based Optical System Design (KBOSD) is realized; the progress in the domain of artificial intelligence helps a great deal.

Unlike the existing expert systems this KBOSD uses neither a lens library nor a lens data base; like the its name indicates it, the KBOSD is entirely based on optical design knowledge and heuristics.

With this KBOSD, one proposes a new approach for the optical design of start systems and demonstrates its feasibility and validity. The combination of two different disciplines, artificial intelligence techniques and optical design knowledge base is also new and represents an important contribution to optical design. The use of artificial intelligence to treat optical lens design is a new approach to the problem of lens design start systems.

At what point can artificial intelligence be useful in the design of optical systems? Why does one select artificial intelligence techniques to solve optical design problems? In response to these questions, one should note that it is complicated to solve optical design problems in a classical procedural program where the solution must be written. The solution of the lens design problem is based on innumerable knowledges such as heuristics, optical design rules, as well as technical and numerical constraints that should be explored. The modular structure of the optical element and its assembly typically pose a problem that can be solved by artificial intelligence algorithms. Additionally, with artificial intelligence the optical knowledge is expressed in terms of the optical user point of view. For this reason, the artificial intelligence exploration techniques are adopted.

The artificial intelligence approach helps us to attempt many goals, especially in :

- making up the base of facts containing optical glass and its properties.
- making up the lens design knowledge base containing lens design constraints and rules, as well as heuristics on optical lens design, noting that a heuristic is a method of learning that involves using reasoning and past experience rather than formulas or solutions that are given.
- making up optical design knowledge models(formal and informal) such as:

    - heuristics and the context of use of optical systems are informal
    - algebraic models of optical systems are empirically
    - physical laws are formal

- implementation of algorithms of calculation and test of parameters of the optical systems

- transfer of the optical design knowledge from the expert to the computer in terms of clauses
- manipulations and treatments of optical knowledge
- solving optical design problems in a declarative way and not by a procedural algorithm
- thanks to the inference engine and backtracking principle all solutions of a lens design problem are explored

The Prolog programming language was chosen because this language is equipped with an inference engine and the possibility of algorithmic programming. The user has only to describe the optical problem and the program will then find all possible solutions.
The Prolog programming language is oriented toward the implementation of artificial intelligence principles and the transfer of knowledge from the expert to the computer through rules and facts.
A knowledge base is implemented in Prolog in terms of clauses. In our case this knowledge base contains optical knowledge, rules, laws of lens design, heuristics and constraints of lens design as well as a base knowledge of optical glass.

To solve an optical problem in Prolog, the desired optical system is described. Then the inference engine explores the optical knowledge base and returns a solution. In using the backtracking principle, the inference engine explores all the possible solutions to a lens design problem. This method of problem solving is typical of the declarative or descriptive programming languages such as Prolog. We are specially attracted by this declarative aspect that allows the solution of optical problems in an intelligent way.
In using the programming language Prolog, equipped with an inference engine, one has only to describe the base of facts, the optical rules, the heuristics and the relationship between the elements of the knowledge base as well as the optical problems, and the program will find all possible solutions. For the above mentionned reasons Prolog is called descriptive or declarative programming language. The possibility of algorithmic or procedural programming in Prolog allows the control of the inference engine and the searching algorithms.

The application domain of the KBOSD is the dioptrical optical system of low aperture and low field angle.

This knowledge based optical system design generates optical systems that are used as a start optical system for programs of optical optimizations.
The KBOSD describes how and why a solution is selected!

The KBOSD proposes a start optical system in a few seconds. This start optical system is subsequently optimized by a lens design program and after several iterations a good optical design system is obtained.

To achieve the same goal, an optical design expert using just his experience needs much more time.

The optical start system proposed by the KBOSD and optimized have a better quality than the optical system obtained from parallel plane plates starting system and optimized using an existing commercial optimization program.

To obtain optical systems from parallel plane plates with an optical quality comparable to the optical systems obtained from the optimization of the KBOSD start systems one has to master the optimization program and to do some preliminary calculations conceming the start system such as the selection of the appropriated optical glass. A novice in the optical design field spendes much of his time mastering the optimization program, the KBOSD start system is a helpful tool for him. In same simple situation(singlet lenses) the optimization of parallel plane plates start system leads to the KBOSD optical start system before their optimization.

The KBOSD optical start systems are pre-calculated and pre-designed and have a low third order aberrations, because of these reasons they lead rapidly to a good quality optical systems after the optimizations.

Equipped with optical lens design knowledge, with cognitive and metacognitive of optical lens design and the possibility of solving lens design problems, this KBOSD has all properties of an intelligent knowledge based system.

Thus the KBOSD is a helpful tool for assisting an optical design expert in saving time and reducing the probability of error. Additionally, an optical design expert can quite easily enter his own knowledge on optical design to the KBOSD. In any case the optical knowledge contained in the KBOSD is not absolute.

The results obtained by this KBOSD are very encouraging and prove the feasibility of such a program. It is highly recommended to continue the investigations and to expand upon this knowledge based systems to other domains of optics such as optical computing, polarization and thin films, holography and optical fibers.

Future realizations of KBOSD based on the parallel programming and/or on neural computing are also highly recommended.

Acknowledgements

I would like to especially thank the following persons for their assistance:

• Professor P. J. Erard, director of this thesis, for his supervision and his knowledge in the fields of programming and artificial intelligence, for his patience as well as his pedagogy. It was a great pleasure for me to work with him. Thanks to his advice, I saved considerable time.

• Dr. H. Buczek for his advisement and knowledge in the field of optical systems design as well as for his encouragement throughout the duration of this work.

• Professor R. Azzam from the University of New Orleans for his interest displayed in this research as well as the constructive discussions I had with him concerning this thesis.

• Mr. R. Czichy from ESA Noordwijk (Holland) for his interest in this project and for the fruitful discussions I had with him.

• Professor R. Dændliker from the University of Neuchâtel for serving as a member of the jury.

References

[1] H. Haferkorn, "Optik", VEB Deutscher Verlag der Wissenschaften, Berlin(1980).

[2] H. Haferkorn, W. Richter, "Synthese optischer Systeme", VEB Deutscher Verlag der Wissenschaften, Berlin(1984).

[3] H. Haferkorn, "Bewertung optischer Systeme", VEB Deutscher Verlag der Wissenschaften, Berlin(1986).

[4] W.T. Welford, "Aberrations of optical systems", Adam Hilger Ltd, Bristol BS1 6NX, (1986).

[5] Rudolf Kindslake, "Lens design fundamentals", Academic press, INC., London (1978).

[6] Patrick Henry Winston, "Intelligence artificielle", InterEdition, Paris(1988).

[7] Ivan Bratko, "Programmation en Prolog pour l'intelligence artificielle", Paris, (1988).

[8] Guy Boy, "Assistance à l'opérateur, une approche de l'intelligence artificielle", teknea,Toulouse, (1988).

[9] Schott, "Verre d'optique", Mainz, (1990).

[10] SPIE Vol. 766, "Recent trends in optical Systems Design; Computer Lens Workshop", (1987).

[11] SPIE Vol. 554, "International Lens Design Conference", (1985).

[12] Alain Bonnet, "L'intelligence artificielle, promesse et réalités", InterEdition, (1984).

[13] Alain Bonnet, "Systèmes-experts, vers la maîtrise technique", InterEdition, (1986).

[14] PrologIA, "Prolog II+, version 1.4", luminy, Marseille (1989).

[15] H. Kanoui, "Prolog", InterEdition, (1985).

[16] FNAC, "Dossier FNAC", FNAC, Colmar, (1990).

[17] Kidger optics Ltd, "Sigma PC, version 3.7", Crowborough, (1989).

[18] H. Buczek, T. Nouri, "Lens design software has wide capability range", Laser Focus World, (November 1990).

[19] D. C. Dilworth, SPIE Vol. 766, "Recent trends in optical Systems Design; Computer Lens Workshop", (1987).

[20] M. Kidger, SPIE Vol. 766, "Recent trends in optical Systems Design; Computer Lens Workshop", (1987).

[21] P. Bihan/P.E. Parizot, "Exercice en turbo prolog", Eyrolles, (1987).

[22] Leon Sterling, "The Art of Prolog", The MIT Press, (1986).

[23] Robert Kowalski, Information processing 74, "Predicate Logic as Programming Language", North-Holland Pubkishing Compagny(1974).
[24] R. Kingslake, "Optical system design", Academic press Inc. London(1983).
[25] W. F. Clocksin & C. S. Mellish, "Programming in Prolog", Springer-Verlag, Berlin(1981).
[25] "Optimization and AI take lead at ILCD '90", Laser Focus World, (August 1990).
[26] "SYNOPSYS", Farnham. Point, P.O. Box # 247, East Boothbay, Maine 04544, USA(1990).
[27] "Genesee Optical Sotfware", 3136 Winton Road South, Rochester, New York 14623, USA(1990).

Appendix 1: Comparative Lens Data

We report below the data [16] of some lenses on the optical design market so that one get an idea on the performance and the quality of the mentioned lenses.

This lens data is reported for comparison with optical systems generated by our KBOSD.
The focal length of all lenses is 50 mm.

Lens	Leitz Summicron M50 F/2	Minolta AF 50 mm F/1.7	Canon EF 50 mm F/1.8	Praktica MC 50 mm F/1.8
Full apperture resolution	64 lines/mm	52 lines/mm	48 lines/mm	38 lines/mm
Throu-focus MTF full aperture	92%	87%	90%	65%
Distortion	-0%	-0.8%	-1%	-1.8%
Resolution F/8	75 lines/mm	71 lines/mm	54 lines/mm	46 lines/mm
Image quality	Excellent	Very good	Good	Mean

Appendix 2: Interfaces KBOSD-User and KBOSD-lens Design Programs

Using the Knowledge Based Optical System Design one has to enter the following predicate: calcul(Focal_length, Magnification, l, l', z, z'). The user has to input at least two identified parameters of the following list: Focal length, Lateral Magnification, l, l', z and z'.

Example 1:

calcul(100.0,-2.0,l,l',z,z') this predicate indicates to the KBOSD that the user needs a focal optical system with a focal length of 100.0 mm and a lateral magnification of -2.0.

If the focal length is zero then the KBOSD proposed an afocal optical system.

Example 2:

calcul(0.0,-3.0, 200.0,_,_,_) this predicate indicated to the KBOSD that the user needs an afocal optical system with an angular magnification -3.0 and a total length of 200.0 mm.

If the user types more than two identified parameters, then the KBOSD tests the compatibility of these parameters. The signification of the parameters l, l', z and z' are displayed below.

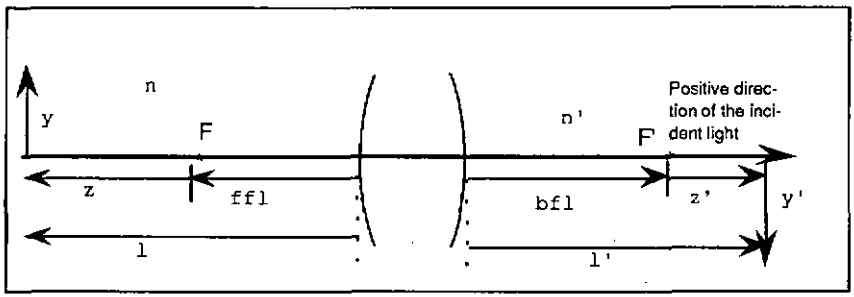


Fig. 3.1.4 Parameters of an Optical System

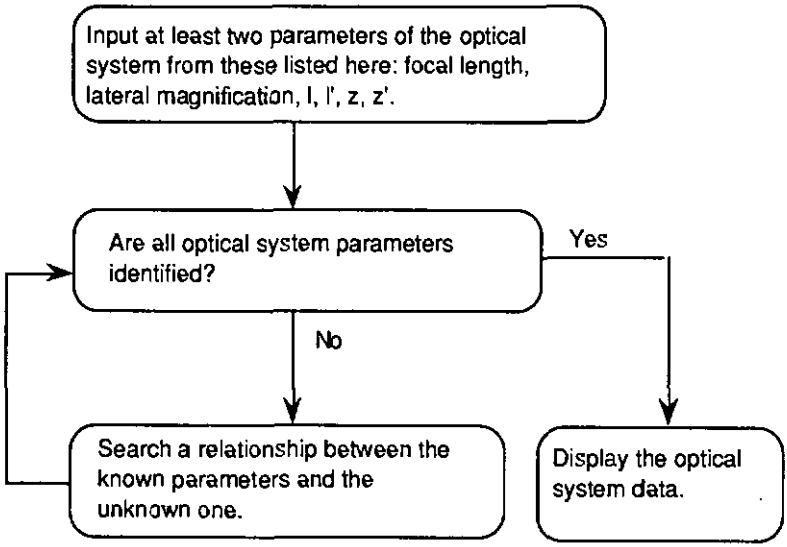
The following parameters: wavelengths, entrance aperture and optical glass constraints should be entered by the user into the KBOSD.

The KBOSD uses the user's optical parameters to identify the the optical system and its final conditions of use. The KBOSD assembles the optical system according to its parameters and its context of use.

The KBOSD answers by displaying the parameters of the optical system such as optical glass, thickness, curvature radii, number of elements forming the optical system...
The parameters of the optical system displayed by the KBOSD are in a text file and are readable in the existing optimization programs.

Appendix 3: Rules and Algorithms of Calculation of Parameters of Optical Systems

The algorithms displayed below serve to calculate the paraxial parameters of the optical system. The user should enter at least two parameters.



Algorithms of calculation of optical system parameters

The algorithms of calculation of optical system parameters are coded in the programming language Prolog as below.

```
/*

The predicate output_calcul_results displays the results of the calcul, i.e.
the parameters: Optical_Power, Transversal_Magnification, L, L', Z and
Z'.

*/

calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        know_variable([Optical_Power, Transversal_Magnification, L, L', Z, Z']), !,
        output_calcul_results(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*
```

In the following rule if the variables L and L', know_variable([L, L']), are known, then, the variable Optical Power is calculated according to the relation:

Optical_power = (Ni/L) + (No/L').

Where Ni, No are the refractive indices in the object and image space.

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([L, L']),
        relation(calculation_rule_No1, Optical_Power, (Ni/L) + (No/L'), "(Ni/L) +
        (No/L')", "Optical_Power"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule if the variables Z' and L', know_variable([L', Z']), are known, then, the variable Optical Power is calculated according to the relation:

Optical_power = No/(L'-Z').

Where Ni, No are the refractive indices in the object and image space.

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([L', Z']),
        relation(calculation_rule_No2, Optical_Power, No/(L'-Z'), "No/(L'-Z')",
        "Optical_Power"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule if the variables Z' and L', know_variable([Z, Z']), are known, then, the variable Optical Power is calculated according to the relation:

Optical_power = sqrt((Ni*No)/(Z*Z')).

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
```

```
        know_variable([Z, Z']),
        relation(calculation_rule_No3, Optical_Power, sqrt((Ni*No)/(Z*Z')), "sqrt((Ni
        *No)/(Z*Z'))", "Optical_Power"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule if the variables Z' and L', know_variable([Z, Z']), are known, then, the variable Optical Power is calculated according to the relation:

Optical_power = -sqrt((Ni*No)/(Z*Z')).

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Z, Z']),
        relation(calculation_rule_No4, Optical_Power, -sqrt((Ni*No)/(Z*Z')), "-
        sqrt((Ni*No)/(Z*Z'))", "Optical_Power"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule if the variables Z and L, know_variable([L, Z]), are known, then, the variable Optical Power is calculated according to the relation:

Optical_power = Ni/(L-Z).

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([L, Z]),
        relation(calculation_rule_No5, Optical_Power, Ni/(L-Z), "Ni/(L-Z)",
        "Optical_Power"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule if the variables Optical_Power and L', know_variable([Optical_Power, L']), are known, then, the variable L is calculated according to the relation:

L = (Ni*L')/((Optical_Power*L') - No).

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L']),
        relation(calculation_rule_No6, L, (Ni*L')/((Optical_Power*L') - No), "(Ni*L')/
        ((Optical_Power*L') - No)", "L"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule if the variables Transversal_Magnification and L', know_variable([Transversal_ Magnification, L']), are known, then, the variable L is calculated according to the relation:

$$L = (-L'*Ni)/(Transversal\_Magnification*No).$$

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Transversal_Magnification, L']),
        relation(calculation_rule_No7, L, (-L'*Ni)/(Transversal_Magnification*No), "(-
        L'*Ni)/(Transversal_Magnification*No)", "L"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule if the variables Optical_Power and Z, know_variable([Optical_Power, Z]), are known, then, the variable L is calculated according to the relation:

$$L = Z + (Ni/Optical\_Power).$$

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, Z]),
        relation(calculation_rule_No8, L, Z + (Ni/Optical_Power), "Z + (Ni/ Opti-
        cal_Power) ", "L"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule if the variables Optical_Power and L, know_variable([Optical_Power, L]), are known, then, the variable L' is calculated according to the relation

$$L' = (No*L)/((Optical\_Power*L) - Ni).$$

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L]),
        relation(calculation_rule_No9, L', (No*L)/((Optical_Power*L) - Ni), "(No*L)/
        ((Optical_Power*L) - Ni)", "L'"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule if the variables Optical_Power and Z', know_variable([Optical_Power, Z']), are known, then, the variable L' is calculated according to the relation

$$L' = Z' + (No/Optical\_Power).$$

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, Z']),
        relation(calculation_rule_No10, L', Z' + (No/Optical_Power), "Z' + (No/ Opti-
        cal_Power)", "L'"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule if the variables Transversal_Magnification and L, know_ variable([Transversal_ Magnification, L]), are known, then, the variable L' is calculated according to the relation

$$L' = Z' + (-Transversal\_Magnification*No*L)/Ni.$$

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Transversal_Magnification, L]),
        relation(calculation_rule_No11, L', (-Transversal_Magnification*No*L)/Ni, "(-
        Transversal_Magnification*No*L)/Ni)", "L'"), !,
```

```
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').

/*
```

In the following rule if the variables Optical_Power and L, know_variable ([Optical_Power, L]), are known, then, the variable Z is calculated according to the relation

$$Z = L - (Ni/Optical\_Power).$$

```
*/

calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L]),
        relation(calculation_rule_No12, Z, L - (Ni/Optical_Power), "L - (Ni/ Opti-
        cal_Power)", "Z"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').

/*
```

In the following rule if the variables Optical_Power and Z', know_variable([Optical_Power, Z']), are known, then, the variable Z is calculated according to the relation

$$Z = (Ni*No)/(Optical\_Power*Optical\_Power*Z').$$

```
*/

calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, Z']),
        relation(calculation_rule_No13, Z, (Ni*No)/(Optical_Power*Optical_Power*Z'),
        "(Ni*No)/Optical_Power*Optical_Power*Z'", "Z"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').

/*
```

In the following rule if the variables Optical_Power and Z, know_variable([Optical_Power, Z]), are known, then, the variable Z' is calculated according to the relation

$$Z' = (Ni*No)/(Optical\_Power*Optical\_Power*Z).$$

```
*/
```

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, Z]),
        relation(calculation_rule_No14, Z', (Ni*No)/(Optical_Power*Optical_Power*Z),
        "(Ni*No)/Optical_Power*Optical_Power*Z", "Z'"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule if the variables Optical_Power and L', know_variable([Optical_ Power, L']), are known, then, the variable Z' is calculated according to the relation

$$Z' = L' - (No/Optical\_Power).$$

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L']),
        relation(calculation_rule_No15, Z', L' - ( No/Optical_Power), "L' -( No/ Optical_Power)", "Z'"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule if the variables L and L', know_variable([L, L']), are known, then, the variable Transversal_Magnification is calculated according to the relation

$$Transversal\_Magnification = - (Ni*L')/(No*L).$$

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([L, L']),
        relation(calculation_rule_No16, Transversal_Magnification, - (Ni*L')/(No*L), "-(Ni*L')/ (No*L)", "Transversal_Magnification"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule if the variables Transversal_Magnification and L', know_variable([Transversal_ Magnification, L']), are known, then, the variable Optical_Power is calculated according to the relation:

Optical_Power = (- No*Transversal_Magnification+ No)/L'.

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Transversal_Magnification, L']),
        relation(calculation_rule_No17, Optical_Power, (- No*Transversal_Magnification
        + No) / L', "(- No* Transversal_Magnification+No)/ L'", "Optical_Power"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule if the variables Transversal_Magnification and Optical_Power, know_variable([Optical_Power, Transversal_ Magnification]), are known, then, the variable L' is calculated according to the relation

L' = (No*Transversal_Magnification+No)/ Optical_Power.

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, Transversal_Magnification]),
        relation(calculation_rule_No18,L',(No*Transversal_Magnification+No)/ Opti-
        cal_Power, "(- No*Transversal_Magnification+No)/Optical_Power", "L'"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule if the variables Optical_Power and L, know_variable([Optical_Power, L]), are known, then, the variable Transversal_Magnification is calculated according to the relation:

Transversal_Magnification = (Ni/((-Optical_Power* L) + Ni))

*/

```
calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L]),
        relation(calculation_rule_No19, Transversal_Magnification, (Ni/((-Optical_Power*
        L ) + Ni)), "(Ni/((-Optical_Power*L)+Ni))", "Transversal_Magnification"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').
```

```
/*
```

In the following rule if the variables Optical_Power and L, know_variable([Optical_ Power, L]), are known, then, the variable Z' is calculated according to the relation

Z' = (No*L/((Optical_Power*L)-Ni))-No/ Optical_Power.

```
*/

calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L]),
        relation(calculation_rule_No20, Z', (No*L/((Optical_Power*L)-Ni))-No/ Opti-
        cal_Power, "(No*L/((Optical_Power*L)-Ni))-No/Optical_Power", "Z'"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').

/*
```

In the following rule if the variables Optical_Power and L', know_variable([Optical_ Power, L']), are known, then, the variable Transversal_Magnification is calculated according to the relation

Transversal_Magnification = (No-Optical_Power* L')/No.

```
*/

calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L']),
        relation(calculation_rule_No21, Transversal_Magnification, (No-Optical_Power
        *L')/No, "(No-Optical_Power*L')/No", "Transversal_Magnification"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').

/*
```

In the following rule if the variables Optical_Power and Z', know_variable([Optical_ Power, Z']), are known, then, the variable Transversal_ Magnification is calculated according to the relation

L=((Ni*No)/(Optical_Power*Optical_Power*Z'))+(Ni/Optical_Power).

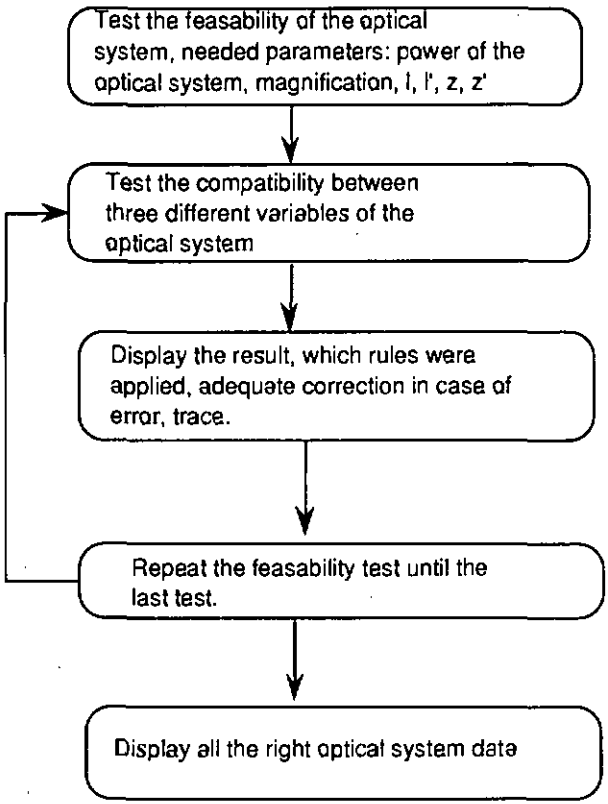
```
*/

calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
```

```
        know_variable([Optical_Power, Z']),
        relation(calculation_rule_No22,L,((Ni*No)/(Optical_Power*Optical_Power*Z'))+
        (Ni/Optical_Power),"((Ni*No)/(Optical_Power*Optical_Power*Z'))+ (Ni/ Opti-
        cal_Power)","L"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').

/*

In the following rule  if the variables Optical_Power and L,
know_variable([Optical_ Power, L]), are known, then, the variable Z' is
calculated according to the relation

                Z' =(Ni*No)/(Optical_Power*(Optical_Power*L-Ni)).

*/


calcul(Optical_power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L]),
        relation(calculation_rule_No23, Z', (Ni*No)/(Optical_Power*(Optical_Power*L-
        Ni)), "(Ni*No)/(Optical_Power*(Optical_Power*L-Ni))", "Z'"), !,
        calcul(Optical_power, Transversal_Magnification, L, L', Z, Z').

/*

This rule is used by the calcul predicates.

This rule  links the value, Var_Free, and the formula, Formule, during the
algorithm calculation of an optical system.
If a variable is free(Var_Free), then, the formula is assigned to
it(Var_Free is Formule).

This rule displays the relation between an optical variable and the optical
rule (Text_Formule) that indicts the way to calculate it.

*/

relation(Rule_No, Var_Free, Formule, Text_Formule, Var_Text) :-
        free(Var_Free),
        Var_Free is Formule,
        nl,
        outl( Rule_No),
        out(Var_Text),
        outm(" = "),
        out(Text_Formule),
        outm(" --> "),
        nl,
        out(Var_Free),
        outm(" = "),
        out(Formule),
        nl.
```

```
/*

This rule testes if a variable is known, not free.
It is applied to a list of variables.

*/

know_variable([]).
know_variable([P|A])        :-
        bound(P),
        know_variable(A).

/*

This rule is used to display the data of a focal optical system after the calculation algorithm.

*/

output_calcul_results(Optical_Power, Transverse_Magnification, L, L', Z, Z') :-
        power_focal(Focal_Length, Optical_Power),
        nl,
        outm("Focal_Length  --> "),
        outl( Focal_Length),
        outm("Optical_Power          --> "),
        outl( Optical_Power),
        outm("Transversal_Magnification      --> "),
        outl( Transversal_Magnification),
        outm("Obj-Lens          --> "),
        outl( L),
        outm("Lens-Image     --> "),
        outl( L'),
        outm("Obj-Focal_Length      --> "),
        outl( Z),
        outm("Focal_Length-Image --> "),
        outl( Z'),
        nl.
```

Appendix 4: Rules and Algorithms of Test of Parameters of Optical Systems

The algorithm below tests the compatibility of the paraxial parameters of the optical systems when the user enters more than two parameters.



Algorithms of the Test of Parameters of the Optical Systems

The algorithms of the test of parameters of the optical systems are coded in the programming language Prolog as displayed below.

```
test(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        nl,
        know_variable([L, L',Optical_Power]),
        relationtest(test1,Optical_Power,(Ni/L)+(No/L'),"(Ni/L)+(No/L')","Optical_Power")
        ,
```

```
        test1(Optical_Power, Transversal_Magnification, L, L', Z, Z').

test(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([L, L',Optical_Power]),
        relationtest(test1, Optical_Power, (Ni/L) + (No/L'), "(Ni/L) + (No/L')",
        "Optical_Power"),
        test1(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*
```

If three variables, Optical_Power, L' and Z' are known, know_ variable([L', Z', Optical_Power]), then, their compatablility is tested according to the relation:

$$Optical\_Power = No/(L'-Z').$$

```
*/

test1(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([L', Z',Optical_Power]),
        relationtest(test2, Optical_Power, No/(L'-Z'), "No/(L'-Z')", "Optical_Power"), !,
        test2(Optical_Power, Transversal_Magnification, L, L', Z, Z').
/*
```

If three variables, Transversal_Magnification, L'. and L are known, know_variable([Transversal_Magnification, L', L]), then, their compatablility is tested according to the relation:

$$L = (-L'*Ni)/(Transversal\_Magnification*No).$$

```
*/

test2(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Z, Z',Optical_Power]),
        relationtest(test3, Optical_Power, sqrt(Ni*No/(Z*Z')), "sqrt(Ni*No/(Z*Z'))",
        "Optical_Power"), !,
        test3(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*
```

If three variables, Z, Z' and Optical_Power are known, know_variable([Z, Z',Optical_ Power]), then, their compatablility is tested according to the relation:

$$Optical\_Power = sqrt(Ni*No/(Z*Z')).$$

```
*/
```

```
test3(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Z, Z',Optical_Power]),
        relationtest(test4, Optical_Power, sqrt(Ni*No/(Z*Z')), "sqrt(Ni*No/(Z*Z'))",
        "Optical_Power"), !,
        test4(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*
```

If three variables, L, Z, and Optical_Power are known, know_variable([L, Z, Optical_ Power]), then, their compatablility is tested according to the relation:

Optical_Power = Ni/(L-Z).

```
*/

test4(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([L, Z,Optical_Power]),
        relationtest(test5, Optical_Power, Ni/(L-Z), "Ni/(L-Z)", "Optical_Power"), !,
        test5(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*
```

If three variables, Transversal_Magnification, L' and L are known, know_variable([Transversal_Magnification, L', L]), then, their compatablility is tested according to the relation:

L = (-L'*Ni)/(Transversal_Magnification*No).

```
*/

test5(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L', L]),
        relationtest(test6, L, (Ni*L')/((Optical_Power*L')-No),"(Ni*L')/ ((Optical_ Po-
        wer*L') - No)", "L"), !,
        test6(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*
```

If three variables, Transversal_Magnification, L' and L are known, know_variable([Transversal_Magnification, L', L]), then, their compatablility is tested according to the relation:

L = (-L'*Ni)/(Transversal_Magnification*No).

```
*/

test6(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
```

```
        indice(Ni, No),
        know_variable([Transversal_Magnification, L', L]),
        relationtest(test7, L, (-L'*Ni)/(Transversal_Magnification*No), "(-L'*Ni)/
        (Transversal_Magnification*No)", "L"), !,
        test7(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*
```

If three variables, Optical_Power, Z and L are known, know_variable([Optical_Power, Z, L]), then, their compatablility is tested according to the relation:

$$L = Z + (Ni/Optical\_\ Power).$$

```
*/

test7(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, Z, L]),
        relationtest(test8, L, Z + (Ni/Optical_Power), "Z + (Ni/Optical_Power)", "L"), !,
        test8(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*
```

If three variables, Optical_Power, L', and L are known, know_variable([Optical_ Power, L, L']), then, their compatablility is tested according to the relation:

$$L' = Z(No*L)/((Optical\_Power*L) - Ni)$$

```
*/


test8(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L, L']),
        relationtest(test9, L', (No*L)/((Optical_Power*L) - Ni), "(No*L)/((Optical_ Po-
        wer*L) - Ni)", "L'"), !,
        test9(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*
```

If three variables, Optical_Power, L', and Z' are known, know_variable([Optical_Power, Z', L']), then, their compatablility is tested according to the relation:

$$L' = (Z' + (No/Optical\_Power))$$

```
*/

test9(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
```

```
        indice(Ni, No),
        know_variable([Optical_Power, Z', L']),
        relationtest(test10, L', (Z' + (No/Optical_Power)), "Z' + (No/Optical_Power)",
        "L'"), !,
        test10(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*

If three variables, Transversal_Magnification, L and L' are known, know_
variable([Transversal_Magnification, L, L']), then, their compatablility is
tested according to the relation:

                    L' = (-Transversal_Magnification*No*L)/Ni.

*/

test10(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Transversal_Magnification, L, L']),
        relationtest(test11, L', (-Transversal_Magnification*No*L)/Ni, "(-Transver-
        sal_Magnification*No*L)/Ni)", "L'"), !,
        test11(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*

If three variables, Optical_Power, L and Z are
know_variable([Optical_Power, L, Z]), then, their compatablility is tested
according to the relation:

                    Z = L - (Ni/Optical_Power)

*/

test11(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L, Z]),
        relationtest(test12, Z, L - (Ni/Optical_Power), "L - (Ni/Optical_Power)", "Z"), !,
        test12(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*

If three variables, Optical_Power, Z' and Z are
know_variable([Optical_Power, Z', Z]), then, their compatablility is tested
according to the relation:

                    Z = (Ni*No)/(Optical_Power*Optical_Power*Z').

*/

test12(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
```

```
        know_variable([Optical_Power, Z', Z]),
        relationtest(test13, Z, (Ni*No)/(Optical_Power*Optical_Power*Z'), "(Ni*No)/
        (Optical_Power*Optical_Power*Z')", "Z"), !,
        test13(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*
```

If three variables, Optical_Power, Z' and Z are know_variable([Optical_Power, Z, Z']), then, their compatablility is tested according to the relation:

$$Z' = (Ni*No)/ (Optical\_Power*Optical\_Power*Z).$$

```
*/

test13(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, Z, Z']),
        relationtest(test14, Z', (Ni*No)/(Optical_Power*Optical_Power*Z),
        "(Ni*No)/(Optical_Power*Optical_Power*Z)", "Z'"), !,
        test14(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*
```

If three variables, Optical_Power, Z' and L are know_variable([Optical_Power, L', Z']), then, their compatablility is tested according to the relation:

$$Z' = L' -(No/Optical\_Power).$$

```
*/

test14(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L', Z']),
        relationtest(test15, Z', L' -( No/Optical_Power), "L' -(No/Optical_Power)", "Z'"),
        !,
        test15(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*
```

In the following rule, if three variables, Transversal_Magnification, L and L', are know know_variable([Transversal_Magnification, L, L']), then, their compatablility is tested according to the relation:

$$Transversal\_Magnification = - (Ni*L')/(No*L).$$

```
*/

test15(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
```

```
        know_variable([Transversal_Magnification, L, L']),
        relationtest(test16, Transversal_Magnification, - (Ni*L')/(No*L), "- (Ni*L')/
        (No*L)", "Transversal_Magnification"), !,
        test16(Optical_Power, Transversal_Magnification, L, L', Z, Z').
/*
```

In the following rule, if three variables, Transversal_Magnification, L' and Optical_Power, are know know_variable([Transversal_Magnification, Optical_Power, L']), then, their compatablility is tested according to the relation:

Optical_Power] = (-No*Transversal_ Magnification+No)/L'.

```
*/

test16(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Transversal_Magnification, L', Optical_Power]),
        relationtest(test17, Optical_Power, (-No*Transversal_Magnification+No)/L', "(-
        No* Transversal_Magnification+No)/L'", "Optical_Power"), !,
        test17(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*
```

In the following rule, if three variables, Transversal_Magnification, L' and Optical_Power, are know know_variable([Transversal_Magnification, Optical_Power, L']), then, their compatablility is tested according to the relation:

L' = (-No*Transversal_Magnification+No)/Optical_Power.

```
*/

test17(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, Transversal_Magnification, L']),
        relationtest(test18, L', (-No*Transversal_Magnification+No)/Optical_Power, "(-
        No*Transversal_Magnification+No)/Optical_Power", "L'"), !,
        test18(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*
```

In the following rule, if three variables, Transversal_Magnification, L and Optical_Power, are know_variable([Optical_Power, L, Transversal_ Magnification]), then, their compatablility is tested according to the relation:

Transversal_Magnification = (Ni/((-Optical_Power* L)+Ni)).

```
*/

test18(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L, Transversal_Magnification]),
```

```
        relationtest(test19, Transversal_Magnification, (Ni/((-Optical_Power* L)+Ni)),
        "(Ni/((-Optical_Power*L)+Ni))", "Transversal_Magnification"), !,
        test19(Optical_Power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule, if three variables, L, Z' and Optical_Power, are know_variable([Optical_Power, L, Z']), then, their compatablility is tested according to the relation:

Z' = ((No*L)/((Optical_Power*L)-Ni))-(No/Optical_Power).

*/

```
test19(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L, Z']),
        relationtest(test20, Z', ((No*L)/((Optical_Power*L)-Ni))-(No/Optical_Power),
        "((No*L)/ ((Optical_Power*L)-Ni))-No/Optical_Power", "Z'"), !,
        test20(Optical_Power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule, if three variables, Transversal_Magnification, L' and Optical_Power, are know_variable([Optical_Power, L', Transversal_ Magnification]), then, their compatablility is tested according to the relation:

Transversal_Magnification = (No-Optical_Power*L')/No.

*/

```
test20(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L', Transversal_Magnification]),
        relationtest(test21, Transversal_Magnification, (No-Optical_Power*L')/No, "(No-
        Optical_Power*L')/No", "Transversal_Magnification"), !,
        test21(Optical_Power, Transversal_Magnification, L, L', Z, Z').
```

/*

In the following rule, if three variables, Z', L and Optical_Power, are know_variable([Optical_Power, Z', L]), then, their compatablility is tested according to the relation:

L = ((Ni*No)/(Optical_Power*Optical_Power*Z'))+(Ni/Optical_Power).

*/

```
test21(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
```

```
        indice(Ni, No),
        know_variable([Optical_Power, Z', L]),
        relationtest(test22, L, ((Ni*No)/(Optical_Power*Optical_Power*Z'))+ (Ni/ Opti-
        cal_Power),"(Ni*No/(Optical_Power*Optical_Power*Z'))+(Ni/Optical_Power)",
        "L"), !,
        test22(Optical_Power, Transversal_Magnification, L, L', Z, Z').

/*
```

In the following rule, if three variables, Z', L and Optical_Power, are know_variable([Optical_Power, Z', L]), then, their compatablility is tested according to the relation:

$$Z' = (Ni*No)/(Optical\_Power*(Optical\_Power*L-Ni)).$$

```
*/

test22(Optical_Power, Transversal_Magnification, L, L', Z, Z') :-
        indice(Ni, No),
        know_variable([Optical_Power, L, Z']),
        relationtest(test23, Z', (Ni*No)/(Optical_Power*(Optical_Power*L-Ni)),
        "(Ni*No)/(Optical_Power*(Optical_Power*L-Ni))", "Z'"), !.


/*
```

This rule is used to link the variable, Var_Free, and the equation, Formule, if the algorithm test of an optical system succeeds.

```
*/


        relationtest(Test_No, Var_Free, Formule, Text_Formule, Var_Text) :-
        delta(Var_Free, Formule, Delta),
        abs(Delta) < 0.01,
        nl,
        write(Test_No),
        nl,
        out(Var_Text),
        outm("                    is equal to     "),
        out(Text_Formule),
        outm(" --> "),
        nl,
        write(Var_Free),
        outm(" = "),
        write(Formule),
        nl,
        outm(" ", 30),
        outm("          Test successful ").

/*
```

This rule is used to link the variable, Var_Free, and the formula, Formule, if the algorithm test of an optical system fails.
This rule also proposes the appropriate value of the variable Var_Free.

*/

```
relationtest(Test_No, Var_Free, Formule, Text_Formule, Var_Text) :-
        nl, outm("Error according to "),
        write(Test_No),
        out(Var_Text),
        outm(" <> "),
        out(Text_Formule),
        write(Var_Free),
        outm(" <> "),
        write(Formule),
        outm(" ", 20),
        outm("-->"),
        write(Var_Text),
        write(" can be only "),
        Formula is Formule,
        write(Formula),
        outm("> Test fails <").
```

Appendix 5: Heuristics Concerning Assembling of Optical Lenses

The following clauses are used for assembling the doublet and triplet lenses, as well as high aperture lenses.

```
/*
```

The following rule, assemblingdoublet(Optical_Power, Optical_Power1, Optical_Power2,Refractive_Index1,Refractive_Index2,Glass_Nam1, Glass_Nam2, Diameter, Separation), is used for the assembling process of a doublet lens at low temperature.
The doublet lens is cemented together if it will be used at low temperature i.e the separation between the two singlets is zero.

Two optical glasses are selected according to the clause lens refractive index 2; these two optical glasses satisfy the chromatic correction conditions according to the physical laws:

Optical_Power1 is V1*Optical_Power/(V1-V2),
Optical_Power2 is V2*Optical_Power/(V2-V1).

```
*/

assemblingdoublet(Optical_Power, Optical_Power1, Optical_Power2, Refractive_Index1,
        Refractive_Index2, Glass_Nam1, Glass_Nam2, Diameter, Separation) :-
        low_temperature(T),
        outm("assembling process doublet "),
        nl,
        lens_refractive_index2([Glass_Nam1, Glass_Nam2, Refractive_Index1, Refrac-
        tive_Index2, V1, V2]),
        Separation is 0.0,
        Optical_Power1 is V1*Optical_Power/(V1-V2),
        Optical_Power2 is V2*Optical_Power/(V2-V1).

/*
```

The following rule, assemblingdoublet(Optical_Power, Optical_Power1, Optical_Power2, Refractive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2, Diameter, Separation), is used for an assembling process doublet lens at high temperature.

If a doublet lens will be used at high temperature, then, the separation between the two singlet lenses must be different from zero. It is not possible to cement the two singlet lenses together because the cement melts at high temperature.

The cement usually used melts at 150 degrees celcius. This is a heuristic.

In this case, one has to separate both singlet lenses with a distance of 0.4 mm. This distance of 0.4 mm is considered as optimal for the optical aberrations and for the optical lenses manufacturer.
The manufacturer uses a ring to separate the singlet lenses. It is easy to produce a ring of such as thickness. If the thickness is smaller than 0.4 mm it is difficult to manufacture the ring.
If the thickness is greater than 0.4 mm, it has a bad effect on the optical quality of the doublet lens. This value is a heuristic too.

Two optical glasses are selected according to the clause lens refractive index 2; these two optical glasses satisfy the chromatic correction conditions according to the physical laws:

```
        Optical_Power1 is V1*Optical_Power/(V1-V2),
        Optical_Power2 is V2*Optical_Power/(V2-V1).

*/

assemblingdoublet(Optical_Power, Optical_Power1, Optical_Power2, Refractive_Index1,
        Refractive_Index2, Glass_Nam1, Glass_Nam2, Diameter, Separation) :-
        high_temperature(T),
        outm("assembling process doublet "),
        nl,
        Separation is 0.4,
        lens_refractive_index2([Glass_Nam1, Glass_Nam2, Refractive_Index1, Refrac-
        tive_Index2, V1, V2]),
        Optical_Power1 is V1*Optical_Power/(V1-V2),
        Optical_Power2 is V2*Optical_Power/(V2-V1).
/*
```

The following rule is used for an assembling thickness doublet lens.

```
*/

assemblingthickness(Thickness, Thickness1, Thickness2) :- Thickness is Thickness1 + Thi-
        ckness2.

/*
```

The following rule, assemblingtriplet(Optical_Power0, Optical_Power1, Optical_Power2, Optical_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2), is used to assemble triplet.

The triplets are assembled at low temperature or at high temperature.
At low temperature a triplet lens can be cemented.
At high temperature a triplet lens must be separated by an air space.

One selects three optical glasses according to the rule lens refractive_index3, then, the chromatic abberations are corrected according to the following physical laws:

N is V1*(T3 - T2)+V2*(T1 - T3) + V3*(T2 - T1),
Optical_Power1 is (V1*(T3 - T2)*Optical_Power)/N,
Optical_Power2 is (V2*(T1 - T3)*Optical_Power)/N,
Optical_Power3 is (V3*(T2 - T1)*Optical_Power)/N.

At least two of the three optical glasses must be different.
The separation of the singlet lenses making up the triplet is a heuristic too, the separation1 and separation2 are zero.

*/

```
assemblingtriplet(Optical_Power0, Optical_Power1, Optical_Power2, Optical_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2) :-
        low_temperature(T), corrigefocal(Optical_Power0, Optical_Power),
        lens_refractive_index3([Glass_Nam1, Glass_Nam2, Glass_Nam3, Refractive_Index1, Refractive_Index2, Refractive_Index3, T1, T2, T3, V1, V2, V3]),
        N is V1*(T3 - T2)+V2*(T1 - T3) + V3*(T2 - T1),
        Optical_Power1 is (V1*(T3 - T2)*Optical_Power)/N,
        Optical_Power2 is (V2*(T1 - T3)*Optical_Power)/N,
        Optical_Power3 is (V3*(T2 - T1)*Optical_Power)/N,
        outm("assembling process triplet     "),
        nl,
        Separation1 is 0.0, Separation2 is 0.0.
/*
```

The following rule is used for assembling a triplet at high temperature for correcting the chromatic aberrations.
The separation1 is 0.4 and the separation2 is 3 mm. The second separation will be optimized during the optimization process.

*/

```
assemblingtriplet(Optical_Power0, Optical_Power1, Optical_Power2, Optical_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2) :-
        high_temperature(T),
        corrigefocal(Optical_Power0, Optical_Power),
        lens_refractive_index3([Glass_Nam1, Glass_Nam2, Glass_Nam3, Refractive_Index1, Refractive_Index2, Refractive_Index3, T1, T2, T3, V1, V2, V3]),
        N is V1*(T3 - T2) + V2*(T1 - T3) + V3*(T2 - T1),
        Optical_Power1 is (V1*(T3 - T2)*Optical_Power)/N,
        Optical_Power2 is (V2*(T1 - T3)*Optical_Power)/N,
        Optical_Power3 is (V3*(T2 - T1)*Optical_Power)/N,
        outm("assembling process triplet     "),
        Separation1 is 0.4, Separation2 is 3.0.
```

Appendix 6: Geometrical Heuristics Concerning the Design of Optical Lenses

```
/*

The following heuristic assigns a center thickness to a singlet lens. If the
Focal length is greater than 50 mm, then, the center thickness is 2.0 mm.

*/

edgethickness(Edgethickness,Power) :-
        power_focal(Focal,Power),
        abs(Focal) >= 50.0,
        Edgethickness is 2.0.

/*

The following heuristic assigns a center thickness to a singlet lens. If tthe
Focal length is less than 50 mm, then, the center thickness is 1.5 mm.

*/

edgethickness(Edgethickness,Power) :-
        power_focal(Focal,Power),
        abs(Focal) < 50.0,
        Edgethickness is 1.50.

/*

According to this rule, the curvature radius of a lens must be greater than its
diameter/2.

*/

radius_diametre(Radius, Diameter) :-
        abs(Radius) > Diameter, !,
        nl, outm(" The curvature radius must be >= Diameter/2.0 ").

/*

The following heuristic gives the separation between the entrance and exit
block of an afocal optical system. This separation is given as the sum of focal
length of the entrance and exit optical system.

*/

separation(Focal_Length1, Focal_Length2, _, Separation, _) :-
        Separation is Focal_Length1 + Focal_Length2.

/*
```

The next two rules calculate the sagitta of a singlet lens from its radius, its diameter and its optical power.

*/

```
sagitta(Optical_Power, Radius, Diameter, Sagitta) :-
        radius_to_curvature(Radius, Curvature_Radius), Radius > 0.0,
        radius_diametre(Radius, Diameter),
        Sagitta is Curvature_Radius*(Diameter/2.0)*(Diameter/2.0)/(1.0 + sqrt(1.0 -
        ((Diameter/2.0)*(Diameter/2.0)*(Curvature_Radius*Curvature_Radius)))).


sagitta(Optical_Power, Radius, Diameter, Sagitta) :-
        radius_to_curvature(Radius, Curvature_Radius), Radius < 0.0,
        radius_diametre(Radius, Diameter),
        Sagitta is - Curvature_Radius*(Diameter/2.0)*(Diameter/2.0)/(1.0+sqrt(1.0-
        ((Diameter/2.0)*(Diameter/2.0)*(Curvature_Radius*Curvature_Radius)))).
```

/*

The heurtistics, annulusdiam(Annulus), give an annulus to a singlet lens according to the dimension of the aperture of the lens.
The annulus is the part of the lens from where the lens will be fixed.
If the aperture of a singlet lens is less than 20 mm, then, following heuristics assigns a distance of 2 mm to its annulus.

*/

```
annulusdiam(Annulus) :-
        diampupille(Apenure),
        Aperture<=20,
        Annulus is 2.0.
```

/*

According to the following heuristics, if the aperture of a singlet lens is between 20 mm and 40 mm, then, its annulus is 4 mm.

*/

```
annulusdiam(Annulus) :-
        diampupille(Aperture),
        Aperture >20,
        Aperture<=40,
        Annulus is 4.0.
```

/*

According to the following heuristics, if the aperture of a singlet lens is between 40 mm and 100 mm, then, its annulus is 5 mm.

*/

```
annulusdiam(Annulus) :-
        diampupille(Aperture),
        Aperture > 40,
        Aperture <= 100,
        Annulus is 5.0.

/*
```

According to the following heuristics, if the aperture of a singlet lens is greater than 100 mm, then, its annulus is 6 mm.

```
*/

annulusdiam(Annulus) :-
        diampupille(Aperture),
        Aperture > 100,
        Annulus is 6.0.

/*
```

The following heuristic defines the high aperture condition in case of a singlet lens. If the ratio abs(Focal_Length/Aperture) =< 4.0, then, the singlet is considered as high aperture.

```
*/

high_aperture(Optical_Power, Diameter) :-
        wavelength([Lambda1]),
        pupil_diameter(Aperture),
        power_focal(Focal_Length, Optical_Power),
        abs(Focal_Length/Aperture) =< 4.0,
        diameter(Diameter).

/*
```

The following heuristic defines the high aperture condition in case of a doublet lens. If abs(Focal_Length/Aperture) =< 3.0, then, the doublet is considered as a high aperture optical system.

```
*/

high_aperture(Optical_Power, Diameter) :-
        wavelength([Lambda1, Lambda2]),
        pupil_diameter(Aperture),
        power_focal(Focal_Length, Optical_Power),
        abs(Focal_Length/Aperture) =< 3.0,
        diameter(Diameter).

/*
```

The following heuristic defines the high aperture condition in case of a triplet lens. If abs(Focal_Length/Aperture) =< 2.5, then, the triplet is considered as a high aperture optical system.

```
*/

high_aperture(Optical_Power, Diameter) :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        pupil_diameter(Aperture),
        power_focal(Focal_Length, Optical_Power),
        abs(Focal_Length/Aperture) =< 2.5,
        diameter(Diameter).
/*
```

The following heuristic defines the low aperture condition in case of a singlet lens. If abs(Focal_Length/Aperture) >4.0, then, the singlet is considered as a low aperture optical system.

```
*/

low_aperture(Optical_Power, Diameter) :-
        wavelength([Lambda1]),
        pupil_diameter(Aperture),
        power_focal(Focal_Length, Optical_Power),
        abs(Focal_Length/Aperture) > 4.0,
        diameter(Diameter),
        outm(" low aperture   ").

/*
```

The following heuristic defines the low aperture condition in case of a doublet lens. If the ratio abs(Focal_Length/Aperture) > 3.0, then, the doublet is considered as a low aperture optical system.

```
*/

low_aperture(Optical_Power, Diameter) :-
        wavelength([Lambda1, Lambda2]),
        pupil_diameter(Aperture),
        power_focal(Focal_Length, Optical_Power),
        abs(Focal_Length/Aperture) > 3.0,
        diameter(Diameter),
        outm(" low aperture   ").

/*
```

The following heuristic defines the low aperture condition in case of a triplet lens. If the ratio abs(Focal_Length/Aperture) > 2.5, then, the triplet is considered as a low aperture optical system.

```
*/
```

```
low_aperture(Optical_Power, Diameter) :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        pupil_diameter(Aperture),
        power_focal(Focal_Length, Optical_Power),
        abs(Focal_Length/Aperture) > 2.5,
        diameter(Diameter),
        outm(" low aperture   ").

/*
```

This rule calculates the diameter of a singlet lens. The following heuristic gives the diameter as the sum of the aperture and the annulus.

```
*/

diameter(Diameter) :-
        annulusdiam(Annulus),
        pupil_diameter(Aperture),
        Diameter is Aperture + Annulus.
/*
```

This rules states the conditions of virtual image: Optical_Power > 0.0 and L < Focal_Length.

```
*/

virtuel_image(Optical_Power, L) :-
        power_focal(Focal_Length, Optical_Power),
        Optical_Power > 0.0, L < Focal_Length,
        outm(" virtuel image ").

/*
```

If the Optical_Power of a lens is negative, then, image is virtual.

```
*/

virtuel_image(Optical_Power, L) :-
        Optical_Power < 0.0,
        outm(" virtuel_image ").

/*
```

The following heuristic defines a low power lens. In fact, if a focal length of an optical system is greater than 100 mm, then, the optical system is considered as low power optical system.

```
*/

low_power_lens(Optical_Power) :-
        power_focal(Focal_Length, Optical_Power),
        abs(Focal_Length) > 100.0,
        outm("low power lens power and ").
```

```
/*
```

By this heuristic, one defines a high power optical system; if the focal length is <= 100 mm, then, the optical system is considered as high power optical system.

```
*/

high_power_lens(Optical_Power) :-
        power_focal(Focal_Length, Optical_Power),
        abs(Focal_Length) =< 100.0,
        outm("high optical power lens and").

/*
```

The following heuristic defines if an object is at a near distance. If the condition: abs(L*5.0) =< Optical_Power - 1.0/L, then, the object is considered to be at a near distance.

```
*/

not_object_at_infinity(Optical_Power, L) :-
        power_focal(Focal_Length, Optical_Power),
        abs(L*5.0) =< Optical_Power - 1.0/L,
        outm("not object at infinity").

/*
```

The following heuristic states if an object is at infinity. If the condition: abs(L*5.0) >= Optical_Power -1.0/L, is satisfied, then, the object is considered to be at infinity.

```
*/

object_at_infinity(Optical_Power, L) :-
        power_focal(Focal_Length, Optical_Power),
        abs(L*5.0) >= Optical_Power -1.0/L,
        outm("object at infinity and").

/*
```

The following three rules convert the focal length of an optical system to its power and vice versa.

```
*/

power_focal(Focal_Length, Optical_Power) :-
        bound(Focal_Length),
        Focal_Length =\= 0.0,
        Optical_Power is 1.0/Focal_Length.
```

```
power_focal(Focal_Length, Optical_Power) :-
        bound(Optical_Power),
        Optical_Power =\= 0.0,
        Focal_Length is 1.0/Optical_Power.

power_focal(Focal_Length, Optical_Power) :-
        free(Focal_Length),
        free(Optical_Power).

/*
```

The next three rules convert the radius of an optical system to its curvature radius and vice versa.

```
*/

radius_to_curvature(Radius, Curvature_Radius) :-
        bound(Radius),
        Radius =\= 0.0,
        Curvature_Radius is 1.0/Radius.

radius_to_curvature(Radius, Curvature_Radius) :-
        bound(Curvature_Radius),
        Curvature_Radius =\= 0.0,
        Radius is 1.0/Curvature_Radius.

radius_to_curvature(Radius, Curvature_Radius) :-
        free(Radius),
        free(Curvature_Radius).

/*
```

The next two heuristics describe the near infinite ratio condition.
If the magnification of an optical system is greater than five or less than 0.2, it is considered as a near infinite optical system.

```
*/

nearinfiniteratio(Transverse_Magnification) :-
        abs(Transverse_Magnification) > 5.0,
        outm("near infinite ratio      ").

nearinfiniteratio(Transverse_Magnification) :-
        abs(Transverse_Magnification) =< 0.2,
        outm("near infinite ratio      ").

/*
```

The following heuristic describes the finite ratio condition.

```
*/

finiteratio(Transverse_Magnification) :-
        abs(Transverse_Magnification) =< 5.0,
```

```
        abs(Transverse_Magnification) >= 0.95,
        outm("finite ratio and "),
        nl.

/*

The following heuristic describes the unit ratio condition.

*/

unitratio(Transverse_Magnification) :-
        abs(Transverse_Magnification) =:= 1.0,
        outm("unit ratio "),
        nl.

/*

The following heuristic describes the not unit ratio.

*/

notunitratio(Transverse_Magnification) :-
        abs(Transverse_Magnification) =\= 1.0,
        outm("not unit ratio and "),
        nl.

/*

The following heuristic states the aplanatic conditions.

*/

aplanatic(Transverse_Magnification) :-
         abs(Transverse_Magnification) < 0.95,
         abs(Transverse_Magnification) > 0.2,
        outm("aplanatisme condition and "),
        nl.

/*

The following heuristic indicates the stop surface of a triplet lens.
A triplet is composed of six optical surfaces, the surface number four is the
stop surface or an addional surface located between surface number four and
surface number five.

*/

stopsurface(Position, Aperture) :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        pupil_diameter(Aperture),
        Position is 4,
        outm("Stop surface is surface number 4, 4 surface from the left,  its diameter is "),
        Diameter is Aperture*0.9
```

```
        outl(Diameter).

/*

The following heuristic indicates the stop surface of a doublet lens. A doublet
is composed of four optical surfaces, the surface number one is the stop sur-
face.

*/

stopsurface(Position, Aperture) :-
        wavelength([Lambda1, Lambda2]),
        pupil_diameter(Aperture),
        Position is 1,
        outm("Stop surface is surface number 1, 1 surface from the left, its diameter is "),
        outl(Aperture).

/*

The following heuristic indicates the stop surface of a singlet lens.
A singlet is composed of two optical surfaces, the surface number one is the
stop surface.

*/

stopsurface(Position, Aperture) :-
        wavelength([Lambda1]),
        pupil_diameter(Aperture),
        Position is 1,
        outm("Stop surface is surface number 1, 1 surface from the left, its diameter is "),
        outl(Aperture).

/*

The following heuristic indicates the stop surface of a high aperture singlet
lens.
A high aperture singlet lens is composed of two singlet lens mounted back to
back, the stop surface is located between surface number three and surface
number four.

*/

stopsurface(Position, Aperture) :-
        wavelength([Lambda1]),
        usehighaperture(_,_,_,_),
        pupil_diameter(Aperture),
        Position is 3,
        outm("Stop surface is surface number 3, 3 surface from the left, its diameter is "),
        outl(Aperture).
/*
```

The following heuristic indicates the stop surface of a high aperture doublet lens.
A high aperture doublet lens is composed of eight optical surfaces, the surface number four or surface number five is the stop surface.
The stop surface can be an additionaly plane located between surface number four and surface number five.

*/

```
stopsurface(Position, Aperture) :-
        wavelength([Lambda1, Lambda2]),
        usehighaperture(_,_,_,_),
        pupil_diameter(Aperture),
        Position is 5,
        outm("Stop surface is surface number 5, 5 surface from the left, its diameter is "),
        outl(Aperture).
```

/*

The following heuristic indicates the stop surface of a high aperture triplet lens.
A high aperture triplet lens is composed of twelve optical surfaces, the surface number six or surface number seven is the stop surface.
The stop surface can be an additionaly plane located between surface number six and surface number seven.

*/

```
stopsurface(Position, Aperture) :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        usehighaperture(_,_,_,_),
        Position is 7,
        pupil_diameter(Aperture),
        outm("Stop surface is surface number 7, 7 surface from the left, its diameter is "),
        outl(Aperture).
```

/*

The following heuristic defines a long length. If a distance is greater than twenty millimeters, it is considered a long distance.
The following heuristic is used in the case of calculation of an afocal optical system.

*/

```
longlength(Length, Angular_Magnification) :-
        Length >= 20.0.
```

/*

The following heuristic gives the sense of a short length. If a distance is less than twenty millimeters, it is considered a short distance. This heuristic is used in the case of calculation of an afocal optical system.
If the Angular_Magnification is greater or egal to three, then, a distance of 50 mm is considered short distance.

```
*/

shortlength(Length, Angular_Magnification)        :-
        Length < 50.0, abs(Angular_Magnification) >=3.0.

infini(Infini) :-
        Infini is 1.0e+38.

/*
```

The following heuristic defines high temperature. It is used in the assembling algorithm of an optical system. 150 degree is considered as high temperature because the cement melts at this temperature.

```
*/

hightemperature(T) :-
        temperature(T),
        T > 150.0,
        outm("high temperature and ").
/*
```

The following heuristic defines low temperature. It is used in the assembling algorithm of an optical system.

```
*/

lowtemperature(T) :-
        temperature(T), T =< 150.0,
        outm("low temperature and ").

/*
```

The following heuristic states the breakdown conditions.
The value ten of the laser power is arbitrary. The user can enter his own value.

```
*/

breakdownrisk(Laserpower) :-
        laserpower(Laserpower),
        Laserpower > 10.0,
        outml("break down risk ==> ").

/*
```

The following heuristic states the no breakdown condition. The user can enter his own value.

```
*/

nobreakdownrisk(Laserpower) :-
        laserpower(Laserpower),
        Laserpower < 10.0,
        outml("no break down risk ==> ").

/*
```

This rule calculates the thickness of a convergent singlet lens according to its sagitta.
The thickness of a singlet lens is given by the following relation:
Thickness = Sagitta1 + Sagitta2 + Edgethickness.
Where the sagitta1 and sagitta2 are relative to the surface number one and surface number two of the singlet.

```
*/

lensthickness(Optical_Power, Radius1, Radius2, Diameter, Thickness) :-
        edgethickness(Edgethickness, Optical_Power),
        Optical_Power > 0.0,
        sagitta(Optical_Power, Radius1, Diameter, Sagitta1),
        sagitta(Optical_Power, Radius2, Diameter, Sagitta2),
        Thickness is Sagitta1+Sagitta2 + Edgethickness.

/*
```

This rule assigns the thickness of a negative lens to its center thickness.

```
*/

lensthickness(Optical_Power, Radius1, Radius2, Diameter, Thickness) :-
        centerthickness(Centerthickness, Optical_Power),
        Optical_Power < 0.0,
        Thickness is Centerthickness.

/*
```

The following rule defines a convergent lens.
If the focal length of a lens is positive it is called a convergent lens.

```
*/

convergent(Optical_Power) :-
        Optical_Power > 0.0,
        outm(" convergent lens").

/*
```

The following rule defines a divergent lens.
If the focal length of a lens is negative it is called a divergent lens.

```
*/

divergent(Optical_Power) :-
        Optical_Power < 0.0,
        outm(" divergent lens  and ").

/*
```

The following rule defines a negative transversal magnification.

```
*/

negative_transversal_magnification(Transverse_Magnification) :-
        Transverse_Magnification < 0.0,
        outm("negative transverse magnification and ").
/*
```

The following rule defines a positive transversal magnification.

```
*/

positive_transversversal_magnification(Transverse_Magnification) :-
        Transverse_Magnification > 0.0,
        outm("positive transversversal magnification and ").

/*
```

The following rule defines a negative angular magnification.

```
*/

negative_angular_magnification(Angular_Magnification ) :-
        Angular_Magnification < 0.0,
        outm("negative angular magnification and").

/*
```

The following rule defines a positive angular magnification.

```
*/

positive_angular_magnification(Angular_Magnification) :-
        Angular_Magnification > 0.0,
        outm("positive angular magnification and").

/*
```

The following heuristic introduces a small correction to the optical power in case of a triplet lens because of the separation heuristic.

```
*/
```

```
corrigefocal(Optical_Power0, Optical_Power) :-
        power_focal(Focal_Length, Optical_Power0),
        Focal_Length =< 125.0,
        Optical_Power is 2.250*Optical_Power0/3.0.
```

Appendix 7: Heuristics Concerning the Context of Use of Focal Optical Systems

/*

The following heuristics are used to determine the use conditions of an optical system.
The relationship between the following parameters Optical_Power, Transversal_Magnification, L and the Diameter of the optical system are used to identify the use context.

A clause is satisfied if all the conditions of the body of the clause are satisfied.
The comment found in each clause such as, outm("plano-convex conditions -->"), is used to trace the reasoning and the path of the inference engine.

For example, the clause:

useplanoconvex(Optical_Power,Transversal_Magnification,L,Diameter)

is satisfied if all the following rules, convergent(Optical_Power), low_aperture(Optical_Power, Diameter), negative_transversal_magnification (Transversal_Magnification), nearinfiniteratio(Transversal_Magnification), low_power_lens(Optical_Power) and object_at_infinity(Optical_Power, L) are satisfied.

There are ten different context of use of optical systems according to the sign, the ratio and the size of the following parameters: Optical_Power, Transversal_Magnification, L and the Diameter.

The rule below indicates when a plano-convex optical system is used.

*/

```
useplanoconvex(Optical_Power, Transversal_Magnification, L, Diameter) :-
        convergent(Optical_Power),
        low_aperture(Optical_Power, Diameter),
        negative_transversal_magnification(Transversal_Magnification),
        nearinfiniteratio(Transversal_Magnification),
        low_power_lens(Optical_Power),
        object_at_infinity(Optical_Power, L),
        outm("plano-convex conditions --> ").
```

/*

The following heuristic indicates when a bi-convex optical system is used.

*/

```
usebiconvex(Optical_Power, Transversal_Magnification, L, Diameter) :-
        convergent(Optical_Power),
        low_aperture(Optical_Power, Diameter),
        negative_transversal_magnification(Transversal_Magnification),
        high_power_lens(Optical_Power),
        finiteratio(Transversal_Magnification),
        outm("bi-convex conditions --> ").
```

/*

The following heuristic indicates when a bi-convex optical system is used.

*/

```
usebiconvex(Optical_Power, Transversal_Magnification, L, Diameter) :-
        convergent(Optical_Power),
        low_aperture(Optical_Power, Diameter),
        negative_transversal_magnification(Transversal_Magnification),
        unitratio(Transversal_Magnification),
        outm("bi-convex conditions --> ").
```

/*

The following heuristic indicates when a plano-concave optical system is used.

*/

```
useplanoconcave(Optical_Power, Transversal_Magnification, L, Diameter) :-
        divergent(Optical_Power),
        low_aperture(Optical_Power, Diameter),
        positive_transverse_magnification(Transversal_Magnification),
        nearinfiniteratio(Transversal_Magnification),
        low_power_lens(Optical_Power),
        object_at_infinity(Optical_Power, L),
        outm("plano-concave conditions --> ").
```

/*

The following heuristic indicates when an asymetric bi-convex optical system is used.

*/

```
useasymetricbiconvex(Optical_Power, Transversal_Magnification, L, Diameter) :-
        convergent(Optical_Power),
        low_aperture(Optical_Power, Diameter),
        negative_transversal_magnification(Transversal_Magnification),
        notunitratio(Transversal_Magnification),
        low_power_lens(Optical_Power),
        finiteratio(Transversal_Magnification),
        outm("asymetric bi-convex conditions --> ").
```

```
/*
```

The following heuristic indicates when a bi-concave optical system is used.

```
*/

usebiconcave(Optical_Power, Transversal_Magnification, L, Diameter) :-
        divergent(Optical_Power),
        low_aperture(Optical_Power, Diameter),
        positive_transverse_magnification(Transversal_Magnification),
        high_power_lens(Optical_Power),
        finiteratio(Transversal_Magnification),
        outm("biconcave conditions --> ").

/*
```

The following heuristics indicates when an asymetric bi-concave optical system is used.

```
*/

useasymetricbiconcave(Optical_Power, Transversal_Magnification, L, Diameter) :-
        divergent(Optical_Power),
        low_aperture(Optical_Power, Diameter),
        positive_transverse_magnification(Transversal_Magnification),
        low_power_lens(Optical_Power),
        finiteratio(Transversal_Magnification),
        notunitratio(Transversal_Magnification),
        outm("asymetric biconcave condions --> ").

/*
```

The following heuristics indicates when a concave convex optical system is used.

```
*/

useconcaveconvex(Optical_Power, Transversal_Magnification, L, Diameter) :-
        convergent(Optical_Power),
        low_aperture(Optical_Power, Diameter),
        aplanatic(M),
        outm("concave convex conditions --> ").

/*
```

The following heuristic indicates when a concave convex, meniscus optical system is used.

```
*/

useconvexconcave(Optical_Power, Transversal_Magnification, L, Diameter) :-
        divergent(Optical_Power),
        high_aperture(Optical_Power, Diameter),
        aplanatic(M),
```

```
    , outm("concave convex, meniscus,  conditions --> ").

/*

The following heuristic indicates when a high aperture  optical system is
used.

*/

usehighaperture(Optical_Power, Transversal_Magnification, L, Diameter) :-
        high_aperture(Optical_Power, Diameter),
        outm(" high aperture conditions --> ").
```

Appendix 8: Rules used in the Calculation of Singlet Lenses

The following rules are used for the calculation of a singlet lens.
The parameters, Optical_Power, Thickness, Refractive_Index and Diameter, are needed for this calculation. As output , the following clauses return the first and the second radius(Radius1 and Radius2) of the singlet lens as well as the lens thickness at the center.

The trace comment written as (" ---> Plano-convex lens calculations, "), is located at the beginning of the clause and thus, can follow the inference engine reasoning.

One distinguishes nine different simple singlet lenses which are plano-convex, bi-convex, bi-concave, plano-concave, convex-concave, concave-convex, asymetric bi-convex, asymetric bi-concave and high aperture lenses.
The asymetric bi-convex and asymetric bi-concave are called best form lens.
The high aperture lens is made up of two lenses located back to back.

A singlet lens is composed of two optical surfaces and is identified by two radii, radius 1 and radius 2, by its optical power, by its diameter and by its optical glass material as well as its thickness.

```
/*

The following rule calculates a plano-convex singlet lens.
It returns also the thickness, the refractive index, the diameter and the fisrt
and second radius of the singlet lens.

*/

planoconvexlens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index, Diame-
        ter) :-
        nl,
        write(" ---> Plano-convex lens calculations, "),
        infinity(Infini),
        Radius1 is (Refractive_Index - 1.0)/Optical_Power,
        Radius2 is Infini,
        lensthickness(Optical_Power, Radius1, Radius2, Diameter, Thickness).


/*

The following rule calculates a bi-convex singlet lens.
It returns also the thickness, the refractive index, the diameter and the fisrt
and second radius of the singlet lens.

*/
```

```
biconvexlens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index, Diameter) :-
        nl,
        write(" ---> Biconvex lens calculations, "),
        nl,
        Radius1 is 2.0*(Refractive_Index-1.0)/Optical_Power,
        Radius2 is - Radius1,
        lensthickness(Optical_Power, Radius1, Radius2, Diameter, Thickness).

/*

The following rule calculates a bi-concave singlet lens.
It returns also the thickness, the refractive index, the diameter and the fisrt
and second radius of the singlet lens.

*/

biconcavelens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index, Diameter):-
        nl,
        write(" ---> Biconcave lens calculations, "),
        nl,
        Radius1 is (2.0*(Refractive_Index -1.0)/Optical_Power),
        Radius2 is  - Radius1,
        lensthickness(Optical_Power, Radius1, Radius2, Diameter, Thickness).

/*

The following rule calculates a plano-concave singlet lens.
It returns also the thickness, the refractive index, the diameter and the fisrt
and second radius of the singlet lens.

*/

planoconcavelens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index, Diame-
        ter) :-
        write(" ---> Plano-concave lens  calculations, "),
        infinity(Infini),
        Radius1 is (Refractive_Index-1.0)/Optical_Power,
        Radius2 is Infini,
        lensthickness(Optical_Power, Radius1, Radius2, Diameter, Thickness).

/*

The following rule calculates a convex concave singlet lens.
It returns also the thickness, the refractive index, the diameter and the fisrt
and second radius of the singlet lens.
It returns also the thickness, the refractive index, the diameter and the fisrt
and second radius of the singlet lens.

*/
```

```
convexconcavelens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index, Dia-
        meter) :-
        write(" ---> Concave-convex, meniscus lens calculations "),
        nl,
        Optical_Power1 is -Optical_Power/2.0, Optical_Power2 is -3.0*Optical_Power/2.0,
        Radius1 is (Refractive_Index-1.0)/Optical_Power1,
        Radius2 is (Refractive_Index-1.0)/(Optical_Power2),
        lensthickness(Optical_Power, Radius1, Radius2, Diameter, Thickness).

/*
```

The following rule calculates a concave-convex singlet lens.
It returns also the thickness, the refractive index, the diameter and the fisrt and second radius of the singlet lens.

```
*/

concaveconvexlens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index, Dia-
        meter) :-
        nl,
        write(" ---> Concave-convex, meniscus lens calculations "),
        nl,
        Optical_Power1 is Optical_Power/2.0, Optical_Power2 is 3.0*Optical_Power/2.0,
        Radius1 is -(Refractive_Index-1.0)/Optical_Power1,
        Radius2 is -(Refractive_Index-1.0)/(Optical_Power2),
        lensthickness(Optical_Power, Radius1, Radius2, Diameter, Thickness).

/*
```

The following rule calculates a best form asymetric bi-convex singlet lens.
It returns also the thickness, the refractive index, the diameter and the fisrt and second radius of the singlet lens.

```
*/

asymetricbiconvexlens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index,
        Diameter) :-
        nl,
        write(" ---> Best form lens, asymetricbi-convexlens calculations "),
        nl,
        Optical_Power1 is Optical_Power/6.0, Optical_Power2 is (10.0/12.0)* Opti-
        cal_Power,
        Radius1 is ((Refractive_Index -1.0)/Optical_Power1),
        Radius2 is -((Refractive_Index-1.0)/Optical_Power2),
        lensthickness(Optical_Power, Radius1, Radius2, Diameter, Thickness).

/*
```

The following rule calculates a best form asymetric bi-concave singlet lens.
It returns also the thickness, the refractive index, the diameter and the fisrt and second radius of the singlet lens.

```
*/

asymetricbiconcavelens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index,
        Diameter) :-
        nl,
        write(" ---> Best form lens, asymetricbiconcavelens calculations "),
        nl,
        Optical_Power1 is Optical_Power/6.0, Optical_Power2 is (10.0/12.0)* Opti-
        cal_Power,
        Radius1 is ((Refractive_Index -1.0)/Optical_Power1),
        Radius2 is - ((Refractive_Index-1.0)/Optical_Power2),
        lensthickness(Optical_Power, Radius1, Radius2, Diameter, Thickness).

/*
```

The following rule calculates a high aperture singlet lens.
It returns also the thickness, the refractive index, the diameter and second radii of the plano-convex singlet lenses.

```
*/

highaperturelens(Optical_Power1, Optical_Power2, Thickness1, Thickness2, Radius1, Ra-
        dius2, Radius3, Radius4, Refractive_Index, Diameter) :-
        nl,
        write(" ---> high aperture lens calculations "),
        nl,
        planoconvexlens(Optical_Power1, Thickness1, Radius1, Radius2, Refrac-
        tive_Index, Diameter),
        planoconvexlens(Optical_Power2, Thickness2, Radius4, Radius3, Refrac-
        tive_Index, Diameter).
```

Appendix 9: Heuristics Used to Design Doublet Lenses

/*

The following clauses calculate the doublet optical lenses. One distinguishes seven different optical doublet lenses which are plano-convex doublet, bi-convex doublet, plano-concave doublet, bi-concave doublet, asymetric bi-concave doublet, asymetric bi-concave doublet, concave-convex doublet, convex-concave doublet and high aperture doublet.
Each doublet is identified by its optical power, its thickness, its four radii, (Radius1, Radius2, Radius3 and Radius4) its diameter and its two optical glasses.

Each doublet is made up of two singlet lenses of which one must be Flint and the other Crown optical glass. These two singlet lenses are assembled with the clauses assembling doublet.
The high aperture doublet is made up of two doublets assembled back to back.

The following heuristic is used to make a plano-convex doublet lens.
This plano-convex doublet is made up of two singlet lenses, a bi-convex lens and a plano-concave lens.

*/

```
planoconvexdoublet(Optical_Power, Thickness1, Thickness2, Radius1, Radius2, Radius3,
        Radius4, Refractive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2, Dia-
        meter, Lambda1, Lambda2, Separation) :-
        nl,
        outml(" --> plano-convex doublet "),
        assemblingdoublet(Optical_Power, Optical_Power1, Optical_Power2, Refrac-
        tive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2, Diameter, Separation),
        biconvexlens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        planoconcavelens(Optical_Power2, Thickness2, Radius3, Radius4, Refrac-
        tive_Index2, Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda2),
        assemblingthickness(Thickness, Thickness1, Thickness2).
```

/*

The following heuristic is used to make a bi-convex doublet lens.
This bi-convex doublet is made up of two singlet lenses, a bi-convex lens and a concave-convex lens.

*/

```
biconvexdoublet(Optical_Power, Thickness1, Thickness2, Radius1, Radius2, Radius3,
        Radius4, Refractive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2, Dia-
        meter, Lambda1, Lambda2, Separation) :-
        nl,
        outml(" --> biconvex doublet "),
        assemblingdoublet(Optical_Power, Optical_Power1, Optical_Power2, Refrac-
        tive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2, Diameter, Separation),
        biconvexlens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        concaveconvexlens(Optical_Power2, Thickness2, Radius4, Radius3, Refrac-
        tive_Index2, Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda2),
        assemblingthickness(Thickness, Thickness1, Thickness2).

/*
```

The following heuristic is used to make a plano-concave doublet lens.
This plano-convave doublet is made up of two singlet lenses, a bi-concave lens and a plano-convex lens.

```
*/

planoconcavedoublet(Optical_Power, Thickness1, Thickness2, Radius1, Radius2, Radius3,
        Radius4, Refractive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2, Dia-
        meter, Lambda1, Lambda2, Separation) :-
        nl,
        outml(" --> plano-concave doublet "),
        assemblingdoublet(Optical_Power, Optical_Power1, Optical_Power2, Refrac-
        tive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2, Diameter, Separation),
        biconcavelens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        planoconvexlens(Optical_Power2, Thickness2, Radius3, Radius4, Refrac-
        tive_Index2, Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda2),
        assemblingthickness(Thickness, Thickness1, Thickness2).

/*
```

The following heuristic is used to make a bi-concave doublet lens.
This bi-concave doublet is made up of two singlet lenses, two plano-concave lenses which are mounted facing one another.

```
*/

biconcavedoublet(Optical_Power, Thickness1, Thickness2, Radius1, Radius2, Radius3,
        Radius4, Refractive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2, Dia-
        meter, Lambda1, Lambda2, Separation) :-
        nl,
```

```
        outml(" --> biconcave doublet "),
        assemblingdoublet(Optical_Power, Optical_Power1, Optical_Power2, Refrac-
        tive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2, Diameter, Separation),
        planoconcavelens(Optical_Power1, Thickness1, Radius1, Radius2, Refrac-
        tive_Index1, Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        planoconcavelens(Optical_Power2, Thickness2, Radius4, Radius3, Refrac-
        tive_Index2, Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda2),
        assemblingthickness(Thickness, Thickness1, Thickness2).

/*
```

The following heuristic is used to make an asymetric bi-concave doublet lens.

This asymetric bi-concave doublet is made up of two singlet lenses, a plano-concave lens and a second plano-concave singlet lens different from the first one.

```
*/

asymetricbiconcavedoublet(Optical_Power, Thickness1, Thickness2, Radius1, Radius2,
        Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Nam1,
        Glass_Nam2, Diameter, Lambda1, Lambda2, Separation) :-
        nl,
        outml(" --> asymetric biconcave doublet "),
        assemblingdoublet(Optical_Power, Optical_Power1, Optical_Power2, Refrac-
        tive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2, Diameter, Separation),
        planoconcavelens(Optical_Power1, Thickness1, Radius1, Radius2, Refrac-
        tive_Index1, Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        planoconcavelens(Optical_Power2, Thickness2, Radius4, Radius3, Refrac-
        tive_Index2, Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda2),
        assemblingthickness(Thickness, Thickness1, Thickness2).

/*
```

The following heuristic is used to make an asymetric bi-convex doublet lens. This asymetric bi-convex doublet is made up of two singlet lenses, a bi-convex lens and a second concave-convex singlet lens different from the first one.

```
*/

asymetricbiconvexdoublet(Optical_Power, Thickness1, Thickness2, Radius1, Radius2,
        Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Nam1,
        Glass_Nam2, Diameter, Lambda1, Lambda2, Separation) :-
        nl,
        outml(" --> asymetric biconvex doublet "),
```

```
        assemblingdoublet(Optical_Power, Optical_Power1, Optical_Power2, Refrac-
        tive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2, Diameter, Separation),
        biconvexlens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        concaveconvexlens(Optical_Power2, Thickness2, Radius3, Radius4, Refrac-
        tive_Index2, Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda2),
        assemblingthickness(Thickness, Thickness1, Thickness2).

/*
```

The following heuristic is used to make a concave convex doublet lens. This concave-convex doublet is made up of two singlet lenses, a plano-convex lens and a second plano-concave singlet lens different from the first one.

```
*/

concaveconvexdoublet(Optical_Power, Thickness1, Thickness2, Radius1, Radius2, Ra-
        dius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2,
        Diameter, Lambda1, Lambda2, Separation) :-
        nl,
        outml(" --> concave convex doublet "),
        assemblingdoublet(Optical_Power, Optical_Power1, Optical_Power2, Refrac-
        tive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2, Diameter, Separation),
        planoconvexlens(Optical_Power1, Thickness1, Radius1, Radius2, Refrac-
        tive_Index1, Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        planoconcavelens(Optical_Power2, Thickness2, Radius3, Radius4, Refrac-
        tive_Index2, Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda2),
        assemblingthickness(Thickness, Thickness1, Thickness2).

/*
```

The following heuristic is used to make a convex concave doublet lens.

This convex-concave doublet is made up of two singlet lenses, a plano-concave lens and a second plano-convex singlet lens different from the first one.

```
*/

convexconcavedoublet(Optical_Power, Thickness1, Thickness2, Radius1, Radius2, Ra-
        dius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2,
        Diameter, Lambda1, Lambda2, Separation) :-
        nl,
        outml(" --> convex concave doublet "),
```

```
assemblingdoublet(Optical_Power, Optical_Power1, Optical_Power2, Refrac-
tive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2, Diameter, Separation),
planoconcavelens(Optical_Power2, Thickness2, Radius3, Radius4, Refrac-
tive_Index2, Diameter),
output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
Glass_Nam2, Diameter, Lambda2),
planoconvexlens(Optical_Power1, Thickness1, Radius1, Radius2, Refrac-
tive_Index1, Diameter),
output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
Glass_Nam1, Diameter, Lambda1),
assemblingthickness(Thickness, Thickness1, Thickness2).

/*
```

The following heuristic is used to make a high aperture doublet lens.
This high aperture doublet is made up of two doublet lenses mounted back to back, both are plano-convex doublet lenses.
This high aperture lens is made up of four singlet lenses. In reality, it is not a doublet lens but it is considered to be in the same group as the doublet lenses.

```
*/

highaperturedoublet(Optical_Power1, Optical_Power2, Thickness1, Thickness2, Thick-
ness3, Thickness4, Radius1, Radius2, Radius3, Radius4, Radius5, Radius6, Ra-
dius7, Radius8, Refractive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2,
Diameter, Lambda1, Lambda2, Separation) :-
nl,
outm(" --> high aperture two reversed identical achromats calculation "),
planoconvexdoublet(Optical_Power1, Thickness1, Thickness2, Radius1, Radius2,
Radius3, Radius4,
Refractive_Index1, Refractive_Index2, Glass_Nam1, Glass_Nam2, Diameter,
Lambda1, Lambda2, Separation),
planoconvexdoublet(Optical_Power2, Thickness3, Thickness4, Radius5, Radius6,
Radius7, Radius8, Refractive_Index1, Refractive_Index2, Glass_Nam1,
Glass_Nam2, Diameter, Lambda1, Lambda2, Separation).
```

Appendix 10: Heuristics Used to Design Triplet Lenses

/*

A triplet lens is composed of three singlet lenses assembled according to the clause assembling triplet. A triplet lens is identified by its optical power, its thickness, the radius of the three singlet lenses, the refractive indices of the singlet lenses and its thickness. There are seven different possible triplet lenses which are plano-convex triplet, bi-convex triplet, asymetric bi-convex triplet, asymetric bi-concave triplet, plano-concave triplet, bi-concave triplet and concave-convex triplet according to the clauses found below. A triplet can be composed of a doublet lens and a singlet lens or of three singlet lenses. These triplets can be composed of in many possible ways as demonstrated below.

The following heuristic is used to make a plano-convex triplet.

*/

```
planoconvextriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1, Ra-
        dius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diame-
        ter, Lambda1, Lambda2, Lambda3, Separation1, Separation2):-
        assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
        cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
        Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
        Optical_Power1 > 0.0,
        Optical_Power2 < 0.0,
        Optical_Power3 > 0.0,
        outml(" --> plano-convex triplet "),
        planoconvexlens(Optical_Power1, Thickness1, Radius, Radius2, Refrac-
        tive_Index1, Diameter),
        Radius1 is -1.0*Radius,
        output_lens(Optical_Power1, Thickness1, Radius2, Radius1, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        biconcavelens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda1),
        biconvexlens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Diameter),
        output_lens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Glass_Nam3, Diameter, Lambda1).
```

/*

The following heuristic is used to make a plano-convex triplet.

*/

```
planoconvextriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1, Ra-
        dius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refractive_Index2,
        Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Lambda1,
        Lambda2, Lambda3, Separation1, Separation2) :-
        assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
        cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
        Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
        Optical_Power1 < 0.0,
        Optical_Power2 > 0.0,
        Optical_Power3 < 0.0,
        outml(" --> plano-convex triplet "),
        planoconcavelens(Optical_Power1, Thickness1, Radius, Radius2, Refrac-
        tive_Index1, Diameter),
        Radius1 is -1.0*Radius,
        output_lens(Optical_Power1, Thickness1, Radius2, Radius1, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        biconvexlens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda1),
        convexconcavelens(Optical_Power3, Thickness3, Radius5, Radius6, Refrac-
        tive_Index3, Diameter),
        output_lens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Glass_Nam3, Diameter, Lambda1).

/*

The following heuristic is used to make a plano-convex triplet.

*/

planoconvextriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1, Ra-
        dius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refractive_Index2,
        Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Lambda1,
        Lambda2, Lambda3, Separation1, Separation2) :-
        assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
        cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
        Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
        Optical_Power1 < 0.0,
        Optical_Power2 > 0.0,
        Optical_Power3 > 0.0,
        outml(" --> plano-convex triplet "),
        planoconcavelens(Optical_Power1, Thickness1, Radius, Radius2, Refrac-
        tive_Index1, Diameter),
        Radius1 is -1.0*Radius,
        output_lens(Optical_Power1, Thickness1, Radius2, Radius1, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        biconvexlens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda1),
        planoconvexlens(Optical_Power3, Thickness3, Radius5, Radius6, Refrac-
        tive_Index3, Diameter),
        output_lens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Glass_Nam3, Diameter, Lambda1).
```

```
/*
```

The following heuristic is used to make a bi-convex triplet.

```
*/

biconvextriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1, Radius2,
        Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refractive_Index2, Re-
        fractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Lambda1,
        Lambda2, Lambda3, Separation1, Separation2) :-
        assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
        cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
        Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
        Optical_Power1 > 0.0,
        Optical_Power2 < 0.0,
        Optical_Power3 > 0.0,
        outml(" --> biconvextriplet "),
        biconvexlens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        biconcavelens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda1),
        biconvexlens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Diameter),
        output_lens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Glass_Nam3, Diameter, Lambda1).

/*
```

The following heuristic is used to make a bi-convex triplet.

```
*/

biconvextriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1, Radius2,
        Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refractive_Index2, Re-
        fractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Lambda1,
        Lambda2, Lambda3, Separation1, Separation2) :-
        assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
        cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
        Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
        Optical_Power1 < 0.0,
        Optical_Power2 > 0.0,
        Optical_Power3 < 0.0,
        outml(" --> biconvextriplet "),
        convexconcavelens(Optical_Power1, Thickness1, Radius1, Radius2, Refrac-
        tive_Index1, Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        biconvexlens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda1),
```

```
        convexconcavelens(Optical_Power3, Thickness3, Radius5, Radius6, Refrac-
        tive_Index3, Diameter),
        output_lens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Glass_Nam3, Diameter, Lambda1).

/*

The following heuristic is used to make a bi-convex triplet.

*/

biconvextriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1, Radius2,
        Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refractive_Index2, Re-
        fractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Lambda1,
        Lambda2, Lambda3, Separation1, Separation2) :-
        assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
        cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
        Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
        Optical_Power1 < 0.0,
        Optical_Power2 > 0.0,
        Optical_Power3 > 0.0,
        outml(" --> biconvextriplet "),
        convexconcavelens(Optical_Power1, Thickness1, Radius1, Radius2, Refrac-
        tive_Index1, Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        biconvexlens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda1),
        biconvexlens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Diameter),
        output_lens(Optical_Power3, Thickness3, Radius6, Radius5, Refractive_Index3,
        Glass_Nam3, Diameter, Lambda1).

/*

The following heuristic is used to make an asymetric bi-convex triplet.

*/

asymetricbiconvextriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
        Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diame-
        ter, Lambda1, Lambda2, Lambda3, Separation1, Separation2) :-
        assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
        cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
        Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
        Optical_Power1 > 0.0,
        Optical_Power2 < 0.0,
        Optical_Power3 > 0.0,
        outml(" --> asymetric biconvex triplet "),
        biconvexlens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
```

```
        asymetricbiconcavelens(Optical_Power2, Thickness2, Radius3, Radius4, Refrac-
        tive_Index2, Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda1),
        asymetricbiconvexlens(Optical_Power3, Thickness3, Radius5, Radius6, Refrac-
        tive_Index3, Diameter),
        output_lens(Optical_Power3, Thickness3, -Radius6, -Radius5, Refractive_Index3,
        Glass_Nam3, Diameter, Lambda1).

/*
```

The following heuristic is used to make an asymetric bi-convex triplet.

```
*/

asymetricbiconvextriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
        Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diame-
        ter, Lambda1, Lambda2, Lambda3, Separation1, Separation2) :-
        assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
        cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
        Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
        Optical_Power1 < 0.0,
        Optical_Power2 > 0.0,
        Optical_Power3 < 0.0,
        outml(" --> asymetric biconvex triplet "),
        convexconcavelens(Optical_Power1, Thickness1, Radius1, Radius2, Refrac-
        tive_Index1, Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        biconvexlens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda1),
        convexconcavelens(Optical_Power3, Thickness3, Radius5, Radius6, Refrac-
        tive_Index3, Diameter),
        output_lens(Optical_Power3, Thickness3, Radius6, Radius5, Refractive_Index3,
        Glass_Nam3, Diameter, Lambda1).

/*
```

The following heuristic is used to make an asymetric bi-convex triplet.

```
*/

asymetricbiconvextriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
        Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diame-
        ter, Lambda1, Lambda2, Lambda3, Separation1, Separation2) :-
        assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
        cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
        Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
        Optical_Power1 < 0.0, Optical_Power2 > 0.0, Optical_Power3 > 0.0,
        outml(" --> asymetric biconvex triplet "),
        convexconcavelens(Optical_Power1, Thickness1, Radius1, Radius2, Refrac-
        tive_Index1, Diameter),
```

```
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        biconvexlens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda1),
        biconvexlens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Diameter),
        output_lens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Glass_Nam3, Diameter, Lambda1).

/*

The following heuristic is used to make a plano-concave triplet.

*/

planoconcavetriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1, Ra-
        dius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refractive_Index2,
        Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Lambda1,
        Lambda2, Lambda3, Separation1, Separation2) :-
        assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
        cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
        Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
        Optical_Power1 > 0.0,
        Optical_Power2 < 0.0,
        Optical_Power3 > 0.0,
        outml(" --> plano-concave triplet "),
        planoconvexlens(Optical_Power1, Thickness1, Radius, Radius2, Refrac-
        tive_Index1, Diameter),
        Radius1 is -1.0*Radius,
        output_lens(Optical_Power1, Thickness1, Radius2, Radius1, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        biconcavelens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Namv2, Diameter, Lambda1),
        concaveconvexlens(Optical_Power3, Thickness3, Radius5, Radius6, Refrac-
        tive_Index3, Diameter),
        output_lens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Glass_Nam3, Diameter, Lambda1).

/*

The following heuristic is used to make a plano-concave triplet.

*/

planoconcavetriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1, Ra-
        dius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refractive_Index2,
        Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Lambda1,
        Lambda2, Lambda3, Separation1, Separation2) :-
        assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
        cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
        Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
        Optical_Power1 < 0.0, Optical_Power2 > 0.0, Optical_Power3 < 0.0,
```

```
        outml(" --> plano-concave triplet "),
        planoconcavelens(Optical_Power1, Thickness1, Radius, Radius2, Refrac-
        tive_Index1, Diameter),
        Radius1 is -1.0*Radius,
        output_lens(Optical_Power1, Thickness1, Radius2, Radius1, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        biconvexlens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Namv2, Diameter, Lambda1),
        biconcavelens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Diameter),
        output_lens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Glass_Nam3, Diameter, Lambda1).
```

/*

The following heuristic is used to make a plano-concave triplet.

*/

```
planoconcavetriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1, Ra-
        dius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refractive_Index2,
        Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Lambda1,
        Lambda2, Lambda3, Separation1, Separation2) :-
        assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
        cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
        Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
        Optical_Power1 < 0.0, Optical_Power2 > 0.0, Optical_Power3 > 0.0,
        outml(" --> plano-concave triplet "),
        planoconcavelens(Optical_Power1, Thickness1, Radius, Radius2,
        Refractive_Index1, Diameter),
        Radius1 is -1.0*Radius,
        output_lens(Optical_Power1, Thickness1, Radius2, Radius1, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        biconvexlens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Namv2, Diameter, Lambda1),
        concaveconvexlens(Optical_Power3, Thickness3, Radius5, Radius6, Refrac-
        tive_Index3, Diameter),
        output_lens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Glass_Nam3, Diameter, Lambda1).
```

/*

The following heuristic is used to make a bi-concave triplet.

*/

```
biconcavetriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1, Radius2,
        Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refractive_Index2, Re-
        fractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Lambda1,
        Lambda2, Lambda3, Separation1, Separation2) :-
        assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
        cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
```

```
        Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
        Optical_Power1 < 0.0,
        Optical_Power2 > 0.0,
        Optical_Power3 < 0.0,
        outml(" --> biconcave triplet "),
        convexconcavelens(Optical_Power1, Thickness1, Radius1, Radius2, Refrac-
        tive_Index1, Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        biconvexlens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda1),
        biconcavelens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Diameter),
        output_lens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Glass_Nam3, Diameter, Lambda1).

/*

The following heuristic is used to make a bi-concave triplet.

*/

biconcavetriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1, Radius2,
        Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refractive_Index2, Re-
        fractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Lambda1,
        Lambda2, Lambda3, Separation1, Separation2) :-
        assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
        cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
        Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
        Optical_Power1 < 0.0, Optical_Power2 > 0.0, Optical_Power3 > 0.0,
        outml(" --> biconcave triplet "),
        convexconcavelens(Optical_Power1, Thickness1, Radius1, Radius2, Refrac-
        tive_Index1, Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter, Lambda1),
        biconvexlens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter, Lambda1),
        concaveconvexlens(Optical_Power3, Thickness3, Radius5, Radius6, Refrac-
        tive_Index3, Diameter),
        output_lens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
        Glass_Nam3, Diameter, Lambda1).

/*

The following heuristic is used to make an asymetrical bi-concave triplet.

*/

asymetricbiconcavetriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
        Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
```

```
tive_Index2, Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diame-
ter, Lambda1, Lambda2, Lambda3, Separation1, Separation2) :-
assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
Optical_Power1 < 0.0, Optical_Power2 > 0.0, Optical_Power3 < 0.0,
outml(" --> asymetric biconcave triplet "),
convexconcavelens(Optical_Power1, Thickness1, Radius1, Radius2, Refrac-
tive_Index1, Diameter),
output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
Glass_Nam1, Diameter, Lambda1),
biconvexlens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
Diameter),
output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
Glass_Nam2, Diameter, Lambda1),
biconcavelens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
Diameter),
output_lens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
Glass_Nam3, Diameter, Lambda1).

/*

The following heuristic is used to make an asymetrical bi-concave triplet.

*/

asymetricbiconcavetriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
tive_Index2, Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diame-
ter, Lambda1, Lambda2, Lambda3, Separation1, Separation2) :-
assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Opti-
cal_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3,
Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
Optical_Power1 < 0.0,
Optical_Power2 > 0.0,
Optical_Power3 > 0.0,
outml(" --> asymetric biconcave triplet "),
convexconcavelens(Optical_Power1, Thickness1, Radius1, Radius2, Refrac-
tive_Index1, Diameter),
output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
Glass_Nam1, Diameter, Lambda1),
biconvexlens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
Diameter),
output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
Glass_Nam2, Diameter, Lambda1),
concaveconvexlens(Optical_Power3, Thickness3, Radius5, Radius6, Refrac-
tive_Index3, Diameter),
output_lens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3,
Glass_Nam3, Diameter, Lambda1).

/*

The following heuristic is used to make a concave convex triplet.

*/
```

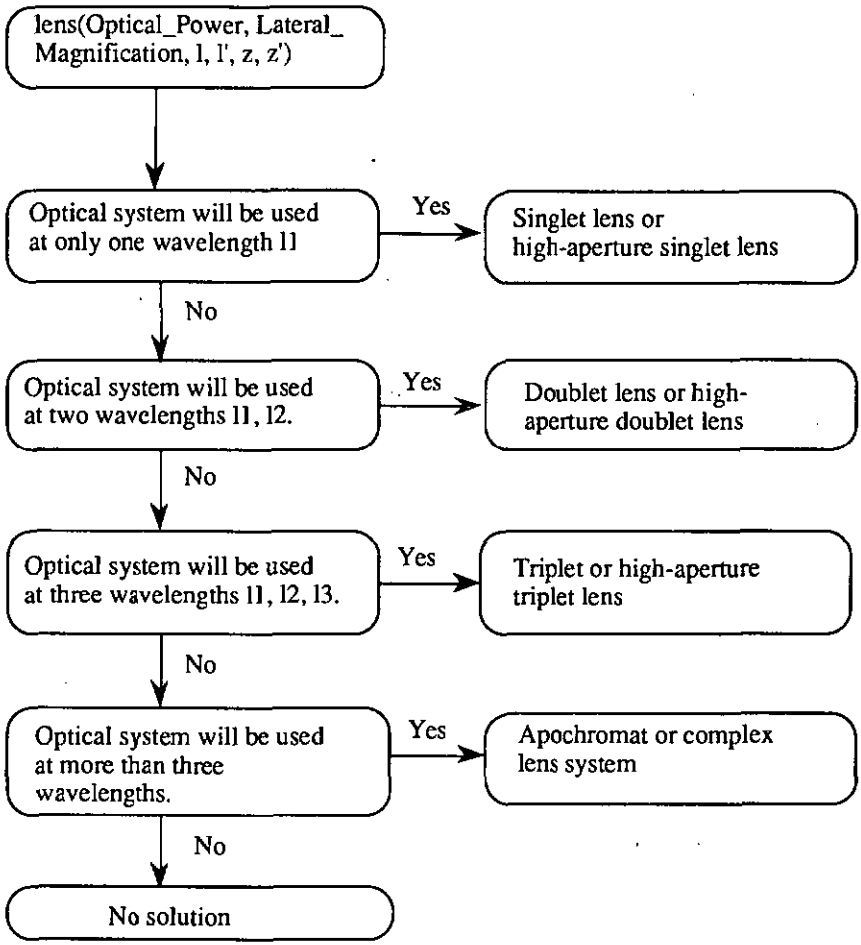
```
concaveconvextriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1, Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refractive_Index2, Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2) :-
        assemblingtriplet(Optical_Power, Optical_Power1, Optical_Power2, Optical_Power3, Refractive_Index1, Refractive_Index2, Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2),
        Optical_Power1 > 0.0,
        Optical_Power2 < 0.0,
        Optical_Power3 > 0.0,
        outml(" --> concave convex triplet "),
        asymetricbiconvexlens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1, Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index1, Glass_Nam1, Diameter, Lambda1),
        asymetricbiconcavelens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2, Diameter),
        output_lens(Optical_Power2, Thickness2, Radius3, Radius4, Refractive_Index2, Glass_Nam2, Diameter, Lambda1),
        concaveconvexlens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3, Diameter),
        output_lens(Optical_Power3, Thickness3, Radius5, Radius6, Refractive_Index3, Glass_Nam3, Diameter, Lambda1).

/*
```

The following heuristic is used for a making high aperture triplet .
This high aperture triplet is made up of six singlet lenses: two triplet lenses mounted face to face.
In reality, it is not a triplet lens, but it is categorised with the triplet lens group.
It is considered as a complex optical lens.

```
*/

highaperturetriplet(Optical_Powera, Optical_Powerb, Thickness1a, Thickness2a, Thickness3a, Thickness1b, Thickness2b, Thickness3b, Radius1a, Radius2a, Radius3a, Radius4a, Radius5a, Radius6a, Radius1b, Radius2b, Radius3b, Radius4b, Radius5b, Radius6b, Refractive_Index1, Refractive_Index2, Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Separation1, Separation2) :-
        outml(" --> high aperture triplet "),
        wavelength([Lambda1, Lambda2, Lambda3]),
        planoconvextriplet(Optical_Powera, Thickness1a, Thickness2a, Thickness3a, Radius1a, Radius2a, Radius3a, Radius4a, Radius5a, Radius6a, Refractive_Index1, Refractive_Index2, Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        planoconvextriplet(Optical_Powerb, Thickness1b, Thickness2b, Thickness3b, Radius1b, Radius2b, Radius3b, Radius4b, Radius5b, Radius6b, Refractive_Index1, Refractive_Index2, Refractive_Index3, Glass_Nam1, Glass_Nam2, Glass_Nam3, Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2).
```

Appendix 11: Searching Algorithms of Focal Optical Systems

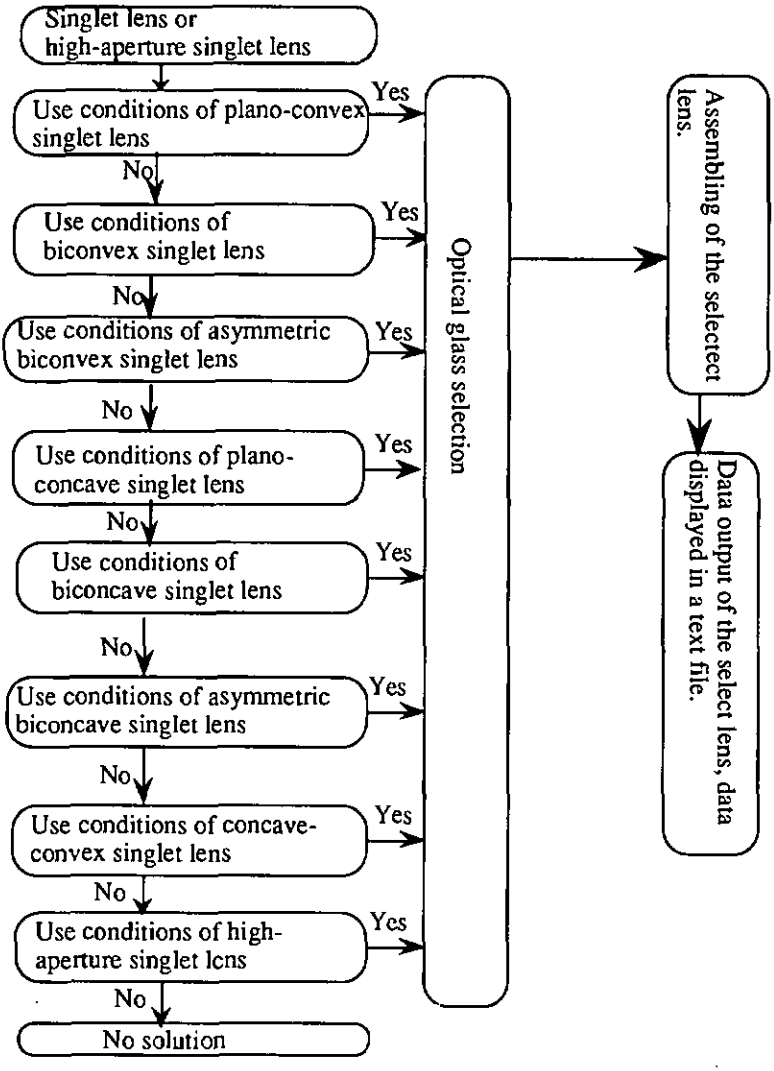
The algorithm displayed in this diagram searches for and identifies the focal optical systems.



Searching algorithms of focal optical systems

Appendix 12: Rules and Searching Algorithms of Singlet Lenses

The below algorithm searches within the single lens and large aperture singlet lens set in order to find the appropriate lens for the desired application. This search strategy is based on heuristics.



Searching algorithms of a singlet lens

The searching algorithms of singlet lenses are coded in the programming language Prolog as displayed below.

```
/*

If only one wavelength will be used, wavelength([Lambda1]), then the use
condition useplanoconvex(Optical_Power, Transversal_Magnification, L,
Diameter), will be tested then the predicate lens indice, lensin-
dice([Glass_Name, Refractive_Index]), is activated then the predicate, pla-
noconvexlens(Optical_Power, Thickness, Radius1, Radius2, Refrac-
tive_Index, Diameter), is activated and finally the lens data is displayed ac-
cording to the predicate output_lens(Optical_Power, Thickness, Radius1,
Radius2, Refractive_Index, Glass_Name, Diameter, Lambda1).
The predicate, stop_surface(Position, entrance_Diameter), proposes the
position and the diameter of the stop surface.
The predicate lensindice gives the refractive index of the optical glass.

*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1]),
        useplanoconvex(Optical_Power, Transversal_Magnification, L, Diameter),
        lensindice([Glass_Name, Refractive_Index]),
        planoconvexlens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index,
        Diameter),
        output_lens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index,
        Glass_Name, Diameter, Lambda1),
        stop_surface(Position, entrance_Diameter).

/*

The following rule generates a bi-convex lens.
The predicate wavelength tests if only one wavelength will be used.
The predicate usebiconvex tests if the bi-convex use conditions are satisfied.
The predicate lensindice gives the refractive index of the optical glass.
The predicate biconvexlens gives the data of the bi-convex singlet lens.
The predicate output_lens displays the data of the bi-convex singlet lens.
The predicate stop_surface returns the diameter and the position of the
stop surface.

*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1]),
        usebiconvex(Optical_Power, Transversal_Magnification, L, Diameter),
        lensindice([Glass_Name, Refractive_Index]),
        biconvexlens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index,
        Diameter),
        output_lens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index,
        Glass_Name, Diameter, Lambda1),
        stop_surface(Position, entrance_Diameter).
```

```
/*

The following rule generates an asymetric bi-convex lens.
The predicate wavelength tests if only one wavelength will be used.
The predicate useasymetricbiconvex tests if the asymetric bi-convex use
conditions are satisfied.
The predicate  lensindice gives the refractive index of the optical glass.
The predicate  asymetricbiconvexlens gives the data of the asymetric bi-
convex singlet lens.
The predicate output_lens displays the data of the  asymetric bi-convex sin-
glet lens.
The predicate stop_surface  returns the diameter and the position of the
stop surface.

*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1]),
        useasymetricbiconvex(Optical_Power, Transversal_Magnification, L, Diameter),
        lensindice([Glass_Name, Refractive_Index]),
        asymetricbiconvexlens(Optical_Power, Thickness, Radius1, Radius2, Refrac-
        tive_Index, Diameter),
        output_lens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index,
        Glass_Name, Diameter, Lambda1),
        stop_surface(Position, entrance_Diameter).

/*

The following rule generates a plano-concave lens.
The predicate wavelength tests if only one wavelength will be used.
The predicate  lensindice gives the refractive index of the optical glass.
The predicate useplanoconcave tests if the plano-concave use conditions are
satisfied.
The predicate  planoconcavelens gives the data of the asymetric plano-con-
cave singlet lens.
The predicate output_lens displays the data of the plano-concave singlet
lens.
The predicate stop_surface  returns the diameter and the position of the
stop surface.

*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1]),
        useplanoconcave(Optical_Power, Transversal_Magnification, L, Diameter),
        lensindice([Glass_Name, Refractive_Index]),
        planoconcavelens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index,
        Diameter),
```

```
        output_lens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index,
        Glass_Name, Diameter, Lambda1),
        stop_surface(Position, entrance_Diameter).

/*
```

The following rule generates a bi-concave lens.
The predicate wavelength tests if only one wavelength will be used.
The predicate lensindice gives the refractive index of the optical glass.
The predicate usebiconcave tests if the bi-concave use conditions are satisfied.
The predicate biconcavelens gives the data of the bi-concave singlet lens.
The predicate output_lens displays the data of the bi-concave singlet lens.
The predicate stop_surface returns the diameter and the position of the stop surface.

```
*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1]),
        usebiconcave(Optical_Power, Transversal_Magnification, L, Diameter),
        lensindice([Glass_Name, Refractive_Index]),
        biconcavelens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index,
        Diameter),
        output_lens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index,
        Glass_Name, Diameter, Lambda1),
        stop_surface(Position, entrance_Diameter).

/*
```

The following rule generates an asymetric bi-concave lens.
The predicate wavelength tests if only one wavelength will be used.
The predicate lensindice gives the refractive index of the optical glass.
The predicate useasymetricbiconcave tests if the asymetric bi-concave use conditions are satisfied.
The predicate asymetricbiconcavelens gives the data of the asymetric bi-concave singlet lens.
The predicate output_lens displays the data of the asymetric bi-concave singlet lens.
The predicate stop_surface returns the diameter and the position of the stop surface.

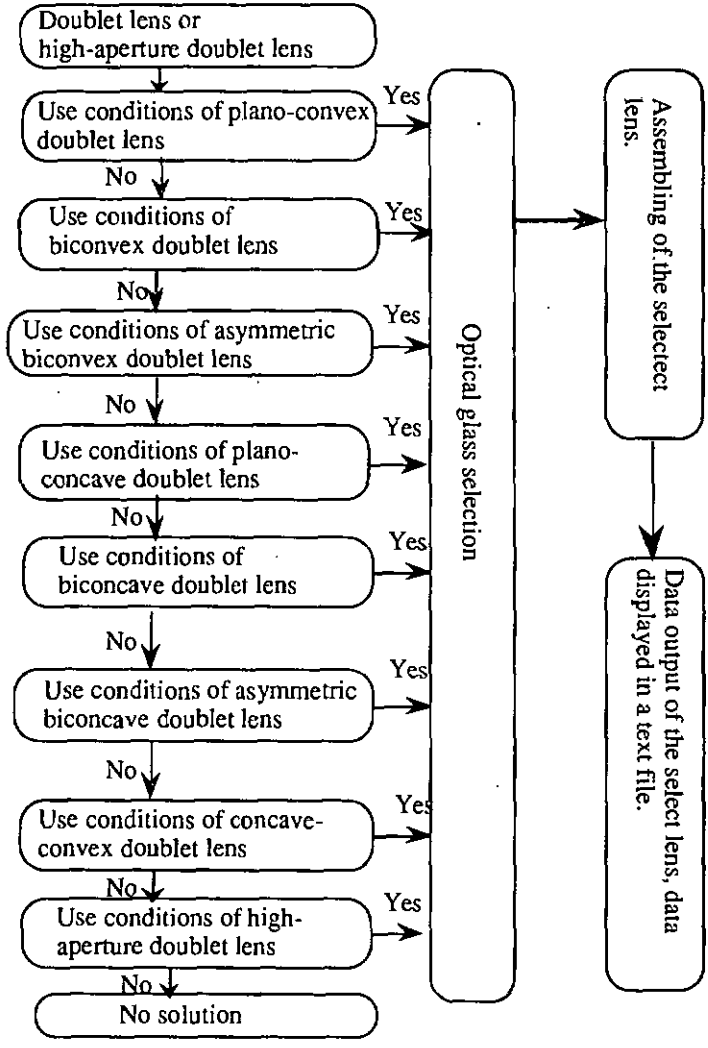
```
*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1]),
        useasymetricbiconcave(Optical_Power, Transversal_Magnification, L, Diameter),
        lensindice([Glass_Name, Refractive_Index]),
        asymetricbiconcavelens(Optical_Power, Thickness, Radius1, Radius2, Refrac-
        tive_Index, Diameter),
```

```
        output_lens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index,
        Glass_Name, Diameter, Lambda1),
        stop_surface(Position, entrance_Diameter).

/*

The following rule generates a concave-convex lens with a positive optical
power.
The predicate wavelength tests if only one wavelength will be used.
The predicate  lensindice gives the refractive index of the optical glass.
The predicate useconcaveconvex tests if the concaveconvex use conditions
are satisfied.
The predicate concaveconvexlens gives the data of the concave-convex sin-
glet lens.
The predicate output_lens displays the data of the concave-convex singlet
lens.
The predicate stop_surface  returns the diameter and the position of the
stop surface.

*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1]),
        useconcaveconvex(Optical_Power, Transversal_Magnification, L, Diameter),
        lensindice([Glass_Name, Refractive_Index]),
        Optical_Power > 0.0,
        Radius1 is Si/(Refractive_Index +1.0),
        concaveconvexlens(Optical_Power, Thickness, Radius1, _, Refractive_Index, Dia-
        meter), Radius2 is So - Thickness, MenOptical_Power is (Refractive_Index-1.0)*(
        ((Refractive_Index+1.0)/Si) + 1.0/(So - Thickness)),
        output_lens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index,
        Glass_Name, Diameter, Lambda1),
        stop_surface(Position, entrance_Diameter).

/*

The following rule generates a concave-convex lens.
The predicate wavelength tests if only one wavelength will be used.
The predicate  lensindice gives the refractive index of the optical glass.
The predicate useconcaveconvex tests if the concaveconvex use conditions
are satisfied.
The predicate concaveconvexlens gives the data of the concave-convex sin-
glet lens.
The predicate output_lens displays the data of the  concave-convex singlet
lens.
The predicate stop_surface  returns the diameter and the position of the
stop surface.

*/
```

```
lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1]),
        useconcaveconvex(Optical_Power, Transversal_Magnification, L, Diameter),
        convexconcave(Optical_Power, Transversal_Magnification, L, Diameter),
        lensindice([Glass_Name, Refractive_Index]),
        concaveconvexlens(Optical_Power, Thickness, Radius1, Radius2, Refrac-
        tive_Index, Diameter),
        output_lens(Optical_Power, Thickness, Radius1, Radius2, Refractive_Index,
        Glass_Name, Diameter, Lambda1),
        stop_surface(Position, entrance_Diameter).

/*

The predicate below generates a high aperture singlet lens  used at only one
wavelength and tests whether the high aperture conditions are satisfied.
The predicate usehighaperture tests the high aperture conditions.
This high aperture singlet is made up of two singlet lenses mounted back to
back.
The predicate highaperturelens returns the data of the high aperture singlet.
The predicate output lens displays the data of the singlet lens.
The predicate stop surface displays the position and the diameter of the stop
surface.
The stop surface is located at the symetrical plane i.e. between the two sin-
glets.

*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1]),
        usehighaperture(Optical_Power, Transversal_Magnification, L, Diameter),
        lensindice([Glass_Name, Refractive_Index]),
        outm(" Reversed  singlet --> "),
        Optical_Power2 is 1.0/So, Optical_Power1 is 1.0/L,
        highaperturelens(Optical_Power1, Optical_Power2, Thickness1, Thickness2, Ra-
        dius1, Radius2, Radius3, Radius4, Refractive_Index, Diameter),
        output_lens(Optical_Power1, Thickness1, Radius1, Radius2, Refractive_Index,
        Glass_Name, Diameter, Lambda1),
        output_lens(Optical_Power2, Thickness2, Radius4, Radius3, Refractive_Index,
        Glass_Name, Diameter, Lambda1).
        stopsurface(Position, entrance_Diameter).
```

Appendix 13: Searching Algorithms of Doublet Lenses

The below algorithm searches within the doublet lens and large aperture doublet lens set in order to find the appropriate lens for the desired application. This search strategy is also based on heuristics.



Searching algorithms of doublet lenses.

The searching algorithms of doublet lenses are coded in the programming language Prolog as displayed below

```
/*

The folowing rule is applied if the optical system will be used at two wavelengths.
The predicate below tests wavelength([Lambda1, Lambda2]) and the use conditions.
If the use conditions of a plano-convex doublet are satisfied then this doublet is calculated and the data is displayed according to the predicate output_doublet_lens.

*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2]),
        useplanoconvex(Optical_Power, Transversal_Magnification, L, Diameter),
        planoconvexdoublet(Optical_Power, Thickness1, Thickness2, Radius1, Radius2,
        Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        output_doublet_lens(Optical_Power, Thickness1, Thickness2, Radius1, Radius2,
        Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        stop_surface(Position, entrance_Diameter).

/*

The following rule generates a bi-convex doublet lens used at two wavelengths.
The predicate wavelength tests if the doublet lens will be used at two wavelengths.
The predicate usebiconvex tests if the bi-convex use conditions are satisfied.
The predicate biconvexdoublet returns the data of the bi-convex doublet lens.
The predicate output_doublet_lens displays the data of the doublet lens.
The predicate stop_surface returns the diameter and the position of the stop surface.

*/


lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2]),
        usebiconvex(Optical_Power, Transversal_Magnification, L, Diameter),
        biconvexdoublet(Optical_Power, Thickness1, Thickness2, Radius1, Radius2, Ra-
        dius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
```

```
        output_doublet_lens(Optical_Power, Thickness1, Thickness2, Radius1, Radius2,
        Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        stop_surface(Position, entrance_Diameter).

/*

The following rule generates an asymetric bi-convex doublet lens used at
two wavelengths.
The predicate wavelength tests if the doublet lens will be used at two wave-
lengths.
The predicate useasymetricbiconvex tests if the asymetric bi-convex use
conditions are satisfied.
The predicate asymetricbiconvexdoublet returns the data of the asymetric
bi-convex doublet lens.
The predicate output_doublet_lens displays the data of the doublet lens.
The predicate stop_surface returns the diameter and the position of the stop
surface.

*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2]),
        useasymetricbiconvex(Optical_Power, Transversal_Magnification, L, Diameter),
        asymetricbiconvexdoublet(Optical_Power, Thickness1, Thickness2, Radius1, Ra-
        dius2, Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        output_doublet_lens(Optical_Power, Thickness1, Thickness2, Radius1, Radius2,
        Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        stop_surface(Position, entrance_Diameter).

/*

The following rule generates a plano-concave doublet lens used at two wa-
velengths.
The predicate wavelength tests if the doublet lens will be used at two wave-
lengths.
The predicate useplanoconcave tests if the plano-concave use conditions are
satisfied.
The predicate planoconcavedoublet returns the data of the plano-concave
doublet lens.
The predicate output_doublet_lens displays the data of the doublet lens.
The predicate stop_surface returns the diameter and the position of the stop
surface.

*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2]),
```

```
        useplanoconcave(Optical_Power, Transversal_Magnification, L, Diameter),
        planoconcavedoublet(Optical_Power, Thickness1, Thickness2, Radius1, Radius2,
        Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lamda1, Lambda2, Separation),
        output_doublet_lens(Optical_Power, Thickness1, Thickness2, Radius1, Radius2,
        Radius3, Radius4, Refractive_Index1, Refractive_Index2,
        Nam1, Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        stop_surface(Position, entrance_Diameter).

/*
```

The following rule generates a bi-concave doublet lens used at two wavelengths.
The predicate wavelength tests if the doublet lens will be used at two wavelengths.
The predicate usebiconcave tests if the bi-concave use conditions are satisfied.
The predicate biconcavedoublet returns the data of the bi-concave doublet lens.
The predicate output_doublet_lens displays the data of the doublet lens.
The predicate stop_surface returns the diameter and the position of the stop surface.

```
*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2]),
        usebiconcave(Optical_Power, Transversal_Magnification, L, Diameter),
        biconcavedoublet(Optical_Power, Thickness1, Thickness2, Radius1, Radius2, Ra-
        dius3, Radius4, Refractive_Index1, Refractive_Index2,
        Nam1, Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        output_doublet_lens(Optical_Power, Thickness1, Thickness2, Radius1, Radius2,
        Radius3, Radius4, Refractive_Index1, Refractive_Index2,
        Nam1, Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        stop_surface(Position, entrance_Diameter).

/*
```

The following rule generates an asymetric bi-concave doublet lens used at two wavelengths.
The predicate wavelength tests if the doublet lens will be used at two wavelengths.
The predicate useasymetricbiconcave tests if the asymetric bi-concave use conditions are satisfied.
The predicate asymetricbiconcavedoublet returns the data of the asymetric bi-concave doublet lens.
The predicate output_doublet_lens displays the data of the doublet lens.
The predicate stop_surface returns the diameter and the position of the stop surface.

```
*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2]),
        useasymetricbiconcave(Optical_Power, Transversal_Magnification, L, Diameter),
        asymetricbiconcavedoublet(Optical_Power, Thickness1, Thickness2, Radius1, Ra-
        dius2, Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        output_doublet_lens(Optical_Power, Thickness1, Thickness2, Radius1, Radius2,
        Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        stop_surface(Position, entrance_Diameter).

/*
```

The following rule generates a concave-convex doublet lens used at two wavelengths.
The predicate wavelength tests if the doublet lens will be used at two wavelengths.
The predicate useconcaveconvex tests if the concave-convex use conditions are satisfied.
The predicate concaveconvexdoublet returns the data of the concave-convex doublet lens.
The predicate output_doublet_lens displays the data of the concave-convex doublet lens.
The predicate stop_surface returns the diameter and the position of the stop surface.

```
*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2]),
        useconcaveconvex(Optical_Power, Transversal_Magnification, L, Diameter),
        concaveconvexdoublet(Optical_Power, Thickness1, Thickness2, Radius1, Radius2,
        Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        output_doublet_lens(Optical_Power, Thickness1, Thickness2, Radius1, Radius2,
        Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        stop_surface(Position, entrance_Diameter).

/*
```

The following rule generates a convex-concave doublet lens used at two wavelengths.
The predicate wavelength tests if the doublet lens will be used at two wavelengths.
The predicate useconvexconcave tests if the convex-concave use conditions are satisfied.
The predicate convexconcavedoublet returns the data of the convex-concave doublet lens.

The predicate output_doublet_lens displays the data of the convex-concave doublet lens.
The predicate stop_surface returns the diameter and the position of the stop surface.

*/

```
lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2]),
        useconvexconcave(Optical_Power, Transversal_Magnification, L, Diameter),
        convexconcavedoublet(Optical_Power, Thickness1, Thickness2, Radius1, Radius2,
        Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        nl,
        output_doublet_lens(Optical_Power, Thickness1, Thickness2, Radius1, Radius2,
        Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        stop_surface(Position, entrance_Diameter).
```

/*

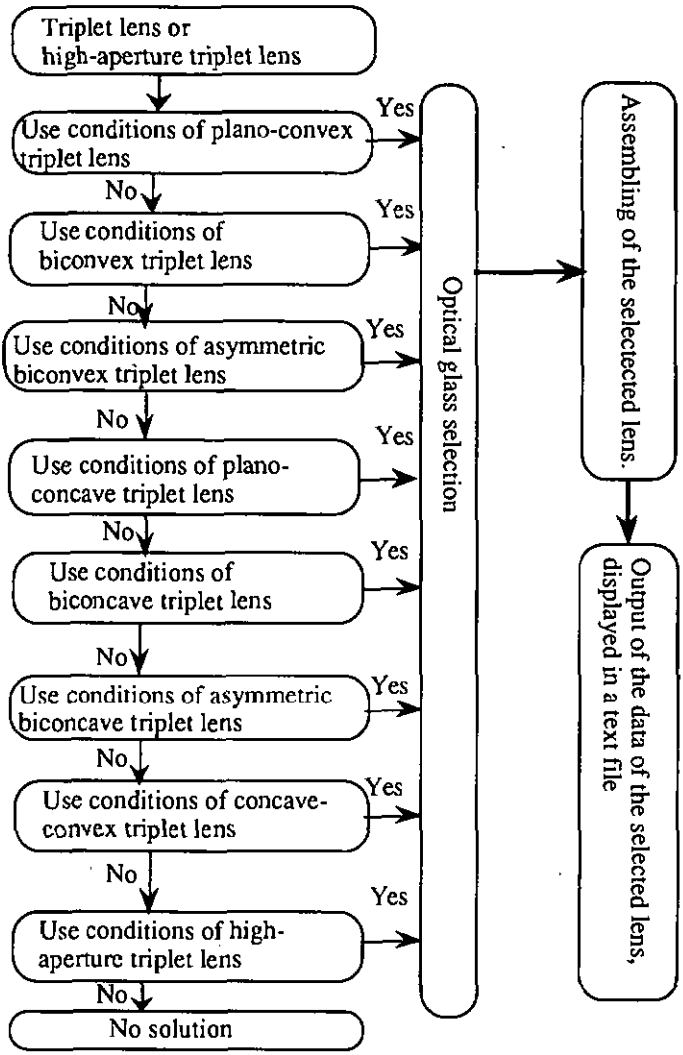
This rule is used at two wavelengths and at high aperture conditions.
This rule explains the mounting of high aperture doublet lenses.
The high aperture doublet lenses are made up of two doublet lenses mounted back to back.
The predicate stop_surface returns the position and the diameter of the stop surface.
In this case, the stop surface is located in the middle of the four lenses, the symetrical plane i.e. between the two doublet lens.

*/

```
lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2]),
        usehighaperture(Optical_Power, Transversal_Magnification, L, Diameter),
        outm(" Reversed achromat --> "),
        Optical_Power2 is 1.0/So,
        Optical_Power1 is 1.0/L,
        highaperturedoublet(Optical_Power1, Optical_Power2, Thickness1, Thickness2,
        Thickness3, Thickness4, Radius1, Radius2, Radius3, Radius4, Radius5, Radius6,
        Radius7, Radius8, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        output_doublet_lens(Optical_Power1, Thickness1, Thickness2, Radius1, Radius2,
        Radius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        output_doublet_lens(Optical_Power2, Thickness3, Thickness4, Radius5, Radius6,
        Radius7, Radius8, Refractive_Index1, Refractive_Index2, Glass_Name1,
        Glass_Name2, Diameter, Lambda1, Lambda2, Separation),
        stop_surface(Position, entrance_Diameter).
```

Appendix 14: Rules and Searching Algorithms of Triplet Lenses

The below algorithm searches within the triplet lens and large aperture triplet lens set in order to find the appropriate lens for the desired application. This search strategy is based on heuristics.



Searching Algorithms of Triplet Lenses

The searching algorithms of triplet lenses are coded in the programming language Prolog as displayed below.

```
/*

This rule is used at three wavelengths in order to generate a plano-convex
triplet if the plano-convex conditions are satisfied.
The predicate planoconvextriplet returns the data of the triplet lens.
The triplet lens data is displayed according to the predicate out-
put_triplet_lens.
The stop_surface predicate gives the diameter and the position of the stop
surface.

*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        useplanoconvex(Optical_Power, Transversal_Magnification, L, Diameter),
        planoconvextriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
        Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        output_triplet_lens(Optical_Power, Thickness1, Thickness2, Thickness3, Radius2,
        -Radius1, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        stop_surface(Position, entrance_Diameter).

/*

This rule generates a bi-convex triplet lens used at three wavelengths.
The predicate wavelength tests if the triplet lens will be used at three wave-
lengths.
The predicate usebiconvex tests if the bi-convex use conditions are satisfied.
The predicate biconvextriplet returns the data of the bi-convex triplet lens.
The predicate output_triplet_lens displays the data of the triplet lens.
The predicate stop_surface returns the diameter and the position of the stop
surface.

*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        usebiconvex(Optical_Power, Transversal_Magnification, L, Diameter),
        biconvextriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
        Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        output_triplet_lens(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
        Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
```

```
        stop_surface(Position, entrance_Diameter).

/*

This rule generates an asymetric bi-convex triplet lens used at three wave-
lengths.
The predicate wavelength tests if the triplet lens will be used at three wave-
lengths.
The predicate useasymetricbiconvex tests if the asymetric bi-convex use
conditions are satisfied.
The predicate asymetricbiconvextriplet returns the data of the asymetric bi-
convex triplet lens.
The predicate output_triplet_lens displays the data of the triplet lens.
The predicate stop_surface returns the diameter and the position of the stop
surface.

*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        useasymetricbiconvex(Optical_Power, Transversal_Magnification, L, Diameter),
        asymetricbiconvextriplet(Optical_Power, Thickness1, Thickness2, Thickness3,
        Radius1, Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Re-
        fractive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        output_triplet_lens(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
        Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        stop_surface(Position, entrance_Diameter).

/*

This rule generates a plano-concave triplet lens used at three wavelengths.
The predicate wavelength tests if the triplet lens will be used at three wave-
lengths.
The predicate useplanoconcave tests if the plano-concave use conditions are
satisfied.
The predicate planoconcavetriplet returns the data of the plano-concave
triplet lens.
The predicate output_triplet_lens displays the data of the triplet lens.
The predicate stop_surface returns the diameter and the position of the stop
surface.

*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        useplanoconcave(Optical_Power, Transversal_Magnification, L, Diameter),
```

```
        planoconcavetriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
        Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        output_triplet_lens(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
        Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        stop_surface(Position, entrance_Diameter).

/*

This rule generates a bi-concave triplet lens used at three wavelengths.
The predicate wavelength tests if the triplet lens will be used at three wave-
lengths.
The predicate usebiconcave tests if the bi-concave use conditions are satis-
fied.
The predicate biconcavetriplet returns the data of the bi-concave triplet
lens.
The predicate output_triplet_lens displays the data of the triplet lens.
The predicate stop_surface returns the diameter and the position of the stop
surface.

*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        usebiconcave(Optical_Power, Transversal_Magnification, L, Diameter),
        biconcavetriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
        Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        output_triplet_lens(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
        Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        stop_surface(Position, entrance_Diameter).

/*

This rule generates an asymetric bi-concave triplet lens used at three wave-
lengths.
The predicate wavelength tests if the triplet lens will be used at three wave-
lengths.
The predicate useasymetricbiconcave tests if the asymetric bi-concave use
conditions are satisfied.
The predicate asymetricbiconcavetriplet returns the data of the asymetric
bi-concave triplet lens.
The predicate output_triplet_lens displays the data of the triplet lens.
The predicate stop_surface returns the diameter and the position of the stop
surface.
```

```
*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        useasymetricbiconcave(Optical_Power, Transversal_Magnification, L, Diameter),
        asymetricbiconcavetriplet(Optical_Power, Thickness1, Thickness2, Thickness3,
        Radius1, Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Re-
        fractive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        output_triplet_lens(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
        Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        stop_surface(Position, entrance_Diameter).

/*
```

This rule generates a concave-convex triplet lens used at three wavelengths.
The predicate wavelength tests if the triplet lens will be used at three wavelengths.
The predicate useconcaveconvex tests if the concave-convex use conditions are satisfied.
The predicate concaveconvextriplet returns the data of the concave-convex triplet lens.
The predicate output_triplet_lens displays the data of the triplet lens.
The predicate stop_surface returns the diameter and the position of the stop surface.

```
*/

lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        useconcaveconvex(Optical_Power, Transversal_Magnification, L, Diameter),
        concaveconvextriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Ra-
        dius1, Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        output_triplet_lens(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
        Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        stop_surface(Position, entrance_Diameter).

/*
```

This rule generates a convex-concave triplet lens used at three wavelengths.
The predicate wavelength tests if the triplet lens will be used at three wavelengths.
The predicate useconvexconcave tests if the convex-concave use conditions are satisfied.

The predicate convexconcavetriplet returns the data of the convex-concave triplet lens.
The predicate output_triplet_lens displays the data of the triplet lens.
The predicate stop_surface returns the diameter and the position of the stop surface.

*/

```
lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        useconvexconcave(Optical_Power, Transversal_Magnification, L, Diameter),
        convexconcavetriplet(Optical_Power, Thickness1, Thickness2, Thickness3, Ra-
        dius1, Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        output_triplet_lens(Optical_Power, Thickness1, Thickness2, Thickness3, Radius1,
        Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3,
        Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
        stop_surface(Position, entrance_Diameter).
```

/*

This rule generates a high aperture lens; corrected at three wavelengths.
This high aperture lens is made up of two triplet lenses mounted back to back.
The predicate usehighaperture tests if the high aperture conditions are satisfied.
The predicate highaperturetriplet returns the data of the two triplet lenses (high aperture lens).
The predicate output_triplet_lens displays the data of the two triplet lens.
The predicate stop_surface returns the diameter and the position of the stop surface.
In this case, the stop surface is located at the symetry plane of the high aperture lens i.e., between the two triplet lenses.

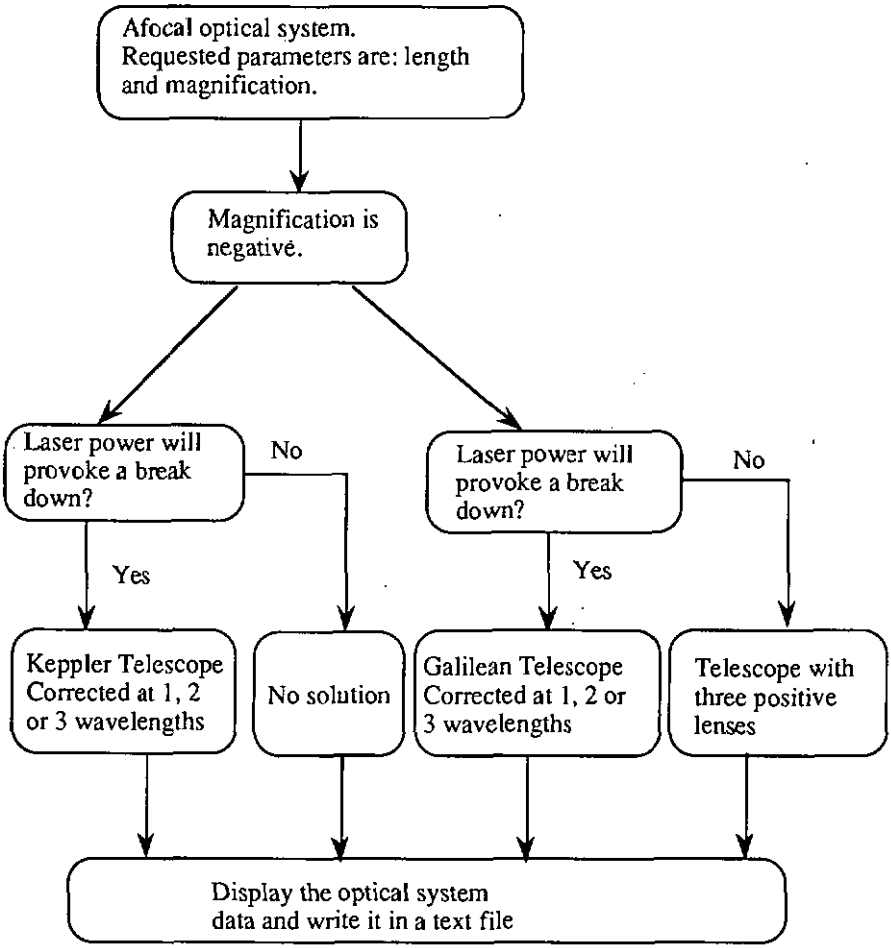
*/

```
lens(Optical_Power, Transversal_Magnification, L, L') :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        usehighaperture(Optical_Power, Transversal_Magnification, L, Diameter),
        outm(" Reversed triplet --> "),
        nl, Optical_Powerb is 1.0/So,
        Optical_Powera is 1.0/L,
        highaperturetriplet(Optical_Powera, Optical_Powerb, Thickness1a, Thickness2a,
        Thickness3a, Thickness1b, Thickness2b, Thickness3b, Radius1a, Radius2a, Ra-
        dius3a, Radius4a, Radius5a, Radius6a, Radius1b, Radius2b, Radius3b, Radius4b,
        Radius5b, Radius6b, Refractive_Index1, Refractive_Index2, Refractive_Index3,
        Glass_Name1, Glass_Name2, Glass_Name3, Diameter, Separation1, Separation2),
        output_triplet_lens(Optical_Powera, Thickness1a, Thickness2a, Thickness3a, Ra-
        dius1a, Radius2a, Radius3a, Radius4a, Radius5a, Radius6a, Refractive_Index1,
        Refractive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2,
```

Glass_Name3, Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
output_triplet_lens(Optical_Powerb, Thickness1b, Thickness2b, Thickness3b, Radius1b, Radius2b, Radius3b, Radius4b, Radius5b, Radius6b, Refractive_Index1, Refractive_Index2, Refractive_Index3, Glass_Name1, Glass_Name2, Glass_Name3, Diameter, Lambda1, Lambda2, Lambda3, Separation1, Separation2),
stop_surface(Position, entrance_Diameter).

Appendix 15: Searching Algorithms of Afocal Optical Systems

This algorithm is concemed with searching for afocal optical systems. This search strategy is based on heuristics which uses the context of use of the optical system.



Searching Algorithms of an Afocal Optical System

The searching algorithms of the afocal optical systems is coded in the programming language Prolog as displayed below:

```
start_afocalens(Angular_Magnification, Length) :-
        write("Optical_Power = 0.0 ==> afocal optical system"),
        nl,
        afocalens(Angular_Magnification, Length).

/*

The following heuristics states: If the angular magnification is negative
and there is a break down risk then it is not possible to design such as
telescope.

*/

afocalens(Angular_Magnification, Length) :-
        negative_angular_magnification(Angular_Magnification),
        breakdownrisk(Laserpower),
        nl,
        outm("such as system is not possible"),
        nl,
        outm("because the angular magnification is < 0"),
        nl,
        outm("and there is a break down").

/*

The following heuristics states: If the angular magnification is negative
and there is no break down risk then a Kepler telescope can be used.

*/

afocalens(Angular_Magnification, Length) :-
        negative_angular_magnification(Angular_Magnification),
        nobreakdownrisk(Laserpower),
        keplerafocalens(Angular_Magnification, Power1, Power2, Length).

/*

The following heuristics states: If the angular magnification is positive and
there is a break down risk then a Galilean telescope can be used.

*/

afocalens(Angular_Magnification, Length) :-
        positive_angular_magnification(Angular_Magnification),
        breakdownrisk(Laserpower),
        galileanafocalens(Angular_Magnification, Power1, Power2, Length).
```

/*

The following heuristics states: If the angular magnification is positive and there is no break down risk then a telescope composed of three positive optical lenses can be used.
This case can be elaborated later.

*/

```
afocalens(Angular_Magnification, Length) :-
        positive_angular_magnification(Angular_Magnification),
        nobreakdownrisk(Laserpower),
        threelementafocalens(Angular_Magnification, Power1, Power2, Power3, Length).
```

/*

The following heuristics states: If the angular magnification is positive and there is no break down risk and the telescope length is short then a Galilean telescope must be used.

*/

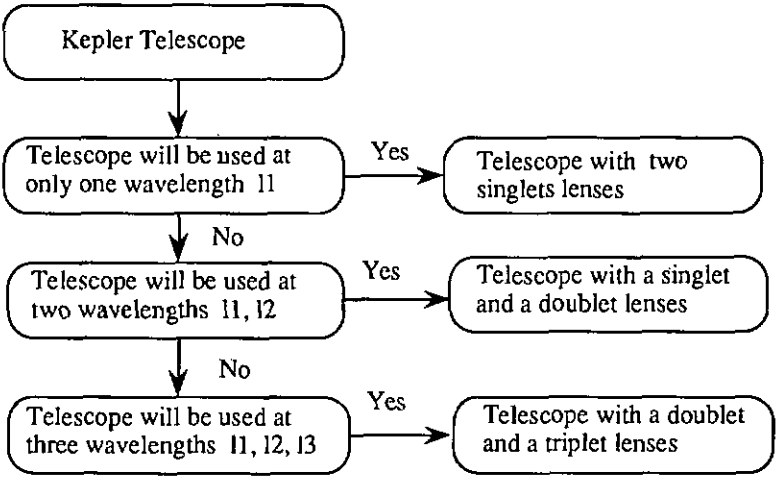
```
afocalens(Angular_Magnification, Length) :-
        positive_angular_magnification(Angular_Magnification),
        nobreakdownrisk(Laserpower),
        shortlength(Length,Angular_Magnification),
        galileanafocalens(Angular_Magnification, Power1, Power2, Length).
```

Appendix 16: Heuristics and Searching Algorithms concerning the Kepler Telescope

A Kepler telescope is formed by two optical systems:

- entrance optical system with a positive focal length
- exit optical system with a positive focal length

These entrance and exit optical systems can be formed by singlet, doublet or triplet lens according to the context of use of the telescope as displayed in the below diagram.



Searching Algorithms concerning the Kepler Telescope

The heuristics and searching algorithms of the Kepler telescope are coded in the programming language Prolog as displayed below.

```
/*

The following rule designs a Kepler telescope used at one wavelength.
A such as telescope is composed of two plano-convex singlet lenses
mounted face to face and separated with a distance, called Length.
The Length is the sum of the focal length of the two plano-convex lenses.

*/

keplerafocalens(Angular_Magnification, Power1, Power2, Length) :-
        wavelength([Lambda1]),
        longlength(Length, Angular_Magnification),
        write("--> kepler telescope ");
```

```
        Power1 is (1.0 - Angular_Magnification)/ Length,
        nl,
        lensindice([Glass_Nam1, Refractive_Index1]),
        angularmagnification(Angular_Magnification, Power1, Power2, Diameter1,
        Diameter2),
        nl,
        planoconvexlens(Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Diameter1),
        nl,
        outputlens(Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter1, Lambda1),
        lensindice([Glass_Nam2, Refractive_Index2]),
        planoconvexlens(Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Diameter2),
        nl,
        outputlens(Power2, Thickness2, Radius4, Radius3, Refractive_Index2,
        Glass_Nam2, Diameter2, Lambda1),
        outputafocalens(Angular_Magnification, Power1, Power2, Diameter1, Diameter2,
        Thickness1, Thickness2, Thickness3, Thickness4, Thickness5, Radius1, Radius2,
        Radius3, Radius4, Radius5, Radius6, Radius7, Radius8, Radius9, Radius10, Re-
        fractive_Index1, Refractive_Index2, Refractive_Index3, Refractive_Index4, Re-
        fractive_Index5, Glass_Nam1, Glass_Nam2, Glass_Nam3, Glass_Nam4,
        Glass_Nam5, Separation1, Separation2, Separation3, Lambda1, Lambda2,
        Lambda3).
/*
```

The following rule designs a Kepler telescope used at two wavelengths.
A such as telescope is composed of a plano-convex singlet lens and a plano-doublet lens mounted face to face and separated with a distance, called Length.
The Length is the sum of the focal length of the plano-convex singlet and the plano-convex doublet lens.

```
*/
keplerafocalens(Angular_Magnification, Power1, Power2, Length) :-
        wavelength([Lambda1, Lambda2]),
        write("--> kepler telescope "),
        longlength(Length, Angular_Magnification),
        Power1 is (1.0 - Angular_Magnification)/Length,
        nl,
        angularmagnification(Angular_Magnification, Power1, Power2, Diameter1,
        Diameter2),
        nl,
        lensindice([Glass_Nam1, Refractive_Index1]),
        planoconvexlens(Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Diameter1),
        nl,
        outputlens(Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter1, Lambda1),
        planoconvexdoublet(Power2, Thickness2, Thickness3, Radius3, Radius4, Ra-
        dius5, Radius6, Refractive_Index2, Refractive_Index3, Glass_Nam2,
        Glass_Nam3, Diameter2, Lambda1, Lambda2, Separation2),
        output_doublet_lens(Power2, Thickness2, Thickness3, Radius3, Radius4, Ra-
        dius5, Radius6, Refractive_Index2, Refractive_Index3, Glass_Nam2,
        Glass_Nam3, Diameter2, Lambda1, Lambda2, Separation2),
```

```
        outputafocalens(Angular_Magnification, Power1, Power2, Diameter1, Diameter2,
        Thickness1, Thickness2, Thickness3, Thickness4, Thickness5, Radius1, Radius2,
        Radius3, Radius4, Radius5, Radius6, Radius7, Radius8, Radius9, Radius10, Re-
        fractive_Index1, Refractive_Index2, Refractive_Index3, Refractive_Index4, Re-
        fractive_Index5, Glass_Nam1, Glass_Nam2, Glass_Nam3, Glass_Nam4,
        Glass_Nam5, Separation1, Separation2, Separation3, Lambda1, Lambda2,
        Lambda3).

/*
```

The following rule designs a Kepler telescope used at two wavelengths and with a short length.
A such as telescope is composed of a bi-convex singlet lens and a bi-doublet lens mounted face to face and separated with a distance, called Length.
The Length is the sum of the focal length of the bi-convex singlet and the bi-convex doublet lens.

```
*/

keplerafocalens(Angular_Magnification, Power1, Power2, Length) :-
        wavelength([Lambda1, Lambda2]),
        write("--> kepller telescope "),
        shortlength(Length, Angular_Magnification),
        Power1 is (1.0 - Angular_Magnification)/Length,
        nl,
        angularmagnification(Angular_Magnification, Power1, Power2, Diameter1,
        Diameter2),
        nl,
        lensindice([Glass_Nam1, Refractive_Index1]),
        biconvexlens(Power1, Thickness1, Radius1, Radius2, Refractive_Index1, Diame-
        ter1),
        nl,
        outputlens(Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter1, Lambda1),
        biconvexdoublet(Power2, Thickness2, Thickness3, Radius3, Radius4, Radius5,
        Radius6, Refractive_Index2, Refractive_Index3, Glass_Nam2, Glass_Nam3,
        Diameter2, Lambda1, Lambda2, Separation2),
        output_doublet_lens(Power2, Thickness2, Thickness3, Radius3, Radius4, Ra-
        dius5, Radius6, Refractive_Index2, Refractive_Index3, Glass_Nam2,
        Glass_Nam3, Diameter2, Lambda1, Lambda2, Separation2),
        outputafocalens(Angular_Magnification, Power1, Power2, Diameter1, Diameter2,
        Thickness1, Thickness2, Thickness3, Thickness4, Thickness5, Radius1, Radius2,
        Radius3, Radius4, Radius5, Radius6, Radius7, Radius8, Radius9, Radius10, Re-
        fractive_Index1, Refractive_Index2, Refractive_Index3, Refractive_Index4, Re-
        fractive_Index5, Glass_Nam1, Glass_Nam2, Glass_Nam3, Glass_Nam4,
        Glass_Nam5, Separation1, Separation2, Separation3, Lambda1, Lambda2,
        Lambda3).

/*
```

The following rule designs a Kepler telescope used at three wavelengths and with a short length.

A such as telescope is composed of a bi-convex singlet lens and a bi-convex triplet lens separated with a distance, called Length.
The Length is the sum of the focal length of the bi-convex triplet and the bi-convex singlet lens.

*/

```
keplerafocalens(Angular_Magnification, Power1, Power2, Length) :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        write("--> kepller telescope "),
        shortlength(Length, Angular_Magnification),
        Power1 is (1.0 - Angular_Magnification)/Length,
        nl,
        angularmagnification(Angular_Magnification, Power1, Power2, Diameter1,
        Diameter2),
        nl,
        lensindice([Glass_Nam1, Refractive_Index1]),
        biconvexlens(Power1, Thickness1, Radius1, Radius2, Refractive_Index1, Diame-
        ter1),
        nl,
        outputlens(Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter1, Lambda1),
        biconvextriplet(Power2, Thickness2, Thickness3, Radius5, Radius6, Radius7,
        Radius8, Radius9, Radius10, Refractive_Index3, Refractive_Index4, Refrac-
        tive_Index5, Glass_Nam3, Glass_Nam4, Glass_Nam5, Diameter2, Lambda1,
        Lambda2, Lambda3, Separation2, Separation3),
        output_triplet_lens(Power2, Thickness2, Thickness3, Radius5, Radius6, Radius7,
        Radius8, Radius9, Radius10, Refractive_Index3, Refractive_Index4, Refrac-
        tive_Index5, Glass_Nam3, Glass_Nam4, Glass_Nam5, Diameter2, Lambda1,
        Lambda2, Lambda3, Separation2, Separation3),
        outputafocalens(Angular_Magnification, Power1, Power2, Diameter1, Diameter2,
        Thickness1, Thickness2, Thickness3, Thickness4, Thickness5, Radius1, Radius2,
        Radius3, Radius4, Radius5, Radius6, Radius7, Radius8, Radius9, Radius10, Re-
        fractive_Index1, Refractive_Index2, Refractive_Index3, Refractive_Index4, Re-
        fractive_Index5, Glass_Nam1, Glass_Nam2, Glass_Nam3, Glass_Nam4,
        Glass_Nam5, Separation1, Separation2, Separation3, Lambda1, Lambda2,
        Lambda3).
/*
```

The following rule designs a Kepler telescope used at three wavelengths and with a long length.
A such as telescope is composed of a plano-convex singlet lens and a plano-convex triplet lens mounted face to face and separated with a distance, called Length.
The Length is the sum of the focal length of the plano-convex doublet and the plano-convex triplet lens.

*/

```
keplerafocalens(Angular_Magnification, Power1, Power2, Length) :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        write("--> kepler telescope "),
        longlength(Length, Angular_Magnification),
        Power1 is (1.0 - Angular_Magnification)/Length,
```

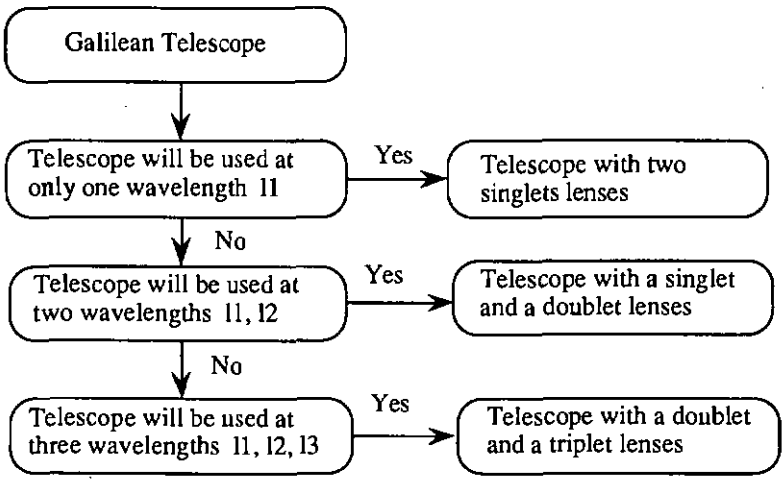
```
nl,
angularmagnification(Angular_Magnification, Power1, Power2, Diameter1,
Diameter2),
nl,
planoconvexdoublet(Power1, Thickness1, Thickness2, Radius1, Radius2, Ra-
dius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Nam1,
Glass_Nam2, Diameter1, Lambda1, Lambda2, Separation1),
output_doublet_lens(Power1, Thickness1, Thickness2, Radius1, Radius2, Ra-
dius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Nam1,
Glass_Nam2, Diameter1, Lambda1, Lambda2, Separation1),
planoconvextriplet(Power2, Thickness3, Thickness4, Thickness5, Radius5, Ra-
dius6, Radius7, Radius8, Radius9, Radius10, Refractive_Index3, Refrac-
tive_Index4, Refractive_Index5, Glass_Nam3, Glass_Nam4, Glass_Nam5, Dia-
meter2, Lambda1, Lambda2, Lambda3, Separation2, Separation3),
outpu_triplet_lens(Power2, Thickness3, Thickness4, Thickness5, Radius1,
Radius2, Radius3, Radius4, Radius5, Radius6, Refractive_Index3,
Refractive_Index4, Refractive_Index5, Glass_Nam3, Glass_Nam4, Glass_Nam5,
Diameter2, Lambda1, Lambda2, Lambda3, Separation2, Separation3),
outputafocalens(Angular_Magnification, Power1, Power2, Diameter1, Diameter2,
Thickness1, Thickness2, Thickness3, Thickness4, Thickness5, Radius1, Radius2,
Radius3, Radius4, Radius5, Radius6, Radius7, Radius8, Radius9, Radius10, Re-
fractive_Index1, Refractive_Index2, Refractive_Index3, Refractive_Index4, Re-
fractive_Index5, Glass_Nam1, Glass_Nam2, Glass_Nam3, Glass_Nam4,
Glass_Nam5, Separation1, Separation2, Separation3, Lambda1, Lambda2,
Lambda3).
```

Appendix 17: Heuristics and Searching Algorithms concerning the Galilean Telescope

A Galilean telescope is formed by two optical systems:

- entrance optical system with a negative focal length
- exit optical system with a positive focal length

These entrance and exit optical systems can be formed by a singlet, doublet or triplet lens according to the context of use of the telescope as displayed in the below diagram.



Searching Algorithms of a Galilean Telescope

The rules conerning the Galilean telescope are coded in the programming language Prolog as displayed below.

```
/*

The following rule designs a Galilean telescope used at one wavelength.
A such as telescope is composed of a plano-concave singlet lens and a
plano-convex singlet lens mounted face to face and separated with a
distance, called Length.
The Length is the difference of the focal length of the plano-concave
singlet and the plano-convex singlet lens.

*/

galileanafocalens(Angular_Magnification, Power1, Power2, Length) :-
        wavelength([Lambda1]),
```

```
        write("--> Galilean telescope "),
        Power1 is (1.0 - Angular_Magnification)/Length,
        nl,
        angularmagnification(Angular_Magnification, Power1, Power2, Diameter1,
        Diameter2),
        nl,
        lensindice([Glass_Nam1, Refractive_Index1]),
        planoconcavelens(Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Diameter1),
        nl,
        outputlens(Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter1, Lambda1),
        lensindice([Glass_Nam2, Refractive_Index2]),
        planoconvexlens(Power2, Thickness2, Radius4, Radius3, Refractive_Index2,
        Diameter2),
        nl,
        outputlens(Power2, Thickness2, Radius3, Radius4, Refractive_Index2,
        Glass_Nam2, Diameter2, Lambda1),
        outputafocalens(Angular_Magnification, Power1, Power2, Diameter1, Diameter2,
        Thickness1, Thickness2, Thickness3, Thickness4, Thickness5, Radius1, Radius2,
        Radius3, Radius4, Radius5, Radius6, Radius7, Radius8, Radius9, Radius10, Re-
        fractive_Index1, Refractive_Index2, Refractive_Index3, Refractive_Index4, Re-
        fractive_Index5, Glass_Nam1, Glass_Nam2, Glass_Nam3, Glass_Nam4,
        Glass_Nam5, Separation1, Separation2, Separation3, Lambda1, Lambda2,
        Lambda3).

/*
```

The following rule designs a Galilean telescope used at one wavelength.
A such as telescope is composed of a plano-concave singlet lens and a plano-convex doublet lens mounted face to face and separated with a distance, called Length.
The Length is the sum of the focal length of the plano-concave singlet and the plano-convex doublet lens.

```
*/

galileanafocalens(Angular_Magnification, Power1, Power2, Length) :-
        wavelength([Lambda1, Lambda2]),
        write("--> Galilean telescope "),
        Power1 is (1.0 - Angular_Magnification)/Length,
        nl,
        angularmagnification(Angular_Magnification, Power1, Power2, Diameter1,
        Diameter2),
        nl,
        lensindice([Glass_Nam1, Refractive_Index1]),
        planoconcavelens(Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Diameter1),
        nl,
        outputlens(Power1, Thickness1, Radius1, Radius2, Refractive_Index1,
        Glass_Nam1, Diameter1, Lambda1),
        planoconvexdoublet(Power2, Thickness2, Thickness3, Radius3, Radius4, Ra-
        dius5, Radius6, Refractive_Index2, Refractive_Index3, Glass_Nam2,
        Glass_Nam3, Diameter2, Lambda1, Lambda2, Separation2),
```

```
        output_doublet_lens(Power2, Thickness2, Thickness3, Radius3, Radius4, Ra-
        dius5, Radius6, Refractive_Index2, Refractive_Index3, Glass_Nam2,
        Glass_Nam3, Diameter2, Lambda1, Lambda2, Separation1),
        outputafocalens(Angular_Magnification, Power1, Power2, Diameter1, Diameter2,
        Thickness1, Thickness2, Thickness3, Thickness4, Thickness5, Radius1, Radius2,
        Radius3, Radius4, Radius5, Radius6, Radius7, Radius8, Radius9, Radius10, Re-
        fractive_Index1, Refractive_Index2, Refractive_Index3, Refractive_Index4, Re-
        fractive_Index5, Glass_Nam1, Glass_Nam2, Glass_Nam3, Glass_Nam4,
        Glass_Nam5, Separation1, Separation2, Separation3, Lambda1, Lambda2,
        Lambda3).
/*
```

The following rule designs a Galilean telescope used at three wavelengths.
A such as telescope is composed of a plano-convex doublet lens and a plano-convex triplet lens mounted face to face and separated with a distance, called Length.
The Length is the sum of the focal length of the plano-convex triplet and the plano-convex doublet lens.

```
*/

galileanafocalens(Angular_Magnification, Power1, Power2, Length) :-
        wavelength([Lambda1, Lambda2, Lambda3]),
        write("--> Galilean telescope "),
        Power1 is (1.0 - Angular_Magnification)/Length,
        nl,
        angularmagnification(Angular_Magnification, Power1, Power2, Diameter1,
        Diameter2),
        nl,
        planoconcavedoublet(Power1, Thickness1, Thickness2, Radius1, Radius2, Ra-
        dius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Nam1,
        Glass_Nam2, Diameter1, Lambda1, Lambda2, Separation),
        output_doublet_lens(Power1, Thickness1, Thickness2, Radius1, Radius2, Ra-
        dius3, Radius4, Refractive_Index1, Refractive_Index2, Glass_Nam1,
        Glass_Nam2, Diameter1, Lambda1, Lambda2, Separation1),
        planoconvextriplet(Power2, Thickness3, Thickness4, Thickness5, Radius5, Ra-
        dius6, Radius7, Radius8, Radius9, Radius10, Refractive_Index3, Refrac-
        tive_Index4, Refractive_Index5, Glass_Nam3, Glass_Nam4, Glass_Nam5, Dia-
        meter2, Lambda1, Lambda2, Lambda3, Separation2, Separation3),
        output_triplet_lens(Power2, Thickness3, Thickness4, Thickness5, Radius5, Ra-
        dius6, Radius7, Radius8, Radius9, Radius10, Indice3, Refractive_Index4, Re-
        fractive_Index5, Glass_Nam3, Glass_Nam4, Glass_Nam5, Diameter2, Lambda1,
        Lambda2, Lambda3, Separation1, Separation2),
        outputafocalens(Angular_Magnification, Power1, Power2, Diameter1, Diameter2,
        Thickness1, Thickness2, Thickness3, Thickness4, Thickness5, Radius1, Radius2,
        Radius3, Radius4, Radius5, Radius6, Radius7, Radius8, Radius9, Radius10, Re-
        fractive_Index1, Refractive_Index2, Refractive_Index3, Refractive_Index4, Re-
        fractive_Index5, Glass_Nam1, Glass_Nam2, Glass_Nam3, Glass_Nam4,
        Glass_Nam5, Separation1, Separation2, Separation3, Lambda1, Lambda2,
        Lambda3).

/*
```

The following rule designs a telescope composed of three positive lenses. Such as telescope needs a special heuristics that are not availible at this moment.

*/

```
threelementafocalens(Angular_Magnification, Power1, Power2, Power3, Length) :-
        outm(" We didn't consider such a telescope at this moment").
```

/*

The next two rules calculate the diameter of the sortance block of telescopic optical system as well as the optical power of the entrance and sortance block.

*/

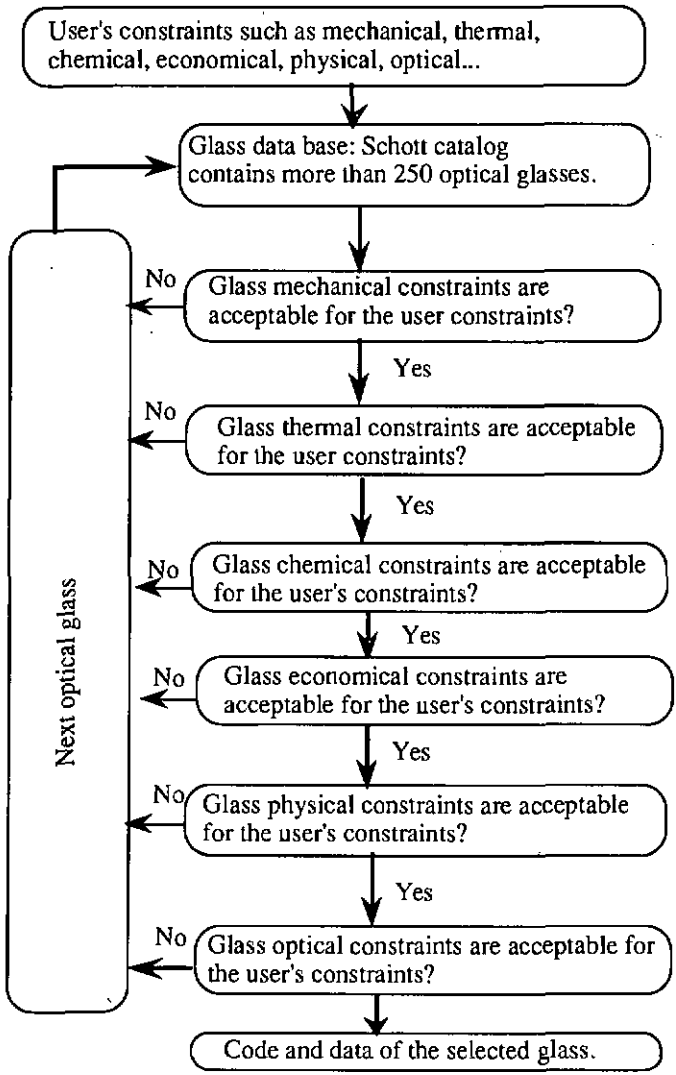
```
angularmagnification(Angular_Magnification, Optical_Power1, Optical_Power2,
        Diameter1, Diameter2) :-
        Angular_Magnification < 0.0,
        diameter(Diameter1),
        Diameter2 is -Angular_Magnification*Diameter1,
        Optical_Power2 is - Optical_Power1/Angular_Magnification.

angularmagnification(Angular_Magnification, Optical_Power1, Optical_Power2,
        Diameter1, Diameter2) :-
        Angular_Magnification > 0.0,
        diameter(Diameter1),
        Diameter2 is Angular_Magnification*Diameter1,
        Optical_Power2 is abs(Optical_Power1/Angular_Magnification).
```

Appendix 18: Rules, Heuristics and Searching Algorithms for the Selection of the Optical Glass

The optical glass is one of the most important parameters during the calculation of the optical system. The search strategy of the optical glass is made according to a depth first searching.

The glass data base contains more than 250 different optical glasses.
The first element in the list of optical glass is taken (depth first searching) then it is tested according to the user glass constraints. The constraint list is reviewed in depth searching too.

This algorithm is repeated until the end of the list of optical glasses is reached.
This search strategy is used at the other section of this chapter.
This searching is also based on heuristics.



Searching algorithms for selection of an optical glass

The searching algorithms for selection of an optical glass is coded in the programming language Prolog as displayed below.

```
/*

The following rule returns the dispersion coefficients of an optical glass.

*/

dispersion_formula_coefficients(Optical_Glass, A0, A1, A2, A3, A4, A5) :-
        coef1_disp(Optical_Glass, A0, A1, A2),
        coef2_disp(Optical_Glass, A3, A4, A5).

/*

The following rule calculates the refractive index of an optical glass according to his dispersion coefficients.

*/

refractive_index_calculation(Optical_Glass, Lambda, Indice_de_refraction) :-
        dispersion_formula_coefficients(Optical_Glass, A0, A1, A2, A3, A4, A5),
        power_of(Lambda, 2.0, Y1),
        power_of(Lambda, -2.0, Y_2),
        power_of(Lambda, -4.0, Y_4),
        power_of(Lambda, -6.0, Y_6),
        power_of(Lambda, -8.0, Y_8),
        Indice_de_refraction is sqrt(A0 + A1*Y1 + A2*Y_2 + A3*Y_4 + A4*Y_6 +
        A5*Y_8).

% Thermical conductivity varies from 0(highst value) to 1.4(lowst value)

thermical_conductivity(Optical_Glass, X) :-
        temperature(T), free_variable(X, 0.0),
        conversion(X, X1), X1 =< 50.0/abs(T).

presences_of_bubbles(Optical_Glass, X) :-
        degre_de_resistance_chimique(Degre_Chimik),
        conversion(X, X1), X1 =< Degre_Chimik.

climatic_resistance(Optical_Glass, X) :-
         degre_de_resistance_chimique(Degre_Chimik),
        conversion(X, X1), X1 =< Degre_Chimik +1.0.

stain_resistance(Optical_Glass, X) :-
        degre_de_resistance_chimique(Degre_Chimik),
        conversion(X, X1), X1 =< Degre_Chimik.

alkaline_resistance(Optical_Glass, X) :-
        free_variable(X, 4.0),
        alcalis_milieu(Alcalis),
        conversion(X, X1), Y1 is 15.0 - X1, Y1 >= Alcalis.

acid_resistance(Optical_Glass, X) :-
```

```
        free_variable(X, 5.0),
        acidite_milieu(Acidite),
        conversion(X, X1), X1 =< Acidite.

density(Optical_Glass, X) :-
        densite_verre(Densite), conversion(X, X1), X1 =< Densite.

elasticity_coefficient(Optical_Glass, X) :-
        elasticite_verre(Elasticite),
        conversion(X, X1),
        X1 =< Elasticite.

poisson_coefficient(Optical_Glass, X) :-
        conversion(X, X1),
        X1 > 0.0.

knoop_hardness(Optical_Glass, X) :-
        durete_verre(Hardness),
        conversion(X, X1), X1 >= Hardness .

/*

The following rule selects an economical glass, if the variable Price is 0.

*/

economical_optical_glass(Optical_Glass) :-
        economical(Price), Price > 0.0,
        economical_glasse(Optical_Glass , Nd , Nued , Densite , A0 , A1 , A2 , A3 , A4 ,
        A5).
/*

The following rule defines the physical properties of an optical glass.

*/

physical_properties(Optical_Glass, Alpha1, Alpha2, Tg, T, Cp, Conduthermique, Rho,
        Emo, Mue, Hk, B, Cr, Fr, Sr, Ar) :-
        prop1_physik(Optical_Glass, Alpha1, Alpha2, Tg, T, Cp, Conduthermique),
        prop2_physik(Optical_Glass, Rho, Emo, Mue, Hk, B, Cr, Fr, Sr, Ar).


/*

Thermical_dilatation varies from 4*10e-6 (highst value) to 16*10e-6
(lowst value)

*/

thermical_dilatation_moins30_plus70(Optical_Glass, X) :-
        temperature(T),
        conversion(X, X1),
        X1 =< 640.0/T.
```

```
/*
```

Thermical dilatation varies from +20 to +300, X varies de 4 (highst value) to 16*10pe-6(lowst value)

```
*/

thermical_dilatation_plus20_plus300(Optical_Glass, X):-
        temperature(T), T > 40.0,
        conversion(X, X1),
        X1 < 3000.0/abs(T).
```

% Melts temperature must be > 4* work temperature.

```
melts_temperature(Optical_Glass, X) :-
        conversion(X, X1), temperature(T), X1>3.2*abs(T).
```

% Specific_heat varies fro 0.4 (highst value) to 1.6 (lowst value)

```
specific_heat(Optical_Glass, X) :-
        temperature(T), free_variable(X, 0.4),
        conversion(X, X1), X1 =< 100.0/abs(T).

krone(Nd, Nued) :-
         Nd < 1.6, Nued > 51.0.

flinte(Nd, Nued) :-
         Nd>1.6, Nued < 30.0.

/*
```

The two following rules defines the climatical properties of an optical glass.

```
*/

climatical_properties(Optical_Glass, Alpha1, Alpha2, Tg, T, Cp, Conduct_thermique) :-
        prop1_physik(Optical_Glass, Alpha1, Alpha2, Tg, T, Cp, Conduct_thermique),
        thermical_dilatation_moins30_plus70(Optical_Glass, Alpha1),
        temperature_of_transformation(Optical_Glass, Tg),
        specific_heat(Optical_Glass, Cp),
        thermical_conductivity(Optical_Glass, Conduct_thermique).

climatical_properties(Optical_Glass, Alpha1, Alpha2, Tg, T, Cp, Conduct_thermique) :-
        prop1_physik(Optical_Glass, Alpha1, Alpha2, Tg, T, Cp, Conduct_thermique),
        thermical_dilatation_plus20_plus300(Optical_Glass, Alpha2),
        temperaturee_of_transformation(Optical_Glass, Tg),
        specific_heat(Optical_Glass, Cp),
        thermical_conductivity(Optical_Glass, Conduct_thermique).

/*
```

The following rule defines the chemical properties of an optical glass.

```
*/

chemical_properties(Optical_Glass, B, Cr, Fr, Sr, Ar) :-
        prop2_physik(Optical_Glass, Rho, Emo, Mue, Hk, B, Cr, Fr, Sr, Ar),
        presences_of_bubbles(Optical_Glass, B),
        climatic_resistance(Optical_Glass, Cr),
        stain_resistance(Optical_Glass, Fr),
        acid_resistance(Optical_Glass, Sr),
        alkaline_resistance(Optical_Glass, Ar).

/*
```

The following rule defines the mechanical properties of an optical glass.

```
*/

mechanical_properties(Optical_Glass, Rho, Emo, Mue, Hk) :-
        prop2_physik(Optical_Glass, Rho, Emo, Mue, Hk, B, Cr, Fr, Sr, Ar),
        density(Optical_Glass, Rho),
        elasticity_coefficient(Optical_Glass, Emo),
        poisson_coefficient(Optical_Glass, Mue),
        knoop_hardness(Optical_Glass, Hk).

/*
```

The following rule returns the name, Glass_Nam1, of an optical glass and its refractive index Nd.
Refering to the Glass_Nam1, one get all information from the facts base about this optical glass.
This selected optical glass has to satisfy the economical, climatical, mechanical and chemical conditions.

```
*/

lensindice([Glass_Nam1, Nd]) :-
        entete_verre(Glass_Nam1, Glass_Nam2, Nd, Nued, Nf_Nc, Ne1, Nuee1, Nf-
        prim_moins_Ncprim ),
        economical_optical_glass(Glass_Nam1),
        climatical_properties(Glass_Nam1, Alpha1, Alpha2, Tg, T, Cp, Con-
        duct_thermique),
        chemical_properties(Glass_Nam1, B, Cr, Fr, Sr, Ar),
        mechanical_properties(Glass_Nam1, Rho, Emo, Mue, Hk) .

/*
```

The following rule returns two optical glasses referred with the names Glass_Nam1 and Glass_Nam2.
It is used in the case of the doublet lens. One of the optical glasses is Krone type and the other one is flint type.

```
*/

lensindice2([Glass_Nam1, Glass_Nam2, Refractive_Index1, Refractive_Index2, Nuee1,
        Nuee2]) :-
        entete_verre(Glass_Nam1, Optical_Glass, Nd, Nued, Nf_Nc, Ne1, Nuee1, Nf-
        prim_moins_Ncprim ),
        economical_optical_glass(Glass_Nam1),
        flinte(Ne1, Nuee1),
        Refractive_Index1 is Ne1,
        chemical_properties(Glass_Nam1, B, Cr, Fr, Sr, Ar),
        climatical_properties(Glass_Nam1, Alpha1, Alpha2, Tg, T, Cp, Conduct_ ther-
        mique),
        mechanical_properties(Glass_Nam1, Rho, Emo, Mue, Hk),
        entete_verre(Glass_Nam2, Optical_Glass111, Nd1, Nued1, Nf1_Nc1, Ne2,
        Nuee2, Nfprim_moins_Ncprim1),
        economical_optical_glass(Glass_Nam2),
        Glass_Nam1 =\= Glass_Nam2,
        krone(Ne2, Nuee2),
        Refractive_Index2 is Ne2,
        chemical_properties(Glass_Nam2, B2, Cr2, Fr2, Sr2, Ar2),
        climatical_properties(Glass_Nam2, Alpha11, Alpha21, Tg1, T1, Cp1, Con-
        duct_thermique1),
        mechanical_properties(Glass_Nam2, Rho2, Emo2, Mue2, Hk2).

/*
```

The following rule returns three different optical glasses referred with the names Glass_Nam1, Glass_Nam2 and Glass_Nam3. It is used in the case of the triplet lens.

```
*/

lensindice3([Glass_Nam1, Glass_Nam2, Glass_Nam3, Refractive_Index1, Refrac-
        tive_Index2, Refractive_Index3, TetagC1, TetagC2, TetagC3, Nued1, Nued2,
        Nued3]) :-
        entete_verre(Glass_Nam1, Optical_Glass1, Nd1, Nued1, Nf_Nc1, Ne1, Nuee1,
        Nfprim_moins_Ncprim1 ),
        economical_optical_glass(Glass_Nam1),
        indice2_refract(Glass_Nam1, NC, Nc', N632, N589 , N587 , N546 ),
        indice3_refract(Glass_Nam1 , Nf, Nf', Ng , Nh, Ni),
        Refractive_Index1 is N587, TetagC1 is (Ng - NC)/(Nf_Nc1),
        krone(Refractive_Index1, Nued1),
        chemical_properties(Glass_Nam1, B, Cr, Fr, Sr, Ar),
        climatical_properties(Glass_Nam1, Alpha1, Alpha2, Tg, T, Cp, Con-
        duct_thermique),
        mechanical_properties(Glass_Nam1, Rho1, Emo1, Mue1, Hk1),
        found_2_optical_glass(Glass_Nam1, Glass_Nam2, Glass_Nam3, Refrac-
        tive_Index1, Refractive_Index2, Refractive_Index3, TetagC1, TetagC2, TetagC3,
        Nued1, Nued2, Nued3).

/*
```

In this case it is not necessary that the second optical glass has a high mechanical resistance!!!

```
*/

found_2_optical_glass(Glass_Nam1, Glass_Nam2, Glass_Nam3, Refractive_Index1, Refractive_Index2, Refractive_Index3, TetagC1, TetagC2, TetagC3, Nued1, Nued2, Nued3) :-
        entete_verre(Glass_Nam2, Optical_Glass1, Nd1, Nued2, Nf_Nc, Nc1, Nuee1, Nfprim_moins_Ncprim1 ),
        economical_optical_glass(Glass_Nam2),
        Glass_Nam2 =\= Glass_Nam1,
        indice2_refract(Glass_Nam2, NC, Nc', N632, N589 , N587 , N546 ),
        indice3_refract(Glass_Nam2 , Nf, Nf', Ng , Nh, Ni),
        Refractive_Index2 is N587, TetagC2 is (Ng - NC)/(Nf_Nc),
        Glass_Nam1=\=Glass_Nam2,
        flinte(Refractive_Index2, Nued2),
        chemical_properties(Glass_Nam2, B, Cr, Fr, Sr, Ar),
        climatical_properties(Glass_Nam2, Alpha1, Alpha2, Tg, T, Cp, Conduct_thermique),
        found_3_optical_glass(Glass_Nam1, Glass_Nam2, Glass_Nam3, Refractive_Index1, Refractive_Index2, Refractive_Index3, TetagC1, TetagC2, TetagC3, Nued1, Nued2, Nued3).


found_3_optical_glass(Glass_Nam1, Glass_Nam2, Glass_Nam3, Refractive_Index1, Refractive_Index2, Refractive_Index3, TetagC1, TetagC2, TetagC3, Nued1, Nued2, Nued3) :-
        entete_verre(Glass_Nam3, Optical_Glass1, Nd1, Nued3, Nf_Nc, Nc1, Nuee1, Nfprim_moins_Ncprim1 ),
        economical_optical_glass(Glass_Nam3),
        Glass_Nam3=\=Glass_Nam2,
        Glass_Nam3=\=Glass_Nam1,
        indice2_refract(Glass_Nam3, NC, Nc', N632, N589 , N587 , N546 ),
        indice3_refract(Glass_Nam3 , Nf, Nf', Ng , Nh, Ni),
        Refractive_Index3 is N587,
        flinte(Refractive_Index3, Nued3) ,
        chemical_properties(Glass_Nam3, B, Cr, Fr, Sr, Ar),
        climatical_properties(Glass_Nam3, Alpha1, Alpha2, Tg, T, Cp, Conduct_thermique),
        mechanical_properties(Glass_Nam3, Rho3, Emo3, Mue3, Hk3),
        TetagC3 is (Ng - NC)/(Nf_Nc).


power_of(X, N, Y) :-
        Y is exp(N*ln(X)).

conversion(X, Y) :-
        integer(X), Y is float(X).

conversion(X, Y) :-
        real(X), Y is X.
```

Appendix 19: Fact Base of the Physical Properties of the Optical Glass

/*

This base of facts contains the physical(thermal) properties of the optical glasses.

The parameters are:

Alpha1:	Thermal coefficient in the domain: -30 degree To 70 Degree.
Alpha2:	Thermal coefficient in the domain: 20 degree To 300 Degree.
Tg:	Melts Temperature in degree.
Cp:	Specific heat coefficient.
Lambda:	Thermal conductivity.

*/

%prop1physik(Glass_Name,Alpha1, Alpha2, Tg, T, Cp, Lambda).

```
prop1_physik(bafn10,    6.8,  7.9,   630,   745,   0.595,   0.798 ).
prop1_physik(bafn11,    6.8,  7.9,   631,   750,   ,   ,         ).
prop1_physik(bafn6,     7.8,  8.5,   549,   ,   ,   ,   ,        ).
prop1_physik(baf12,     6.8,  7.9,   596,   ,   ,   ,   ,        ).
prop1_physik(baf13,     7.4,  8.5,   606,   719,   ,   ,         ).
prop1_physik(baf3,      7.8,  8.8,   533,   720,   ,   ,         ).
prop1_physik(baf4,      7.9,  8.8,   521,   694,   0.557,   0.766 ).
prop1_physik(baf5,      7.0,  7.8,   604,   765,   ,   ,         ).
prop1_physik(baf50,     8.3,  9.6,   546,   682,   ,   ,         ).
prop1_physik(baf51,     8.4,  9.6,   556,   693,   ,   ,         ).
prop1_physik(baf52,     8.4,  9.4,   546,   697,   0.636,   0.884 ).
prop1_physik(baf53,     6.5,  7.4,   630,   ,   ,0.561,0.857     ).
prop1_physik(baf54,     6.2,  7.3,   630,   ,   ,0.574,0.872     ).
prop1_physik(baf8,      7.0,  7.9,   589,   737,   ,   ,         ).
prop1_physik(baf9,      6.5,  7.5,   610,   740,   ,   ,         ).
prop1_physik(bak1,      7.6,  8.7,   602,   746,   0.687,   0.795 ).
prop1_physik(bak2,      8.0,  9.1,   562,   727,   ,   ,         ).
prop1_physik(bak4,      7.0,  8.1,   552,   707,   0.678,   0.856 ).
prop1_physik(bak5,      7.8,  9.0,   580,   737,   ,   ,         ).
prop1_physik(bak50,     3.7,  4.6,   629,   820,   0.758,   1.044 ).
prop1_physik(bak6,      7.3,  8.5,   578,   716,   ,   ,         ).
prop1_physik(balf3,     8.1,  9.4,   543,   714,   ,   ,         ).
prop1_physik(balf4,     6.4,  7.5,   569,   731,   0.670,   0.827 ).
prop1_physik(balf5,     8.1,  9.3,   529,   724,   ,   ,         ).
prop1_physik(balf50,    8.3,  9.6,   555,   710,   0.624,   0.928 ).
prop1_physik(balf51,    8.1,  9.4,   527,   700,   0.615,   0.934 ).
prop1_physik(balf6,     6.7,  7.9,   577,   724,   0.502,   0.779 ).
prop1_physik(balf8,     8.3,  9.3,   513,   701,   ,   ,         ).
prop1_physik(balkn3,    7.9,  9.0,   562,   738,   0.749,   1.029 ).
prop1_physik(balk1,     9.1, 10.4,   509,   681,   0.766,   1.043 ).
prop1_physik(basf1,     8.5,  9.5,   493,   658,   0.553,   0.741 ).
prop1_physik(basf10,    8.6,  9.8,   484,   635,   ,   ,         ).
prop1_physik(basf12,    7.9,  9.0,   532,   656,   ,   ,         ).
```

```
prop1_physik(basf13,    7.1,  8.2,   584,        ,      ,      ,      ).
prop1_physik(basf14,    7.0,  8.0,   538,        ,      ,      ,      ).
prop1_physik(basf2,     8.2,  9.3,   493,      640,     ,      ,      ).
prop1_physik(basf5,     7.9,  8.9,   502,      688,     ,      ,      ).
prop1_physik(basf50,    5.5,  6.5,   500,      589,     ,      ,      ).
prop1_physik(basf51,    5.4,  6.4,   522,      630,   0.536,   0.722  ).
prop1_physik(basf52,    5.2,  6.2,   533,      628,     ,      ,      ).
prop1_physik(basf54,    7.3,  8.2,   515,      627,     ,      ,      ).
prop1_physik(basf55,    5.1,  6.0,   509,      637,   0.544,   0.892  ).
prop1_physik(basf56,    8.1,  9.2,   491,        ,    ,0.511,0.808    ).
prop1_physik(basf57,    7.1,  8.0,   549,        ,    ,0.548,0.872    ).
prop1_physik(basf6,     7.4,  8.6,   570,      706,     ,      ,      ).
prop1_physik(basf64,    7.3,  8.7,   580,      712,     ,      ,      ).
prop1_physik(bk1,       7.7,  8.9,   547,      730,   0.825,   1.069  ).
prop1_physik(bk10,      5.8,  6.6,   532,      753,     ,      ,      ).
prop1_physik(bk3,       5.3,  6.1,   553,      727,     ,      ,      ).
prop1_physik(bk6,       7.8,  9.1,   537,      684,     ,      ,      ).
prop1_physik(bk7,       7.1,  8.3,   559,      719,   0.858,   1.114  ).
prop1_physik(bk8,       7.4,  8.7,   544,      704,     ,      ,      ).
prop1_physik(fk1,       8.4,  9.1,   390,      640,     ,      ,      ).
prop1_physik(fk3,       8.2,  9.4,   362,      622,     ,      ,      ).
prop1_physik(fk5,       9.2, 10.0,   464,      672,   0.808,   0.925  ).
prop1_physik(fk51,     13.6, 16.0,   405,        ,    ,0.636,0.911    ).
prop1_physik(fk52,     14.4, 16.0,   434,        ,    ,0.716,0.861    ).
prop1_physik(fk54,     14.6, 16.9,   403,        ,      ,      ,      ).
prop1_physik(fn11,      7.5,  8.8,   572,      691,   0.787,   0.657  ).
prop1_physik(f1,        8.7,  9.6,   432,      593,   0.523,   0.783  ).
prop1_physik(f13,       8.7,  9.7,   434,      604,     ,      ,      ).
prop1_physik(f14,       7.9,  9.0,   438,        ,      ,      ,      ).
prop1_physik(f15,       8.1,  9.2,   433,      598,     ,      ,      ).
prop1_physik(f2,        8.2,  9.3,   432,      593,   0.557,   0.780  ).
prop1_physik(f3,        8.0,  9.1,   438,      602,     ,      ,      ).
prop1_physik(f4,        8.3,  9.2,   436,      614,   0.553,   0.768  ).
prop1_physik(f5,        8.0,  9.0,   438,      608,     ,      ,      ).
prop1_physik(f6,        8.5,  9.4,   439,      605,     ,      ,      ).
prop1_physik(f7,        9.8, 10.7,   429,      580,     ,      ,      ).
prop1_physik(f8,        8.2,  9.3,   446,      609,     ,      ,      ).
prop1_physik(f9,        7.7,  8.6,   468,      625,     ,      ,      ).
prop1_physik(kf1,       8.8, 10.0,   478,      649,     ,      ,      ).
prop1_physik(kf3,       8.1,  9.4,   470,      657,     ,      ,      ).
prop1_physik(kf50,      7.3,  8.3,   475,        ,    ,0.754,1.093    ).
prop1_physik(kf6,       6.9,  8.0,   446,        ,      ,      ,      ).
prop1_physik(kf9,       6.8,  7.9,   445,      661,     ,      ,      ).
prop1_physik(kzfn1,     7.1,  7.8,   460,      675,     ,      ,      ).
prop1_physik(kzfn2,     6.1,  6.8,   422,      630,     ,      ,      ).
prop1_physik(kzfsn2,    4.5,  5.5,   499,        ,      ,      ,      ).
prop1_physik(kzfsn4,    4.5,  5.5,   492,      594,   0.636,   0.766  ).
prop1_physik(kzfsn5,    4.5,  5.5,   501,        ,      ,      ,      ).
prop1_physik(kzfsn7,    4.8,  5.8,   512,        ,      ,      ,      ).
prop1_physik(kzfsn9,    4.1,  5.1,   512,        ,      ,      ,      ).
prop1_physik(kzfs1,     5.0,  6.0,   472,      547,     ,      ,      ).
prop1_physik(kzfs6,     5.1,  6.1,   482,        ,    ,0.649,0.800    ).
prop1_physik(kzfs8,     5.3,  6.2,   470,        ,    ,0.511,0.841    ).
prop1_physik(kzf6,      5.5,  6.4,   444,        ,      ,      ,      ).
prop1_physik(k10,       6.5,  7.4,   459,      691,     ,      ,      ).
prop1_physik(k11,       6.4,  7.0,   493,      766,     ,      ,      ).
prop1_physik(k3,        8.3,  9.8,   521,      698,     ,      ,      ).
prop1_physik(k4,        7.3,  8.4,   507.     686,     ,      ,      ).
```

```
prop1_physik(k5,      8.2,  9.6,   543,   720,  0.783,  0.950 ).
prop1_physik(k50,     7.0,  8.0,   553,   733,  0.892,  0.964 ).
prop1_physik(k51,     4.3,  4.9,   521,     ,  ,0.804,1.279   ).
prop1_physik(k7,      8.4,  9.7,   513,   712,  ,       ,     ).
prop1_physik(lafn10,  5.7,  6.9,   619,   707,  ,       ,     ).
prop1_physik(lafn21,  6.2,  7.2,   642,   724,  ,       ,     ).
prop1_physik(lafn23,  8.1,  9.2,   609,   704,  ,       ,     ).
prop1_physik(lafn24,  5.4,  6.4,   642,   723,  ,       ,     ).
prop1_physik(lafn28,  5.8,  7.0,   668,   745,  ,       ,     ).
prop1_physik(lafn7,   5.3,  6.4,   500,   573,  ,       ,     ).
prop1_physik(lafn8,   6.6,  7.9,   545,   638,  0.582,         ).
prop1_physik(laf11a,  5.5,  6.6,   470,   567,  0.435,  0.77  ).
prop1_physik(laf13,   5.7,  6.8,   582,     ,  ,       ,     ).
prop1_physik(laf2,    8.1,  9.2,   644,   740,  ,       ,     ).
prop1_physik(laf20,   7.4,  8.5,   609,   743,  ,       ,     ).
prop1_physik(laf22a,  6.9,  8.0,   670,   783,  0.532,  0.80  ).
prop1_physik(laf25,   5.8,  7.0,   588,   678,  0.507,  0.850 ).
prop1_physik(laf26,   5.6,  6.7,   553,   638,  0.565,  0.861 ).
prop1_physik(laf3,    7.6,  8.7,   647,   740,  ,       ,     ).
prop1_physik(laf9,    7.2,  8.1,   450,   555,  ,       ,     ).
prop1_physik(lakl12,  6.8,  8.0,   636,   764,  ,       ,     ).
prop1_physik(lakl21,  6.1,  7.4,   641,     ,  ,       ,  ,  ).
prop1_physik(lakn12,  7.6,  9.0,   621,   714,  ,       ,     ).
prop1_physik(lakn13,  8.4,  9.5,   614,   718,  ,       ,     ).
prop1_physik(lakn14,  5.5,  6.8,   661,   734,  ,       ,     ).
prop1_physik(lakn22,  6.6,  7.4,   679,   807,  ,       ,     ).
prop1_physik(lakn6,   7.0,  8.1,   634,   733,  ,       ,     ).
prop1_physik(lakn7,   7.1,  8.2,   618,   716,  ,       ,     ).
prop1_physik(lak10,   5.7,  6.9,   620,   703,  ,       ,     ).
prop1_physik(lak11,   7.2,  8.4,   616,   695,  ,       ,     ).
prop1_physik(lak16a,  5.5,  6.7,   641,   716,  0.589,  0.870 ).
prop1_physik(lak20,   8.2,  9.3,   592,   703,  ,       ,     ).
prop1_physik(lak21,   6.8,  7.9,   627,   716,  ,       ,     ).
prop1_physik(lak23,   7.9,  9.0,   608,   696,  ,       ,     ).
prop1_physik(lak28,   5.7,  6.8,   625,     ,  ,0.595,0.837   ).
prop1_physik(lak31,   5.7,  6.8,   653,   733,  ,       ,     ).
prop1_physik(lak33,   6.0,  7.0,   664,     ,  ,0.554,0.900   ).
prop1_physik(lak8,    5.6,  6.7,   640,   720,  ,       ,     ).
prop1_physik(lak9,    6.3,  7.6,   650,   722,  0.649,  0.908 ).
prop1_physik(lasfn15, 6.5,  7.2,   671,   753,  ,       ,     ).
prop1_physik(lasfn18, 6.0,  7.0,   660,     ,  ,       ,  ,  ).
prop1_physik(lasfn30, 6.2,  7.3,   672,   740,  ,       ,     ).
prop1_physik(lasfn31, 6.8,  7.8,   760,     ,  ,       ,  ,  ).
prop1_physik(lasfn9,  7.4,  8.4,   698,   825,  ,       ,     ).
prop1_physik(lasf11,  5.8,  7.0,   644,   724,  ,       ,     ).
prop1_physik(lasf13,  6.2,  7.3,   619,     ,  ,0.427,0.802   ).
prop1_physik(lasf3,   5.5,  6.7,   630,   707,  ,       ,     ).
prop1_physik(lasf32,  7.9,  9.4,   544,   663,  ,       ,     ).
prop1_physik(lasf33,  8.7,  9.8,   543,     ,  ,       ,  ,  ).
prop1_physik(lasf8,   5.9,  7.0,   476,   558,  ,       ,     ).
prop1_physik(lf1,     8.5,  9.7,   435,   612,  ,       ,     ).
prop1_physik(lf2,     8.9, 10.2,   440,   600,  ,       ,     ).
prop1_physik(lf3,     8.1,  9.2,   463,   632,  ,       ,     ).
prop1_physik(lf4,     8.1,  9.3,   442,   620,  ,       ,     ).
prop1_physik(lf5,     9.1, 10.3,   419,   585,  0.657,  0.866 ).
prop1_physik(lf6,     8.5,  9.7,   431,   611,  ,       ,     ).
prop1_physik(lf7,     7.9,  9.0,   442,     ,  ,       ,  ,  ).
prop1_physik(lf8,     8.5,  9.6,   439,   612,  ,       ,     ).
```

```
prop1_physik(lgsk2,    12.1, 13.8,   515,        ,      ,      ,0.866).
prop1_physik(llf1,      8.1,  9.2,   448,   628,      ,      ,      ).
prop1_physik(llf2,      7.9,  9.1,   440,   629,      ,      ,      ).
prop1_physik(llf3,      7.0,  7.9,   518,        ,      ,      ,      ).
prop1_physik(llf4,      8.2,  9.3,   465,   640,      ,      ,      ).
prop1_physik(llf6,      7.5,  8.5,   422,   627,      ,      ,      ).
prop1_physik(llf7,      7.3,  8.4,   422,   625,      ,      ,      ).
prop1_physik(pk1,       6.2,  6.8,   572,        ,      ,      ,      ).
prop1_physik(pk2,       6.9,  8.1,   568,   721,      ,      ,      ).
prop1_physik(pk3;       7.1,  8.3,   567,   710,  0.779,  1.193 ).
prop1_physik(pk50,      8.8,  9.9,   496,   620,  0.812,  0.772 ).
prop1_physik(pk51,     13.1, 14.8,   488,        ,      ,      ,      ).
prop1_physik(psk2,      6.4,  7.5,   608,   724,      ,      ,      ).
prop1_physik(psk3,      6.2,  7.4,   602,   736,  0.682,  0.990 ).
prop1_physik(psk50,     8.6,  9.9,   543,   652,      ,      ,      ).
prop1_physik(psk52,     8.5,  9.9,   597,        ,      ,      ,      ).
prop1_physik(psk53,     9.4, 10.7,   614,   701,  0.603,  0.612 ).
prop1_physik(sfl56,     8.7, 10.1,   591,   693,      ,      ,      ).
prop1_physik(sfl6,      9.0, 10.3,   585,        ,      ,      ,      ).
prop1_physik(sfn64,     8.5,  9.7,   578,   666,      ,      ,      ).
prop1_physik(sf1,       8.1,  9.1,   417,   566,      ,      ,      ).
prop1_physik(sf10,      7.5,  8.5,   454,   595,  0.465,  0.741 ).
prop1_physik(sf11,      6.1,  6.9,   503,   635,  0.431,  0.737 ).
prop1_physik(sf12,      7.8,  8.8,   452,   611,      ,      ,      ).
prop1_physik(sf13,      7.1,  8.0,   472,   604,      ,      ,      ).
prop1_physik(sf14,      6.6,  7.4,   478,   617,      ,      ,      ).
prop1_physik(sf15,      7.9,  9.0,   455,   595,      ,      ,      ).
prop1_physik(sf16,      8.4,  9.3,   443,     6,      ,      ,      ).
prop1_physik(sf17,      8.5,  9.3,   434,        ,      ,      ,      ).
prop1_physik(sf18,      8.1,  9.2,   422,   561,      ,      ,      ).
prop1_physik(sf19,      7.7,  8.6,   434,   593,      ,      ,      ).
prop1_physik(sf2,       8.4,  9.2,   441,   600,  0.498,  0.735 ).
prop1_physik(sf3,       8.4,  9.5,   415,   548,  0.423,  0.706 ).
prop1_physik(sf4,       8.0,  9.0,   420,   552,  0.410,  0.650 ).
prop1_physik(sf5,       8.2,  9.2,   425,   580,      ,      ,      ).
prop1_physik(sf50,     10.1, 11.3,   401,   543,  0.645,  0.699 ).
prop1_physik(sf51,      7.8,  8.6,   455,        ,      ,      ,      ).
prop1_physik(sf52,      9.5, 10.5,   406,   547,      ,      ,      ).
prop1_physik(sf53,      8.2,  9.3,   420,   557,      ,      ,      ).
prop1_physik(sf54,      7.7,  8.8,   432,        ,      ,      ,      ).
prop1_physik(sf55,      8.2,  9.3,   421,        ,      ,      ,      ).
prop1_physik(sf56,      7.9,  8.8,   429,   556,      ,      ,      ).
prop1_physik(sf57,      8.3,  9.2,   422,   519,      ,      ,      ).
prop1_physik(sf58,      9.0, 10.1,   390,   471,      ,      ,      ).
prop1_physik(sf59,      9.4, 10.3,   362,        ,0.306,0.506      ).
prop1_physik(sf6,       8.1,  9.0,   423,   538,  0.389,  0.673 ).
prop1_physik(sf61,      7.9,  9.0,   424,        ,0.414,0.800      ).
prop1_physik(sf62,      8.2,  9.0,   439,   581,  0.473,  0.802 ).
prop1_physik(sf63,      8.2,  9.0,   430,        ,0.431,0.744      ).
prop1_physik(sf7,       7.9,  8.9,   448,   608,      ,      ,      ).
prop1_physik(sf8,       8.2,  9.1,   423,   576,      ,      ,      ).
prop1_physik(sf9,       8.1,  9.2,   434,   584,      ,      ,      ).
prop1_physik(skn18,     6.4,  7.6,   643,   751,      ,      ,      ).
prop1_physik(sk1,       6.1,  7.1,   650,   786,      ,      ,      ).
prop1_physik(sk10,      7.0,  8.0,   624,   744,      ,      ,      ).
prop1_physik(sk11,      6.5,  7.6,   610,   760,      ,      ,      ).
prop1_physik(sk12,      6.4,  7.5,   633,   765,      ,      ,      ).
prop1_physik(sk13,      6.8,  7.9,   620,   759,      ,      ,      ).
```

```
prop1_physik(sk14,      6.0,  7.0,   649,    773,   0.636,   0.851 ).
prop1_physik(sk15,      6.9,  7.7,   634,    744,   ,        ).
prop1_physik(sk16,      6.3,  7.3,   638,    750,   0.578,   0.818 ).
prop1_physik(sk19,      6.4,  7.5,   649,    778,   0.611,   0.808 ).
prop1_physik(sk2,       6.0,  7.0,   654,    823,   0.595,   0.776 ).
prop1_physik(sk20,      6.4,  7.5,   605,    ,      ,  ,     ).
prop1_physik(sk3,       6.3,  7.4,   644,    777,   ,        ).
prop1_physik(sk4,       6.4,  7.4,   643,    767,   0.582,   0.875 ).
prop1_physik(sk5,       5.5,  6.5,   658,    791,   ,        ).
prop1_physik(sk51,      8.9, 10.1,   597,    684,   ,        ).
prop1_physik(sk52,      6.0,  7.2,   624,    ,      ,  ,     ).
prop1_physik(sk55,      6.0,  7.2,   605,    700,   ,        ).
prop1_physik(sk6,       6.2,  7.2,   648,    788,   ,        ).
prop1_physik(sk7,       6.4,  7.4,   643,    772,   ,        ).
prop1_physik(sk8,       6.0,  7.0,   638,    802,   ,        ).
prop1_physik(sk9,       6.0,  7.0,   641,    810,   ,        ).
prop1_physik(sskn5,     6.8,  7.9,   641,    751,   0.574,   ).
prop1_physik(sskn8,     7.1,  8.2,   597,    730,   ,        ).
prop1_physik(ssk1,      6.3,  7.3,   621,    759,   0.561,   0.764 ).
prop1_physik(ssk2,      6.2,  7.1,   636,    773,   0.561,   0.763 ).
prop1_physik(ssk3,      6.6,  7.6,   615,773,  ,        ).
prop1_physik(ssk4,      6.1,  7.1,   639,    791,   ,        ).
prop1_physik(ssk50,     7.4,  8.5,   612,    746,   ,        ).
prop1_physik(ssk51,     7.6,  8.7,   616,    771,   ,        ).
prop1_physik(ssk52,     6.7,  7.6,   639,    ,      ,0.553,0.872 ).
prop1_physik(tifn5,     9.0, 10.2,   472,    ,      ,0.808,0.926 ).
prop1_physik(tif1,      9.1,  9.9,   443,    ,      ,  ,     ).
prop1_physik(tif2,      8.6,  9.5,   453,    648,   ,        ).
prop1_physik(tif3,     10.1, 11.0,   419,    601,   ,        ).
prop1_physik(tif4,      8.9,  9.9,   463,    ,      ,  ,     ).
prop1_physik(tif6,     13.9, 16.7,   410,    494,   ,        ).
prop1_physik(tik1,     10.3, 11.3,   340,    ,      ,0.842,0.773 ).
prop1_physik(tisf1,     9.0, 10.4,   495,    ,      ,  ,     ).
prop1_physik(ubk7,      7.0,  8.3,   563,    716,   ,        ).
prop1_physik(uk50,      7.0,  8.1,   554,    735,   ,        0.964 ).
prop1_physik(zkn7,      4.5,  5.4,   528,    721,   0.770,   1.042 ).
prop1_physik(zk1,   7.5, 8.6,562,    732,   ,        ).
prop1_physik(zk5,       8.7, 10.2,   534,    726,   ,        ).
```

This base of facts contains the physical (chemical and mechanical) properties of the optical glasses.

The parameters are:

Rho: Density.
Emo: Elasticity coefficient
Mue: Poisson coefficient
Hk: Knoop Hardness
B: Presence of pubbles
Cr: Climatic resistance
Fr: Stains resistance
Sr: Acid resistance
Ar: alkaline resistance

```
%prop2_physik(Glass_Name,Rho,    Emo,  Mue,Hk,  B,  Cr,  Fr,   Sr,Ar ).

prop2_physik(bafn10,    3.76,89,0.281,480,      0,  2,  3,  51,1.2      ).
prop2_physik(bafn11,    3.76,88,0.279,470,      0,  2,  5,  52,  1.2    ).
prop2_physik(baf6, 3.17,77,  0.234,  460,  1,  2,  0,  2,  2.0  ).
prop2_physik(baf12, 3.60,79,  0.262,  460,  1,  1,  1,  52,     ).
prop2_physik(baf13, 3.80,84,  0.273,  470,  1,  2,  5,  52,  1.2  ).
prop2_physik(baf3,  3.28,64,  0.261,  420,  1,  1,  0,  1,  1.0  ).
prop2_physik(baf4,  3.50,66,  0.247,  400,  1,  2,  0,  2,  1.0  ).
prop2_physik(baf5,  3.54,71,  0.255,  450,  1,  2,  1,  3,  1.0  ).
prop2_physik(baf50, 3.80,93,  0.266,  490,  0,  2,  1,  51,  1.2  ).
prop2_physik(baf51, 3.42,89,  0.266,  470,  0,  3,  1,  51,  1.2  ).
prop2_physik(baf52, 3.32,72,  0.259,  440,  0,  2,  1,  1,  1.2  ).
prop2_physik(baf53, 3.75,90,  0.264,  510,  0,  2,  1.0,  4,  1.2  ).
prop2_physik(baf54, 3.78,90,  0.269,  530,  1,  2,  1,  51,     ).
prop2_physik(baf8,  3.67,73,  0.260,  460,  0,  1,  2.0,  4,  1.0  ).
prop2_physik(baf9,  3.85,78,  0.269,  460,  0,  2,  2,  51,  1.0  ).
prop2_physik(bak1,  3.19,74,  0.253,  460,  1,  2,  1,  4,  1.2  ).
prop2_physik(bak2,  2.86,71,  0.233,  450,  1,  2,  0,  1,  1.0  ).
prop2_physik(bak4,  3.10,77,  0.241,  470,  1,  2,  1.0,  3,  1.0  ).
prop2_physik(bak5,  3.02,72,  0.241,  460,  1,  2,  0,  2,  1.0  ).
prop2_physik(bak50, 2.93,81,  0.259,  520,  1,  1,  0,  3,  1.0  ).
prop2_physik(bak6,  3.10,79,  0.241,  480,  1,  2,  0,  3,  1.0  ).
prop2_physik(balf3, 3.18,70,  0.250,  450,  2,  1,  1.0,  3.0,  1.0  ).
prop2_physik(balf4, 3.17,76,  0.244,  460,  1,  2,  0,  3,  1.0  ).
prop2_physik(balf5, 2.95,65,  0.236,  440,  1,  2,  0,  1,  1.0  ).
prop2_physik(balf50,3.11,79,  0.247,  480,  1,  2,  0,  1,  2.2  ).
prop2_physik(balf51,3.03,75,  0.240,  460,  0,  1,  0,  1,  1.0  ).
prop2_physik(balf6, 3.32,75,  0.254,  470,  2,  1,  1,  51,     ).
prop2_physik(balf8, 2.99,65,  0.233,  420,  2,  1,  0,  1,  1.0  ).
prop2_physik(balkn3,2.61,72,  0.212,  470,  1,  2,  0,  1,  1.0  ).
prop2_physik(balk1, 2.70,68,  0.234,  430,  1,  2,  0,  2,     ).
prop2_physik(basf1, 3.66,62,  0.242,  400,  1,  2,  1,  3,  1.0  ).
prop2_physik(basf10,3.91,67,  0.256,  400,  2,  2,  3.0,  51,  1.0  ).
prop2_physik(basf12,3.90,74,  0.261,  430,  1,  2,  5,  52,  1.0  ).
prop2_physik(basf13,3.97,81,  0.268,  450,  1,  2,  3,  51,  1.2  ).
prop2_physik(basf14,4.00,77,  0.258,  420,  3,  2,  1,  4,  1.2  ).
prop2_physik(basf2, 3.90,66,  0.245,  410,  1,  2,  1,  3,  1.0  ).
prop2_physik(basf5, 3.49,63,  0.231,  400,  2,  2,  0,  2,     ).
prop2_physik(basf50,4.07,77,  0.285,  450,  1,  4,  4,  53,  1.3  ).
prop2_physik(basf51,4.31,80,  0.289,  450,  1,  2,  4,  52,  1.2  ).
prop2_physik(basf52,3.96,86,  0.283,  480,  1,  2,  5,  53,  1.2  ).
prop2_physik(basf54,4.41,71,  0.265,  440,  1,  2,  5,  52,  1.2  ).
prop2_physik(basf55,3.95,74,  0.269,  450,  1,  2,  2.0,  52,     ).
prop2_physik(basf56,3.85,66,  0.250,  430,  1,  1,  1,  3,  1.0  ).
prop2_physik(basf57,3.73,77,  0.248,  450,  1,  1,  1,  3,  1.2  ).
prop2_physik(basf6, 3.79,80,  0.263,  450,  1,  2,  3,  52,  1.2  ).
prop2_physik(basf64,3.20,105, 0.262,  540,  0,  1,  0,  2,  1.2  ).
prop2_physik(bk1,   2.46,74,  0.210,  480,  0,  2,  0,  1,  2.0  ).
prop2_physik(bk10,  2.39,72,  0.205,  490,  1,  1,  0,  1,  1.0  ).
prop2_physik(bk3,   2.37,74,  0.204,  510,  2,  2,  0,  2,  1.0  ).
prop2_physik(bk6,   2.69,81,  0.222,  490,  0,  3,  0,  2,  2.0  ).
prop2_physik(bk7,   2.51,81,  0.208,  520,  0,  2,  0,  1,  2.0  ).
prop2_physik(bk8,   2.57,80,  0.213,  520,  1,  2,  0,  1,  2.0  ).
prop2_physik(fk1,   2.31,50,  0.251,  370,  1,  2,  2,  51,  3.4  ).
prop2_physik(fk3,   2.27,46,  0.245,  340,  1,  2,  3,  51,  2.4  ).
prop2_physik(fk5,   2.45,62,  0.205,  450,  1,  2,  1,  4,  3.0  ).
prop2_physik(fk51,  3.73,79,  0.287,  360,  1,  2,  0,  51,  2.2  ).
```

```
prop2_physik(fk52,  3.64,79,  0.290,  360,  1,  2,  0,  52,  2.2 ).
prop2_physik(fk54,  3.18,76,  0.286,  320,  1,  1,  1,  52,  2.3 ).
prop2_physik(fn11,  2.66,84,  0.230,  510,  1,  1,  0,  1,  1.0 ).
prop2_physik(f1,  3.65,56,  0.299,  350,  0,  1,  0,  2,  2.3 ).
prop2_physik(f13,  3.62,55,  0.226,  370,  1,  1,  1.0,  1,  2.3 ).
prop2_physik(f14,  3.44,58,  0.218,  380,  1,  1,  0,  1,  2.2 ).
prop2_physik(f15,  3.48,58,  0.219,  390,  1,  1,  0,  1,  2.3 ).
prop2_physik(f2,  3.61,58,  0.225,  370,  0,  1,  0,  1,  2.3 ).
prop2_physik(f3,  3.55,58,  0.224,  360,  0,  1,  0,  1,  2.3 ).
prop2_physik(f4,  3.58,55,  0.225,  360,  0,  1,  0,  1,  2.3 ).
prop2_physik(f5,  3.47,58,  0.220,  380,  0,  1,  0,  1,  2.3 ).
prop2_physik(f6,  3.76,57,  0.231,  360,  0,  2,  1,  3.0,  2.3 ).
prop2_physik(f7,  3.62,55,  0.239,  360,  0,  2,  1,  3,  2.3 ).
prop2_physik(f8,  3.38,60,  0.222,  370,  0,  1,  0,  1,  2.3 ).
prop2_physik(f9,  3.56,63,  0.225,  400,  1,  2,  1,  3.0,  2.2 ).
prop2_physik(kf1,  2.78,67,  0.221,  430,  1,  1,  0,  1,  2.1 ).
prop2_physik(kf3,  2.56,66,  0.216,  410,  1,  1,  0,  1,  1.0 ).
prop2_physik(kf50,  2.70,66,  0.203,  430,  1,  2,  0,  2,  1.0 ).
prop2_physik(kf6,  2.67,66,  0.201,  420,  1,  1,  0,  1,  2.0 ).
prop2_physik(kf9,  2.71,67,  0.202,  440,  0,  1,  0,  1,  1.0 ).
prop2_physik(kzfn1,  2.71,61,  0.224,  430,  0,  1,  1,  2,  1.2 ).
prop2_physik(kzfn2,  2.54,54,  0.221,  380,  0,  2,  2,  2,  3.2 ).
prop2_physik(kzfsn2,2.56,66,  0.267,  420,  0,  3,  4,  52,  4.2 ).
prop2_physik(kzfsn4,3.20,60,  0.276,  380,  0,  3,  2,  52,  4.2 ).
prop2_physik(kzfsn5,3.46,65,  0.275,  410,  0,  3,  2,  52,  4.2 ).
prop2_physik(kzfsn7,3.51,75,  0.274,  450,  0,  3,  2,  51,  4.2 ).
prop2_physik(kzfsn9,3.01,66,  0.271,  420,  0,  3,  2,  52,  3.2 ).
prop2_physik(kzfs1,  3.14,55,  0.282,  370,  0,  4,  4,  53,  4.1 ).
prop2_physik(kzfs6,  3.03,59,  0.282,  370,  0,  4,  4,  53,  4.2 ).
prop2_physik(kzfs8,  4.13,70,  0.275,  390,  0,  4,  2.0,  52,  3.2 ).
prop2_physik(kzf6,  2.54,52,  0.212,  380,  1,  3,  2,  51,  4.2 ).
prop2_physik(k10,  2.52,66,  0.192,  440,  1,  1,  0,  1,  1.0 ).
prop2_physik(k11,  2.50,67,  0.202,  460,  2,  1,  0,  1,  1.0 ).
prop2_physik(k3,  2.54,71,  0.220,  470,  1,  1,  0,  1,  1.0 ).
prop2_physik(k4,  2.63,71,  0.212,  460,  2,  1,  0,  1,  1.2 ).
prop2_physik(k5,  2.59,71,  0.227,  450,  0,  1,  0,  1,  1.0 ).
prop2_physik(k50,  2.62,72,  0.222,  450,  0,  1,  0,  1,  1.0 ).
prop2_physik(k51,  2.47,71,  0.197,  460,  1,  2,  0,  2,  2.0 ).
prop2_physik(k7,  2.53,69,  0.218,  450,  0,  3,  0,  2,  1.0 ).
prop2_physik(lafn10,4.16,117,  0.289,  620,  1,  1,  1,  51,  1.0 ).
prop2_physik(lafn21,4.34,126,  0.294,  650,  0,  1,  1,  51,  1.0 ).
prop2_physik(lafn23,4.20,82,  0.284,  400,  0,  4,  2,  53,  2.2 ).
prop2_physik(lafn24,4.04,120,  0.291,  630,  1,  1,  1,  51,  1.0 ).
prop2_physik(lafn28,4.24,123,  0.292,  630,  0,  1,  1,  51,  1.0 ).
prop2_physik(lafn7,  4.38,80,  0.280,  430,  1,  3,  2.0,  52,  2.2 ).
prop2_physik(lafn8,  4.02,98,  0.293,  500,  1,  2,  4,  53,  1.2 ).
prop2_physik(laf11a,4.60,71,  0.261,  390,  0,  1,  2,  52,  1.2 ).
prop2_physik(laf13,  4.55,95,  0.290,  490,  1,  1,  1,  4,  1.2 ).
prop2_physik(laf2,  4.34,93,  0.289,  480,  1,  2,  3,  5.2,  1.2 ).
prop2_physik(laf20,  3.87,94,  0.273,  520,  0,  3,  1.0,  4,  1.2 ).
prop2_physik(laf22a,4.21,100,  0.281,  500,  0,  1,  1,  52,  1.2 ).
prop2_physik(laf25,  4.45,111,  0.303,  580,  1,  1,  2,  52,  1.0 ).
prop2_physik(laf26,  4.20,96,  0.292,  520,  1,  2,  1,  52,  1.2 ).
prop2_physik(laf3,  4.14,95,  0.286,  510,  1,  2,  3,  5.2,  1.2 ).
prop2_physik(laf9,  4.96,67,  0.270,  350,  1,  2,  5,  53,  1.2 ).
prop2_physik(lakl12,3.32,110,  0.281,  600,  0,  3,  3,  52,  2.2 ).
prop2_physik(lakl21,2.97,109,  0.275,  590,  0,  4,  2,  53,  4.2 ).
prop2_physik(lakn12,4.10,87,  0.288,  470,  1,  4,  1,  53,  4.2 ).
```

```
prop2_physik(lakn13,4.24,87,  0.289,  470,  1,  4,  2,  53,  3.2  ).
prop2_physik(lakn14,3.63,111, 0.283,  600,  1,  3,  2,  52,  1.0  ).
prop2_physik(lakn22,3.73,90,  0.268,  510,  1,  2,  3.0, 4,   1.0  ).
prop2_physik(lakn6, 3.80,88,  0.274,  460,  1,  3,  4,   53,  3.2  ).
prop2_physik(lakn7, 3.84,90,  0.277,  460,  1,  4,  2,   53,  4.2  ).
prop2_physik(lak10, 3.81,111, 0.288,  580,  1,  2,  2,   52,  1.2  ).
prop2_physik(lak11, 3.79,90,  0.285,  520,  1,  4,  5,   53,  4.2  ).
prop2_physik(lak16a,3.95,117, 0.291,  630,  0,  1,  1,   52,  1.0  ).
prop2_physik(lak20, 4.13,90,  0.289,  440,  1,  4,  5,   53,  2.2  ).
prop2_physik(lak21, 3.74,91,  0.272,  460,  1,  4,  2,   53,  4.2  ).
prop2_physik(lak23, 4.00,86,  0.287,  460,  1,  4,  2,   53,  4.1  ).
prop2_physik(lak28, 4.09,117, 0.291,  620,  1,  2,  1,   52,  1.0  ).
prop2_physik(lak31, 3.68,114, 0.285,  610,  0,  4,  0,   52,  1.0  ).
prop2_physik(lak33, 4.26,124, 0.291,  640,  0,  1,  1,   51,  1.0  ).
prop2_physik(lak8,  3.78,115, 0.289,  590,  1,  3,  2,   52,  1.0  ).
prop2_physik(lak9,  3.51,110, 0.288,  580,  1,  3,  3,   52,  1.2  ).
prop2_physik(lasfn15,4.75,126,0.293,  650,  0,  2,  0,   3,   1.0  ).
prop2_physik(lasfn18,4.82,125,0.289,  620,  0,  1,  0,   3,   1.0  ).
prop2_physik(lasfn30,4.46,124,0.293,  650,  0,  1,  1,   4,   1.0  ).
prop2_physik(lasfn31,5.41,125,0.299,  650,  0,  1,  0,   2,   1.0  ).
prop2_physik(lasfn9,4.44,109, 0.286,  510,  0,  2,  0,   2,        ).
prop2_physik(lasf11,4.62,119, 0.297,  600,  1,  2,  1,   4,   1.0  ).
prop2_physik(lasf13,5.10,110, 0.309,  540,  1,  1,  1.0, 3,   1.0  ).
prop2_physik(lasf3, 4.21,121, 0.289,  610,  0,  1,  1,   51,  1.0  ).
prop2_physik(lasf32,3.52,113, 0.256,  560,  0,  2,  0,   2.0, 1.0  ).
prop2_physik(lasf33,4.48,92,  0.291,  440,  0,  2,  2,   51,  1.2  ).
prop2_physik(lasf8, 4.87,81,  0.290,  420,  1,  3,  2,   52,  1.2  ).
prop2_physik(lf1,   3.16,60,  0.221,  390,  1,  1,  0,   1,   2.0  ).
prop2_physik(lf2,   3.30,61,  0.227,  390,  1,  2,  0,   1,        ).
prop2_physik(lf3,   3.21,63,  0.222,  420,  0,  1,  0,   1,   2.3  ).
prop2_physik(lf4,   3.21,60,  0.219,  400,  1,  2,  0,   1,   2.2  ).
prop2_physik(lf5,   3.22,59,  0.226,  410,  0,  2,  0,   2.0, 2.3  ).
prop2_physik(lf6,   3.11,60,  0.217,  390,  1,  2,  0,   1,   2.3  ).
prop2_physik(lf7,   3.20,59,  0.217,  390,  0,  1,  1.0, 1,   2.3  ).
prop2_physik(lf8,   3.08,60,  0.218,  380,  0,  1,  0,   2,   2.3  ).
prop2_physik(lgsk2, 4.15,76,  0.290,  340,  0,  2,  1.0, 52,  2.3  ).
prop2_physik(llf1,  2.94,60,  0.210,  390,  0,  1,  0,   1,   2.0  ).
prop2_physik(llf2,  2.87,61,  0.207,  410,  1,  1,  0,   1,   2.0  ).
prop2_physik(llf3,  2.99,69,  0.213,  450,  2,  1,  0,   1,   2.0  ).
prop2_physik(llf4,  3.02,64,  0.219,  400,  1,  2,  0,   1,   2.3  ).
prop2_physik(llf6,  2.81,63,  0.205,  420,  2,  1,  0,   1,   1.0  ).
prop2_physik(llf7,  2.98,62,  0.205,  390,  1,  1,  0,   1,   2.0  ).
prop2_physik(pk1,   2.44,74,  0.204,  570,  1,  1,  0,   1,   1.0  ).
prop2_physik(pk2,   2.51,84,  0.209,  520,  0,  3,  0,   2,   2.0  ).
prop2_physik(pk3,   2.59,84,  0.207,  570,  0,  3,  0,   2,   3.0  ).
prop2_physik(pk50,  2.59,66,  0.235,  350,  1,  2,  0,   2.0, 2.0  ).
prop2_physik(pk51,  3.97,73,  0.293,  340,  1,  2,  0,   51,  2.2  ).
prop2_physik(psk2,  3.06,86,  0.237,  490,  1,  4,  2,   51,  3.0  ).
prop2_physik(psk3,  2.91,84,  0.226,  510,  0,  3,  0,   3,   2.0  ).
prop2_physik(psk50, 2.93,71,  0.260,  390,  1,  4,  0,   51,  2.2  ).
prop2_physik(psk52, 3.38,78,  0.278,  400,  1,  3,  0,   52,  1.0  ).
prop2_physik(psk53, 3.60,77,  0.287,  370,  1,  2,  1,   52,  1.0  ).
prop2_physik(sfl56, 3.28,91,  0.255,  530,  0,  1,  0,   2,   1.3  ).
prop2_physik(sfl6,  3.37,93,  0.260,  500,  0,  1,  0,   2,   1    ).
prop2_physik(sfn64, 3.00,93,  0.250,  500,  1,  1,  0,   2.0, 1.2  ).
prop2_physik(sf1,   4.46,57,  0.234,  340,  1,  2,  1,   3,   2.3  ).
prop2_physik(sf10,  4.28,64,  0.232,  370,  1,  1,  0,   1,   1.2  ).
prop2_physik(sf11,  4.74,66,  0.237,  380,  1,  1,  0,   1,   1.2  ).
```

```
prop2_physik(sf12,  3.74,60,  0.228,  360,  1,  1,  0,  1,  2.2  ).
prop2_physik(sf13,  4.36,64,  0.233,  380,  1,  1,  0,  1,  1.2  ).
prop2_physik(sf14,  4.54,65,  0.235,  370,  1,  1,  0,  1,  1.2  ).
prop2_physik(sf15,  4.06,60,  0.235,  370,  1,  1,  0,  1,  1.2  ).
prop2_physik(sf16,  3.85,55,  0.232,  340,  1,  1,  1.0,  2,  2.3  ).
prop2_physik(sf17,  3.87,55,  0.229,  350,  1,  1,  1,  2,  2.2  ).
prop2_physik(sf18,  4.49,56,  0.238,  330,  1,  2,  1,  3,  2.3  ).
prop2_physik(sf19,  4.02,58,  0.228,  360,  1,  1,  1,  2,  2.3  ).
prop2_physik(sf2,  3.86,55,  0.231,  350,  0,  1,  0,  2,  2.3  ).
prop2_physik(sf3,  4.64,56,  0.239,  330,  1,  1,  2,  51,  2.3  ).
prop2_physik(sf4,  4.79,56,  0.241,  330,  1,  1,  2,  51,  2.3  ).
prop2_physik(sf5,  4.07,56,  0.233,  340,  1,  1,  1,  2,  2.3  ).
prop2_physik(sf50,  3.79,56,  0.242,  360,  1,  2,  1,  4,  ).
prop2_physik(sf51,  3.81,61,  0.229,  380,  1,  1,  1.0,  1,  2.2  ).
prop2_physik(sf52,  4.10,57,  0.242,  350,  1,  1,  2,  4,  2.2  ).
prop2_physik(sf53,  4.45,58,  0.238,  340,  1,  1,  1,  3,  2.2  ).
prop2_physik(sf54,  4.56,58,  0.237,  360,  1,  1,  1,  3,  2.3  ).
prop2_physik(sf55,  4.72,56,  0.247,  330,  1,  1,  2,  3,  2.2  ).
prop2_physik(sf56,  4.92,58,  0.242,  330,  1,  1,  1,  3,  2.2  ).
prop2_physik(sf57,  5.51,55,  0.253,  300,  2,  2,  5,  52,  2.3  ).
prop2_physik(sf58,  5.95,52,  0.264,  270,  3,  3,  5,  53,  3.4  ).
prop2_physik(sf59,  6.26,51,  0.269,  250,  3,  4,  5,  53,  3.4  ).
prop2_physik(sf6,  5.18,56,  0.248,  310,  0,  2,  4.0,  52,  2.3  ).
prop2_physik(sf61,  4.64,58,  0.235,  340,  1,  1,  1,  3,  2.2  ).
prop2_physik(sf62,  4.15,57,  0.226,  360,  1,  1,  1,  3,  2.3  ).
prop2_physik(sf63,  4.62,58,  0.235,  340,  1,  1,  1,  3,  2.2  ).
prop2_physik(sf7,  3.80,56,  0.226,  350,  1,  1,  1.0,  1,  2.3  ).
prop2_physik(sf8,  4.22,56,  0.233,  340,  1,  1,  1,  3,  2.3  ).
prop2_physik(sf9,  3.91,58,  0.229,  360,  1,  1,  0,  2.0,  2.3  ).
prop2_physik(skn18, 3.64,88,  0.296,  470,  0,  3,  5.0,  52,  2.2  ).
prop2_physik(sk1,  3.56,79,  0.262,  490,  0,  2,  1,  51,  1.0  ).
prop2_physik(sk10,  3.66,82,  0.273,  480,  0,  3,  3,  52,  2.0  ).
prop2_physik(sk11,  3.08,79,  0.239,  510,  0,  2,  0,  2,  1.0  ).
prop2_physik(sk12,  3.27,78,  0.251,  490,  0,  2,  1,  4,  1.0  ).
prop2_physik(sk13,  3.37,78,  0.258,  470,  0,  2,  2,  51,  1.0  ).
prop2_physik(sk14,  3.44,86,  0.261,  490,  0,  4,  3.0,  51,  2.0  ).
prop2_physik(sk15,  3.64,84,  0.275,  450,  0,  3,  3,  52,  2.0  ).
prop2_physik(sk16,  3.58,89,  0.267,  490,  0,  4,  4,  52,  3.0  ).
prop2_physik(sk19,  3.56,81,  0.265,  500,  0,  2,  1,  51,  2.0  ).
prop2_physik(sk2,  3.55,78,  0.263,  460,  1,  2,  0,  3,  1.0  ).
prop2_physik(sk20,  3.03,79,  0.235,  530,  1,  3,  0,  3.0,  1.0  ).
prop2_physik(sk3,  3.53,83,  0.263,  480,  0,  3,  2,  51,  2.2  ).
prop2_physik(sk4,  3.57,82,  0.268,  500,  0,  3,  2,  51,  2.0  ).
prop2_physik(sk5,  3.30,84,  0.256,  480,  0,  3,  1,  4,  2.0  ).
prop2_physik(sk51,  3.52,75,  0.291,  380,  0,  2,  3,  51,  1.3  ).
prop2_physik(sk52,  3.30,97,  0.283,  520,  0,  2,  4,  52,  1.0  ).
prop2_physik(sk55,  3.35,97,  0.260,  550,  0,  4,  2,  52,  2.3  ).
prop2_physik(sk6,  3.60,79,  0.268,  450,  0,  2,  3.0,  51,  1.0  ).
prop2_physik(sk7,  3.51,84,  0.264,  490,  0,  3,  2,  51,  2.0  ).
prop2_physik(sk8,  3.57,79,  0.265,  470,  0,  2,  2.0,  51,  1.0  ).
prop2_physik(sk9,  3.60,79,  0.266,  480,  0,  2,  2,  51,  1.0  ).
prop2_physik(sskn5, 3.71,88,  0.278,  470,  0,  2,  3,  52,  2.2  ).
prop2_physik(sskn8, 3.33,83,  0.256,  460,  0,  2,  1.0,  3,  2.2  ).
prop2_physik(ssk1,  3.63,79,  0.263,  450,  1,  2,  3,  51,  1.0  ).
prop2_physik(ssk2,  3.67,78,  0.255,  460,  0,  2,  2,  51,  1.0  ).
prop2_physik(ssk3,  3.61,75,  0.261,  460,  0,  2,  2.0,  4,  1.0  ).
prop2_physik(ssk4,  3.63,79,  0.265,  460,  0,  2,  1,  51,  1.0  ).
prop2_physik(ssk50, 3.42,84,  0.264,  480,  0,  3,  1,  1,  2.2  ).
```

```
prop2_physik(ssk51, 3.28,80,  0.253,  470,  0,  3,  0,  1,  2.2  ).
prop2_physik(ssk52, 3.76,89,  0.267,  500,  0,  3,  1,  52,  1.2  ).
prop2_physik(tifn5,  2.71,66,  0.244,  410,  1,  1,  0,  2.0,  1.2  ).
prop2_physik(tif1,   2.47,58,  0.239,  440,  1,  1,  0,  2,  1.0  ).
prop2_physik(tif2,   2.51,60,  0.238,  450,  1,  1,  0,  2,  ).
prop2_physik(tif3,   2.61,60,  0.245,  390,  1,  1,  0,  2,  1.0  ).
prop2_physik(tif4,   2.68,66,  0.243,  410,  1,  1,  0,  2,  1.0  ).
prop2_physik(tif6,   2.79,65,  0.262,  310,  1,  1,  0,  51,  3.2  ).
prop2_physik(tik1,   2.39,40,  0.254,  330,  1,  4,  3,  51,  ).
prop2_physik(tisf1,  2.81,82,  0.239,  450,  0,  1,  1.0,  3,  ).
prop2_physik(ubk7,  2.51,81,  0.212,  500,  0,  2,  0,  1,  2.0  ).
prop2_physik(uk50,  2.62,73,  0.223,  460,  0,  1,  0,  1,  1.0  ).
prop2_physik(zkn7,  2.49,71,  0.214,  450,  1,  1,  0,  2,  1.2  ).
prop2_physik(zk1,   2.71,68,  0.240,  430,  1,  1,  1,  1,  1.0  ).
prop2_physik(zk5,   2.76,68,  0.238,  430,  1,  1,  0,  2,  1.0  ).
```

Appendix 20: Base of Facts of the Code of the Optical Glass

/*

This base of facts contains the data of the optical glass.
The parameters are:

Glass_name1:	Name of the optical glass
Glass_name2:	Code of the optical glass
Nd:	Refractive indexe of the optical glass at the line d
Nued:	Abbe Number of the optical glass at the line d
Nf-Nc:	The difference between the refractive index at the line f and at the line c; this coefficient is used for the dispersion.
Ne:	Refractive index at the line c
Nuee	Abbe Number at the line e
Nf- Nc'	The difference between the refractive index at the line f' and at the line c'. This coefficient is used for the dispersion.

*/

% entete_verre(Glass_name1,Glass_name2,Nd,Nued, Nf-Nc, Ne, Nuee,Nf'- Nc').

entete_verre(bafn10,670471, 1.67003,47.11, 0.014222, 1.67341, 46.82,0.014384).
entete_verre(bafn11,667484, 1.66672,48.42, 0.013769, 1.67000, 48.13,0.013920).
entete_verre(bafn6,589485, 1.58900,48.45, 0.012158, 1.59189, 48.16,0.012290).
entete_verre(baf12,639452, 1.63930,45.18, 0.014151, 1.64266, 44.88,0.014318).
entete_verre(baf13,669450, 1.66892,44.96, 0.014877, 1.67245, 44.67,0.015055).
entete_verre(baf3,583465, 1.58267,46.47, 0.012538, 1.58565, 46.17,0.012684).
entete_verre(baf4,606439, 1.60562,43.93, 0.013787, 1.60889, 43.63,0.013956).
entete_verre(baf5,607494, 1.60729,49.40, 0.012293, 1.61021, 49.11,0.012426).
entete_verre(baf50,683445, 1.68273,44.50, 0.015341, 1.68637, 44.23,0.015519).
entete_verre(baf51,652449, 1.65224,44.93, 0.014517, 1.65569, 44.64,0.014688).
entete_verre(baf52,609464, 1.60859,46.44, 0.013104, 1.61170, 46.14,0.013257).
entete_verre(baf53,670471, 1.67003,47.10, 0.014226, 1.67341, 46.81,0.014385).
entete_verre(baf54,667482, 1.66672,48.22, 0.013826, 1.67001, 47.94,0.013975).
entete_verre(baf8,624470, 1.62374,47.00, 0.013270, 1.62690, 46.71,0.013422).
entete_verre(baf9,643480, 1.64328,47.96, 0.013414, 1.64647, 47.66,0.013565).
entete_verre(bak1,573575, 1.57250,57.55, 0.009948, 1.57487, 57.27,0.010038).
entete_verre(bak2,540597, 1.53996,59.71, 0.009043, 1.54212, 59.44,0.009120).
entete_verre(bak4,569561, 1.56883,56.13, 0.010135, 1.57125, 55.85,0.010229).
entete_verre(bak5,557587, 1.55671,58.65, 0.009492, 1.55897, 58.37,0.009576).
entete_verre(bak50,568580, 1.56774,57.99, 0.009790, 1.57007, 57.75,0.009872).
entete_verre(bak6,574564, 1.57444,56.40, 0.010186, 1.57687, 56.12,0.010280).
entete_verre(balf3,571529, 1.57135,52.93, 0.010794, 1.57392, 52.64,0.010902).
entete_verre(balf4,580537, 1.57957,53.71, 0.010790, 1.58214, 53.43,0.010896).
entete_verre(balf5,547536, 1.54739,53.63, 0.010207, 1.54982, 53.33,0.010309).
entete_verre(balf50,589514, 1.58893,51.37, 0.011464, 1.59166, 51.08,0.011582).
entete_verre(balf51,574521, 1.57393,52.10, 0.011017, 1.57656, 51.81,0.011129).
entete_verre(balf6,589530, 1.58904,53.01, 0.011111, 1.59169, 52.73,0.011222).
entete_verre(balf8,554512, 1.55361,51.18, 0.010816, 1.55618, 50.89,0.010930).
entete_verre(balkn3,518602, 1.51849,60.25, 0.008606, 1.52054, 59.99,0.008677).

entete_verre(balk1,526600, 1.52642,60.03, 0.008770, 1.52851, 59.75,0.008845).
entete_verre(basf1,626390, 1.62606,38.96, 0.016068, 1.62987, 38.68,0.016283).
entete_verre(basf10,650392, 1.65016,39.15, 0.016608, 1.65410, 38.87,0.016830).
entete_verre(basf12,670392, 1.66998,39.20, 0.017090, 1.67403, 38.92,0.017319).
entete_verre(basf13,698386, 1.69761,38.57, 0.018086, 1.70190, 38.29,0.018330).
entete_verre(basf14,700350, 1.69968,34.96, 0.020012, 1.70442, 34.70,0.020299).
entete_verre(basf2,664358, 1.66446,35.83, 0.018545, 1.66885, 35.56,0.018808).
entete_verre(basf5,603425, 1.60323,42.48, 0.014201, 1.60660, 42.19,0.014378).
entete_verre(basf50,710366, 1.71020,36.62, 0.019395, 1.71480, 36.37,0.019654).
entete_verre(basf51,724381, 1.72373,38.11, 0.018991, 1.72823, 37.85,0.019242).
entete_verre(basf52,702410, 1.70181,41.01, 0.017112, 1.70587, 40.75,0.017320).
entete_verre(basf54,736322, 1.73627,32.15, 0.022902, 1.74169, 31.90,0.023251).
entete_verre(basf55,700347, 1.69981,34.68, 0.020178, 1.70459, 34.42,0.020469).
entete_verre(basf56,657367, 1.65715,36.73, 0.017889, 1.66139, 36.47,0.018137).
entete_verre(basf57,651419, 1.65147,41.90, 0.015547, 1.65516, 41.62,0.015741).
entete_verre(basf6,668419, 1.66755,41.93, 0.015921, 1.67133, 41.64,0.016123).
entete_verre(basf64,701396, 1.70100,39.60, 0.017702, 1.70520, 39.34,0.017927).
entete_verre(bk1,510635, 1.51009,63.46, 0.008038, 1.51201, 63.23,0.008098).
entete_verre(bk10,498670, 1.49782,66.95, 0.007436, 1.49960, 66.77,0.007482).
entete_verre(bk3,498651, 1.49831,65.06, 0.007659, 1.50014, 64.89,0.007707).
entete_verre(bk6,531621, 1.53113,62.15, 0.008546, 1.53317, 61.92,0.008611).
entete_verre(bk7,517642, 1.51680,64.17, 0.008054, 1.51872, 63.96,0.008110).
entete_verre(bk8,520637, 1.52015,63.68, 0.008168, 1.52210, 63.46,0.008227).
entete_verre(fk1, 471673, 1.47069,67.34, 0.006990, 1.47236, 67.14,0.007035).
entete_verre(fk3, 465658, 1.46450,65.77, 0.007063, 1.46619, 65.57,0.007110).
entete_verre(fk5, 487704, 1.48749,70.41, 0.006924, 1.48914, 70.22,0.006966).
entete_verre(fk51,487845, 1.48656,84.47, 0.005760, 1.48794, 84.07,0.005804).
entete_verre(fk52,486818, 1.48605,81.81, 0.005941, 1.48747, 81.41,0.005988).
entete_verre(fk54,437907, 1.43700,90.70, 0.004818, 1.43815, 90.28,0.004853).
entete_verre(fn11,621362, 1.62096,36.18, 0.017163, 1.62502, 35.91,0.017405).
entete_verre(f1, 626357, 1.62588,35.70, 0.017530, 1.63004, 35.45,0.017773).
entete_verre(f13, 622360, 1.62237,36.04, 0.017267, 1.62646, 35.79,0.017506).
entete_verre(f14, 601382, 1.60140,38.23, 0.015731, 1.60513, 37.97,0.015938).
entete_verre(f15, 606378, 1.60565,37.83, 0.016011, 1.60945, 37.56,0.016224).
entete_verre(f2, 620364, 1.62004,36.37, 0.017050, 1.62408, 36.11,0.017284).
entete_verre(f3, 613370, 1.61293,37.04, 0.016549, 1.61685, 36.78,0.016772).
entete_verre(f4, 617366, 1.61659,36.63, 0.016834, 1.62058, 36.37,0.017063).
entete_verre(f5, 603380, 1.60342,38.03, 0.015868, 1.60718, 37.76,0.016078).
entete_verre(f6, 636353, 1.63636,35.34, 0.018005, 1.64062, 35.09,0.018259).
entete_verre(f7, 625356, 1.62536,35.56, 0.017586, 1.62953, 35.30,0.017833).
entete_verre(f8, 596392, 1.59551,39.18, 0.015200, 1.59912, 38.91,0.015398).
entete_verre(f9, 620381, 1.62045,38.08, 0.016293, 1.62431, 37.81,0.016510).
entete_verre(kf1, 540511, 1.54041,51.10, 0.010575, 1.54293, 50.81,0.010685).
entete_verre(kf3, 515547, 1.51454,54.70, 0.009406, 1.51678, 54.43,0.009494).
entete_verre(kf50,531511, 1.53088,51.12, 0.010386, 1.53335, 50.84,0.010491).
entete_verre(kf6, 517522, 1.51742,52.20, 0.009913, 1.51978, 51.93,0.010010).
entete_verre(kf9, 523515, 1.52341,51.49, 0.010166, 1.52583, 51.22,0.010267).
entete_verre(kzfn1,551496, 1.55115,49.64, 0.011104, 1.55379, 49.38,0.011216).
entete_verre(kzfn2,529516, 1.52944,51.63, 0.010255, 1.53188, 51.38,0.010351).
entete_verre(kzfsn2,558542, 1.55836,54.16, 0.010310, 1.56082, 53.97,0.010391).
entete_verre(kzfsn4,613443, 1.61340,44.30, 0.013848, 1.61669, 44.07,0.013994).
entete_verre(kzfsn5,654396, 1.65412,39.63, 0.016507, 1.65804, 39.40,0.016702).
entete_verre(kzfsn7,681372, 1.68064,37.18, 0.018305, 1.68498, 36.95,0.018540).
entete_verre(kzfsn9,599469, 1.59856,46.90, 0.012762, 1.60159, 46.68,0.012887).
entete_verre(kzfs1,613443, 1.61310,44.34, 0.013826, 1.61639, 44.13,0.013966).
entete_verre(kzfs6,592485, 1.59197,48.51, 0.012203, 1.59487, 48.29,0.012319).
entete_verre(kzfs8,720346, 1.72047,34.61, 0.020814, 1.72540, 34.39,0.021094).
entete_verre(kzf6,527511, 1.52682,51.13, 0.010303, 1.52927, 50.90,0.010399).

entete_verre(k10,501564, 1.50137,56.41, 0.008888, 1.50349, 56.15,0.008967).
entete_verre(k11,500614, 1.50013,61.44, 0.008140, 1.50207, 61.21,0.008203).
entete_verre(k3, 518590, 1.51823,58.98, 0.008787, 1.52032, 58.71,0.008862).
entete_verre(k4, 519574, 1.51895,57.40, 0.009041, 1.52111, 57.15,0.009118).
entete_verre(k5, 522595, 1.52249,59.48, 0.008784, 1.52458, 59.22,0.008858).
entete_verre(k50,523602, 1.52257,60.18, 0.008683, 1.52464, 59.94,0.008753).
entete_verre(k51,505596, 1.50518,59.55, 0.008483, 1.50720, 59.34,0.008548).
entete_verre(k7, 511604, 1.51112,60.41, 0.008461, 1.51314, 60.16,0.008530).
entete_verre(lafn10,784439, 1.78443,43.95, 0.017849, 1.78868, 43.70,0.018046).
entete_verre(lafn21,788475, 1.78831,47.47, 0.016605, 1.79226, 47.23,0.016775).
entete_verre(lafn23,689497, 1.68900,49.71, 0.013860, 1.69230, 49.42,0.014009).
entete_verre(lafn24,757478, 1.75719,47.81, 0.015838, 1.76096, 47.57,0.015998).
entete_verre(lafn28,773496, 1.77314,49.57, 0.015598, 1.77686, 49.33,0.015749).
entete_verre(lafn7,750350, 1.74950,34.95, 0.021445, 1.75458, 34.72,0.021736).
entete_verre(lafn8,735416, 1.73520,41.59, 0.017678, 1.73940, 41.32,0.017893).
entete_verre(laf11a,757317, 1.75693,31.70, 0.023875, 1.76258, 31.48,0.024226).
entete_verre(laf13,776378, 1.77551,37.84, 0.020496, 1.78037, 37.58,0.020764).
entete_verre(laf2,744447, 1.74400,44.72, 0.016638, 1.74796, 44.44,0.016832).
entete_verre(laf20,682482, 1.68248,48.20, 0.014159, 1.68585, 47.92,0.014313).
entete_verre(laf22a,782372, 1.78179,37.20, 0.021018, 1.78677, 36.93,0.021304).
entete_verre(laf25,784413, 1.78427,41.30, 0.018989, 1.78878, 41.04,0.019219).
entete_verre(laf26,746400, 1.74597,40.02, 0.018642, 1.75040, 39.76,0.018873).
entete_verre(laf3,717480, 1.71700,47.96, 0.014950, 1.72055, 47.68,0.015113).
entete_verre(laf9,795284, 1.79504,28.39, 0.028001, 1.80166, 28.18,0.028451).
entete_verre(lakl12,678549, 1.67790,54.93, 0.012342, 1.68084, 54.69,0.012450).
entete_verre(lakl21,640597, 1.64048,59.75, 0.010720, 1.64304, 59.55,0.010799).
entete_verre(lakn12,678552, 1.67790,55.20, 0.012280, 1.68083, 54.93,0.012394).
entete_verre(lakn13,694533, 1.69350,53.33, 0.013004, 1.69660, 53.05,0.013130).
entete_verre(lakn14,697554, 1.69680,55.41, 0.012575, 1.69980, 55.19,0.012679).
entete_verre(lakn22,651559, 1.65113,55.89, 0.011650, 1.65391, 55.62,0.011756).
entete_verre(lakn6,643580, 1.64250,57.96, 0.011085, 1.64514, 57.70,0.011181).
entete_verre(lakn7,652585, 1.65160,58.52, 0.011134, 1.65426, 58.27,0.011228).
entete_verre(lak10,720504, 1.72000,50.41, 0.014282, 1.72340, 50.17,0.014418).
entete_verre(lak11,658573, 1.65830,57.26, 0.011497, 1.66104, 57.01,0.011595).
entete_verre(lak16a,734518, 1.73350,51.78, 0.014167, 1.73688, 51.55,0.014295).
entete_verre(lak20,693516, 1.69349,51.56, 0.013450, 1.69669, 51.28,0.013587).
entete_verre(lak21,641601, 1.64050,60.10, 0.010658, 1.64304, 59.85,0.010744).
entete_verre(lak23,669574, 1.66882,57.38, 0.011655, 1.67159, 57.12,0.011757).
entete_verre(lak28,744508, 1.74429,50.77, 0.014659, 1.74778, 50.54,0.014795).
entete_verre(lak31,697564, 1.69673,56.42, 0.012349, 1.69968, 56.21,0.012448).
entete_verre(lak33,754524, 1.75398,52.43, 0.014380, 1.75740, 52.20,0.014510).
entete_verre(lak8,713538, 1.71300,53.83, 0.013245, 1.71616, 53.61,0.013359).
entete_verre(lak9,691547, 1.69100,54.71, 0.012631, 1.69401, 54.48,0.012740).
entete_verre(lasfn15,878381, 1.87800,38.07, 0.023061, 1.88347, 37.83,0.023351).
entete_verre(lasfn18,913324, 1.91348,32.36, 0.028225, 1.92016, 32.14,0.028630).
entete_verre(lasfn30,803464, 1.80318,46.38, 0.017318, 1.80730, 46.13,0.017501).
entete_verre(lasfn31,881410, 1.88067,41.01, 0.021474, 1.88577, 40.76,0.021730).
entete_verre(lasfn9,850322, 1.85026,32.17, 0.026430, 1.85651, 31.93,0.026827).
entete_verre(lasf11,802443, 1.80166,44.26, 0.018111, 1.80597, 44.02,0.018311).
entete_verre(lasf13,855366, 1.85544,36.59, 0.023381, 1.86099, 36.34,0.023694).
entete_verre(lasf3,808406, 1.80801,40.61, 0.019898, 1.81273, 40.36,0.020138).
entete_verre(lasf32,803304, 1.80349,30.40, 0.026433, 1.80974, 30.17,0.026840).
entete_verre(lasf33,806342, 1.80596,34.24, 0.023537, 1.81153, 33.99,0.023877).
entete_verre(lasf8,807316, 1.80741,31.61, 0.025542, 1.81345, 31.38,0.025920).
entete_verre(lf1, 573426, 1.57309,42.58, 0.013458, 1.57629, 42.31,0.013622).
entete_verre(lf2, 589409, 1.58921,40.94, 0.014392, 1.59263, 40.66,0.014575).
entete_verre(lf3, 582421, 1.58215,42.08, 0.013836, 1.58544, 41.80,0.014007).
entete_verre(lf4, 578416; 1.57845,41.59, 0.013908, 1.58175, 41.31,0.014081).

entete_verre(lf5, 581409, 1.58144,40.85, 0.014233, 1.58482, 40.58,0.014413).
entete_verre(lf6, 567428, 1.56732,42.84, 0.013243, 1.57047, 42.56,0.013403).
entete_verre(lf7, 575415, 1.57501,41.49, 0.013860, 1.57830, 41.22,0.014031).
entete_verre(lf8, 564438, 1.56444,43.75, 0.012900, 1.56750, 43.47,0.013055).
entete_verre(lgsk2,586610, 1.58599,61.04, 0.009600, 1.58828, 60.66,0.009698).
entete_verre(llf1, 548458, 1.54814,45.75, 0.011980, 1.55099, 45.47,0.012118).
entete_verre(llf2, 541472, 1.54072,47.17, 0.011464, 1.54344, 46.88,0.011591).
entete_verre(llf3, 560472, 1.56013,47.16, 0.011876, 1.56295, 46.88,0.012008).
entete_verre(llf4, 561452, 1.56138,45.24, 0.012410, 1.56433, 44.95,0.012554).
entete_verre(llf6, 532488, 1.53172,48.76, 0.010905, 1.53431, 48.48,0.011022).
entete_verre(llf7, 549454, 1.54883,45.41, 0.012087, 1.55170, 45.13,0.012226).
entete_verre(pk1,504669, 1.50378,66.92, 0.007528, 1.50558, 66.73,0.007576).
entete_verre(pk2,518651, 1.51821,65.05, 0.007966, 1.52011, 64.85,0.008020).
entete_verre(pk3,525647, 1.52542,64.66, 0.008126, 1.52736, 64.46,0.008181).
entete_verre(pk50,521697, 1.52054,69.71, 0.007467, 1.52232, 69.50,0.007515).
entete_verre(pk51,529770, 1.52855,76.96, 0.006868, 1.53019, 76.57,0.006924).
entete_verre(psk2,569631, 1.56873,63.08, 0.009016, 1.57088, 62.85,0.009083).
entete_verre(psk3,552635, 1.55232,63.46, 0.008704, 1.55440, 63.23,0.008768).
entete_verre(psk50,558673, 1.55753,67.29, 0.008286, 1.55951, 67.03,0.008347).
entete_verre(psk52,603654, 1.60310,65.41, 0.009220, 1.60530, 65.14,0.009292).
entete_verre(psk53,620635, 1.62014,63.52, 0.009763, 1.62247, 63.23,0.009845).
entete_verre(sf156,785261, 1.78470,26.08, 0.030090, 1.79179, 25.87,0.030606).
entete_verre(sf16,805254, 1.80518,25.39, 0.031708, 1.81265, 25.19,0.032259).
entete_verre(sfn64,706303, 1.70585,30.30, 0.023295, 1.71135, 30.07,0.023657).
entete_verre(sf1, 717295, 1.71736,29.51, 0.024307, 1.72311, 29.29,0.024688).
entete_verre(sf10,728284, 1.72825,28.41, 0.025634, 1.73430, 28.19,0.026050).
entete_verre(sf11,785258, 1.78472,25.76, 0.030468, 1.79190, 25.55,0.030998).
entete_verre(sf12,648338, 1.64831,33.84, 0.019158, 1.65285, 33.59,0.019434).
entete_verre(sf13,741276, 1.74077,27.60, 0.026841, 1.74710, 27.38,0.027290).
entete_verre(sf14,762265, 1.76182,26.53, 0.028718, 1.76859, 26.31,0.029211).
entete_verre(sf15,699301, 1.69895,30.07, 0.023246, 1.70445, 29.84,0.023611).
entete_verre(sf16,646341, 1.64611,34.05, 0.018975, 1.65061, 33.80,0.019248).
entete_verre(sf17,650337, 1.65017,33.67, 0.019311, 1.65474, 33.42,0.019591).
entete_verre(sf18,722293, 1.72151,29.25, 0.024671, 1.72734, 29.03,0.025059).
entete_verre(sf19,667330, 1.66680,33.01, 0.020202, 1.67158, 32.76,0.020498).
entete_verre(sf2, 648339, 1.64769,33.85, 0.019135, 1.65222, 33.60,0.019412).
entete_verre(sf3, 740282, 1.74000,28.20, 0.026244, 1.74620, 27.98,0.026666).
entete_verre(sf4, 755276, 1.75520,27.58, 0.027383, 1.76167, 27.37,0.027829).
entete_verre(sf5, 673322, 1.67270,32.21, 0.020884, 1.67764, 31.97,0.021194).
entete_verre(sf50,655329, 1.65473,32.88, 0.019914, 1.65944, 32.63,0.020210).
entete_verre(sf51,660329, 1.66025,32.94, 0.020044, 1.66499, 32.69,0.020342).
entete_verre(sf52,689306, 1.68852,30.62, 0.022484, 1.69384, 30.39,0.022832).
entete_verre(sf53,728287, 1.72830,28.69, 0.025388, 1.73430, 28.47,0.025793).
entete_verre(sf54,741281, 1.74080,28.09, 0.026370, 1.74703, 27.88,0.026797).
entete_verre(sf55,762270, 1.76180,26.95, 0.028267, 1.76847, 26.74,0.028739).
entete_verre(sf56,785261, 1.78470,26.08, 0.030092, 1.79180, 25.87,0.030602).
entete_verre(sf57,847238, 1.84666,23.83, 0.035534, 1.85504, 23.64,0.036165).
entete_verre(sf58,918215, 1.91761,21.51, 0.042658, 1.92765, 21.34,0.043461).
entete_verre(sf59,953204, 1.95250,20.36, 0.046774, 1.96349, 20.21,0.047684).
entete_verre(sf6, 805254, 1.80518,25.43, 0.031660, 1.81265, 25.24,0.032200).
entete_verre(sf61,751275, 1.75084,27.52, 0.027287, 1.75728, 27.30,0.027735).
entete_verre(sf62,681319, 1.68134,31.94, 0.021331, 1.68639, 31.70,0.021650).
entete_verre(sf63,748277, 1.74840,27.71, 0.027004, 1.75477, 27.50,0.027445).
entete_verre(sf7, 640346, 1.63980,34.61, 0.018487, 1.64418, 34.36,0.018750).
entete_verre(sf8, 689312, 1.68893,31.18, 0.022098, 1.69416, 30.95,0.022432).
entete_verre(sf9, 654337, 1.65446,33.65, 0.019447, 1.65907, 33.41,0.019728).
entete_verre(skn18,639554, 1.63854,55.42, 0.011521, 1.64129, 55.15,0.011628).
entete_verre(sk1,610567, 1.61025,56.71, 0.010761, 1.61282, 56.43,0.010860).

entete_verre(sk10,623569, 1.62280,56.90, 0.010945, 1.62541, 56.62,0.011045).
entete_verre(sk11,564608, 1.56384,60.80, 0.009273, 1.56605, 60.55,0.009348).
entete_verre(sk12,583595, 1.58313,59.45, 0.009808, 1.58547, 59.19,0.009892).
entete_verre(sk13,592583, 1.59181,58.30, 0.010151, 1.59423, 58.02,0.010241).
entete_verre(sk14,603606, 1.60311,60.60, 0.009952, 1.60548, 60.35,0.010033).
entete_verre(sk15,623581, 1.62299,58.06, 0.010731, 1.62555, 57.79,0.010825).
entete_verre(sk16,620603, 1.62041,60.33, 0.010284, 1.62286, 60.08,0.010368).
entete_verre(sk19,613574, 1.61342,57.37, 0.010693, 1.61597, 57.08,0.010791).
entete_verre(sk2,607567, 1.60738,56.65, 0.010721, 1.60994, 56.38,0.010819).
entete_verre(sk20,560612, 1.55963,61.21, 0.009143, 1.56181, 60.96,0.009216).
entete_verre(sk3,609589, 1.60881,58.92, 0.010332, 1.61127, 58.66,0.010421).
entete_verre(sk4,613586, 1.61272,58.63, 0.010451, 1.61521, 58.35,0.010543).
entete_verre(sk5,589613, 1.58913,61.27, 0.009615, 1.59142, 61.03,0.009691).
entete_verre(sk51,621603, 1.62090,60.31, 0.010295, 1.62336, 60.02,0.010386).
entete_verre(sk52,639555, 1.63854,55.53, 0.011499, 1.64128, 55.27,0.011603).
entete_verre(sk55,620601, 1.62041,60.12, 0.010320, 1.62287, 59.89,0.010400).
entete_verre(sk6,614564, 1.61375,56.40, 0.010882, 1.61634, 56.11,0.010984).
entete_verre(sk7,607595, 1.60729,59.46, 0.010214, 1.60973, 59.19,0.010301).
entete_verre(sk8,611559, 1.61117,55.92, 0.010929, 1.61378, 55.64,0.011031).
entete_verre(sk9,614552, 1.61405,55.17, 0.011130, 1.61670, 54.88,0.011237).
entete_verre(sskn5,658509, 1.65844,50.88, 0.012940, 1.66152, 50.59,0.013076).
entete_verre(sskn8,618498, 1.61772,49.77, 0.012412, 1.62067, 49.48,0.012545).
entete_verre(ssk1,617539, 1.61720,53.91, 0.011448, 1.61993, 53.62,0.011561).
entete_verre(ssk2,622532, 1.62230,53.16, 0.011707, 1.62509, 52.87,0.011824).
entete_verre(ssk3,615512, 1.61484,51.16, 0.012018, 1.61770, 50.86,0.012144).
entete_verre(ssk4,618551, 1.61765,55.14, 0.011201, 1.62032, 54.86,0.011308).
entete_verre(ssk50,618526, 1.61795,52.61, 0.011745, 1.62075, 52.32,0.011865).
entete_verre(ssk51,604536, 1.60361,53.63, 0.011254, 1.60629, 53.35,0.011365).
entete_verre(ssk52,658509, 1.65844,50.90, 0.012936, 1.66152, 50.61,0.013070).
entete_verre(tifn5,594355, 1.59356,35.51, 0.016713, 1.59751, 35.24,0.016956).
entete_verre(tif1, 511510, 1.51118,51.01, 0.010022, 1.51356, 50.70,0.010130).
entete_verre(tif2, 533460, 1.53256,45.99, 0.011581, 1.53531, 45.67,0.011720).
entete_verre(tif3, 548422, 1.54765,42.20, 0.012976, 1.55072, 41.89,0.013146).
entete_verre(tif4, 584370, 1.58406,37.04, 0.015767, 1.58779, 36.75,0.015993).
entete_verre(tif6, 617310, 1.61650,30.97, 0.019904, 1.62118, 30.67,0.020256).
entete_verre(tik1,479587, 1.47869,58.70, 0.008155, 1.48063, 58.44,0.008224).
entete_verre(tisf1,673289, 1.67339,28.90, 0.023304, 1.67889, 28.66,0.023689).
entete_verre(ubk7,517643, 1.51680,64.29, 0.008039, 1.51872, 64.08,0.008095).
entete_verre(uk50,523604, 1.52257,60.38, 0.008654, 1.52464, 60.14,0.008723).
entete_verre(zkn7,508612, 1.50847,61.19, 0.008310, 1.51045, 60.98,0.008371).
entete_verre(zk1,533580, 1.53315,57.98, 0.009196, 1.53534, 57.71,0.009276).
entete_verre(zk5,534553, 1.53375,55.31, 0.009651, 1.53605, 55.02,0.009743).

Appendix 21: Base of Facts of the Dispersion Coefficients of Optical Glass

/*

This base of facts (coef1_disp, coef2_disp) contains the six dispersion coefficients, A0, A1, A2, A3, A4 and A5 of the optical glass.
This data is used for the calculation of the refractive index according to the dispersion furmula displayed below:

Refractive_Index is $\sqrt{(A_0+A_1*l^2+A_2*l^{-2}+A_3*l^{-4}+A_4*l^{-6}+A_5*l^{-8})}$

where:

l	: wavelength ;
$A_0, A_1, A_2, A_3, A_4, A_5$	: dispersion coefficient
sqrt	: square root.

*/

%coef1_disp(Nam1,	A0,	A1,	A2,).
coef1_disp(bafn10,	2.7293062,	-1.0356456e-2,	2.0236563e-2).
coef1_disp(bafn11,	2.7200652,	-1.0192712e-2,	1.9760925e-2).
coef1_disp(bafn6,	2.4763453,	-9.1229217e-3,	1.6804733e-2).
coef1_disp(baf12,	2.6288183,	-9.3686092e-3,	1.9694576e-2).
coef1_disp(baf13,	2.7225849,	-9.7951672e-3,	2.1079012e-2).
coef1_disp(baf3,	2.4549347,	-8.3372035e-3,	1.6841270e-2).
coef1_disp(baf4,	2.5221558,	-8.3373294e-3,	1.8614464e-2).
coef1_disp(baf5,	2.5332036,	-8.5264801e-3,	1.7164398e-2).
coef1_disp(baf50,	2.7649287,	-9.3828870e-3,	2.2737486e-2).
coef1_disp(baf51,	2.6688306,	-9.2476959e-3,	2.0660252e-2).
coef1_disp(baf52,	2.5352388,	-9.0581553e-3,	1.7360353e-2).
coef1_disp(baf53,	2.7294386,	-1.0668350e-2,	2.0239807e-2).
coef1_disp(baf54,	2.7195755,	-1.0322464e-2,	2.0098244e-2).
coef1_disp(baf8,	2.5817259,	-8.6743047e-3,	1.8611342e-2).
coef1_disp(baf9,	2.6453542,	-9.6193048e-3,	1.8369094e-2).
coef1_disp(bak1,	2.4333007,	-8.4931353e-3,	1.3893512e-2).
coef1_disp(bak2,	2.3370143,	-8.6389345e-3,	1.2265051e-2).
coef1_disp(bak4,	2.4218304,	-9.2167103e-3,	1.3821685e-2).
coef1_disp(bak5,	2.3861698,	-8.4606642e-3,	1.3233029e-2).
coef1_disp(bak50,	2.4195494,	-9.5312125e-3,	1.3775289e-2).
coef1_disp(bak6,	2.4394521,	-9.5754011e-3,	1.3786016e-2).
coef1_disp(balf3,	2.4251277,	-7.4347248e-3,	1.5487534e-2).
coef1_disp(balf4,	2.4528366,	-9.2047678e-3,	1.4552794e-2).
coef1_disp(balf5,	2.3544612,	-7.9053990e-3,	1.3971868e-2).
coef1_disp(balf50,	2.4792852,	-9.2013619e-3,	1.5615134e-2).
coef1_disp(balf51,	2.4334741,	-8.5343207e-3,	1.5241024e-2).
coef1_disp(balf6,	2.4802388,	-8.6000215e-3,	1.5624147e-2).
coef1_disp(balf8,	2.3718251,	-8.3840363e-3,	1.4379250e-2).
coef1_disp(balkn3,	2.2738525,	-8.8519024e-3,	1.1511571e-2).

```
coef1_disp(balk1,   2.2966923,  -8.2975549e-3,   1.1907234e-2).
coef1_disp(basf1,   2.5778903,  -8.3279108e-3,   2.1841868e-2).
coef1_disp(basf10,  2.6531250,  -8.1388553e-3,   2.2995643e-2).
coef1_disp(basf12,  2.7172821,  -9.5055782e-3,   2.3326873e-2).
coef1_disp(basf13,  2.8055334,  -1.0583600e-2,   2.4780423e-2).
coef1_disp(basf14,  2.8040560,  -1.0121606e-2,   2.7277876e-2).
coef1_disp(basf2,   2.6926115,  -8.7432478e-3,   2.5176098e-2).
coef1_disp(basf5,   2.5131519,  -8.5791364e-3,   1.9014140e-2).
coef1_disp(basf50,  2.8430385,  -1.2392675e-2,   2.6677418e-2).
coef1_disp(basf51,  2.8903514,  -1.1808260e-2,   2.6137219e-2).
coef1_disp(basf52,  2.8247094,  -1.3066162e-2,   2.3808337e-2).
coef1_disp(basf54,  2.9179112,  -1.1371629e-2,   2.9697508e-2).
coef1_disp(basf55,  2.8080853,  -1.3076515e-2,   2.4961324e-2).
coef1_disp(basf56,  2.6716908,  -8.8722613e-3,   2.4052884e-2).
coef1_disp(basf57,  2.6629169,  -9.8650297e-3,   2.1537161e-2).
coef1_disp(basf6,   2.7138041,  -9.4656841e-3,   2.2128109e-2).
coef1_disp(basf64,  2.8178914,  -1.1398749e-2,   2.5203102e-2).
coef1_disp(bk1,     2.2513742,  -9.3254015e-3,   1.0539647e-2).
coef1_disp(bk10,    2.2177191,  -1.0248661e-2,   9.6627662e-3).
coef1_disp(bk3,     2.2184519,  -1.0539086e-2,   9.9115699e-3).
coef1_disp(bk6,     2.3125058,  -9.5398792e-3,   1.1668749e-2).
coef1_disp(bk7,     2.2718929,  -1.0108077e-2,   1.0592509e-2).
coef1_disp(bk8,     2.2804948,  -9.4190530e-3,   1.1349888e-2).
coef1_disp(fk1,     2.1392463,  -9.2409374e-3,   8.7449768e-3).
coef1_disp(fk3,     2.1202375,  -8.6868569e-3,   9.1401191e-3).
coef1_disp(fk5,     2.1887621,  -9.5572007e-3,   8.9915232e-3).
coef1_disp(fk51,    2.1883307,  -5.3658786e-3,   7.7436552e-3).
coef1_disp(fk52,    2.1858571,  -5.2014619e-3,   8.1074888e-3).
coef1_disp(fk54,    2.0478023,  -4.8685134e-3,   6.2781975e-3).
coef1_disp(fn11,    2.5610417,  -1.1488430e-2,   2.1189476e-2).
coef1_disp(f1,      2.5713922,  -8.4543797e-3,   2.3611565e-2).
coef1_disp(f13,     2.5614812,  -8.5680987e-3,   2.3074904e-2).
coef1_disp(f14,     2.5012990,  -8.7770607e-3,   2.0833999e-2).
coef1_disp(f15,     2.5138970,  -8.8848319e-3,   2.1048670e-2).
coef1_disp(f2,      2.5554063,  -8.8746150e-3,   2.2494787e-2).
coef1_disp(f3,      2.5342719,  -8.6565414e-3,   2.2160522e-2).
coef1_disp(f4,      2.5446900,  -8.5925662e-3,   2.2583116e-2).
coef1_disp(f5,      2.5069744,  -8.6678569e-3,   2.1105291e-2).
coef1_disp(f6,      2.6038221,  -8.8224838e-3,   2.3872164e-2).
coef1_disp(f7,      2.5700098,  -8.4594488e-3,   2.3243109e-2).
coef1_disp(f8,      2.4847853,  -8.7690408e-3,   2.0213087e-2).
coef1_disp(f9,      2.5599175,  -9.3903357e-3,   2.1745487e-2).
coef1_disp(kf1,     2.3324325,  -8.5563978e-3,   1.4012802e-2).
coef1_disp(kf3,     2.2591522,  -8.6810568e-3,   1.2074479e-2).
coef1_disp(kf50,    2.3045807,  -8.8382960e-3,   1.3484730e-2).
coef1_disp(kf6,     2.2651845,  -8.4401006e-3,   1.3185124e-2).
coef1_disp(kf9,     2.2824396,  -8.5960144e-3,   1.3442645e-2).
coef1_disp(kzfn1,   2.3635655,  -9.6497023e-3,   1.4907410e-2).
coef1_disp(kzfn2,   2.3016068,  -1.0463801e-2,   1.3289732e-2).
coef1_disp(kzfsn2,  2.3912843,  -1.3244644e-2,   1.3524263e-2).
coef1_disp(kzfsn4,  2.5493446,  -1.3234586e-2,   1.8586165e-2).
coef1_disp(kzfsn5,  2.6699840,  -1.3941585e-2,   2.2384056e-2).
coef1_disp(kzfsn7,  2.7497306,  -1.3708753e-2,   2.4819910e-2).
coef1_disp(kzfsn9,  2.5066403,  -1.3178716e-2,   1.7087006e-2).
coef1_disp(kzfs1,   2.5495109,  -1.4614529e-2,   1.8246082e-2).
coef1_disp(kzfs6,   2.4877262,  -1.2894400e-2,   1.6567331e-2).
coef1_disp(kzfs8,   2.8717595,  -1.3501333e-2,   2.8964424e-2).
coef1_disp(kzf6,    2.2934044,  -1.0346122e-2,   1.3319863e-2).
```

coef1_disp(k10, 2.2209762, -8.3295635e-3, 1.1963114e-2).
coef1_disp(k11, 2.2211157, -9.1442577e-3, 1.0670316e-2).
coef1_disp(k3, 2.2727603, -8.7840747e-3, 1.1325842e-2).
coef1_disp(k4, 2.2734025, -8.8768403e-3, 1.2161748e-2).
coef1_disp(k5, 2.2850299, -8.6010725e-3, 1.1806783e-2).
coef1_disp(k50, 2.2861092, -9.1773008e-3, 1.1562643e-2).
coef1_disp(k51, 2.2351459, -9.9823499e-3, 1.1161681e-2).
coef1_disp(k7, 2.2520059, -8.4630818e-3, 1.1351376e-2).
coef1_disp(lafn10, 3.1052418, -1.5458266e-2, 2.6987363e-2).
coef1_disp(lafn21, 3.1239556, -1.5147423e-2, 2.5646258e-2).
coef1_disp(lafn23, 2.7925282, -9.7181630e-3, 2.0756437e-2).
coef1_disp(lafn24, 3.0188196, -1.5176009e-2, 2.4040260e-2).
coef1_disp(lafn28, 3.0750514, -1.4743725e-2, 2.4004375e-2).
coef1_disp(lafn7, 2.9676706, -1.3465547e-2, 3.0562239e-2).
coef1_disp(lafn8, 2.9343747, -1.2591530e-2, 2.5612524e-2).
coef1_disp(laf11a, 2.9829266, -1.2307185e-2, 3.3050301e-2).
coef1_disp(laf13, 3.0604516, -1.2242361e-2, 3.0441692e-2).
coef1_disp(laf2, 2.9673787, -1.0978767e-2, 2.5088607e-2).
coef1_disp(laf20, 2.7698488, -1.0003512e-2, 2.0882171e-2).
coef1_disp(laf22a, 3.0813540, -1.2431075e-2, 3.0277466e-2).
coef1_disp(laf25, 3.0997712, -1.4428294e-2, 2.7899374e-2).
coef1_disp(laf26, 2.9676367, -1.3599942e-2, 2.7026861e-2).
coef1_disp(laf3, 2.8832223, -1.1371082e-2, 2.2149870e-2).
coef1_disp(laf9, 3.0998216, -1.2191689e-2, 3.7147724e-2).
coef1_disp(lakl12, 2.7641980, -1.2896018e-2, 1.8254336e-2).
coef1_disp(lakl21, 2.6496868, -1.4180769e-2, 1.5173226e-2).
coef1_disp(lakn12, 2.7627145, -1.0444563e-2, 1.8601838e-2).
coef1_disp(lakn13, 2.8115119, -1.0386717e-2, 1.9734379e-2).
coef1_disp(lakn14, 2.8272796, -1.4543591e-2, 1.8607179e-2).
coef1_disp(lakn22, 2.6775139, -1.0439699e-2, 1.7312971e-2).
coef1_disp(lakn6, 2.6520918, -1.0634343e-2, 1.6314457e-2).
coef1_disp(lakn7, 2.6820255, -1.1431524e-2, 1.6434096e-2).
coef1_disp(lak10, 2.8984614, -1.4857039e-2, 2.0985037e-2).
coef1_disp(lak11, 2.7031311, -1.2225054e-2, 1.6715221e-2).
coef1_disp(lak16a, 2.9446521, -1.4832620e-2, 2.1265867e-2).
coef1_disp(lak20, 2.8088805, -9.7299067e-3, 2.0547293e-2).
coef1_disp(lak21, 2.6478736, -1.1574789e-2, 1.5782386e-2).
coef1_disp(lak23, 2.7367941, -1.1905799e-2, 1.7099759e-2).
coef1_disp(lak28, 2.9790542, -1.4645416e-2, 2.2376581e-2).
coef1_disp(lak31, 2.8283850, -1.4963716e-2, 1.8210258e-2).
coef1_disp(lak33, 3.0136285, -1.4856543e-2, 2.2361843e-2).
coef1_disp(lak8, 2.8791177, -1.4887202e-2, 1.9662614e-2).
coef1_disp(lak9, 2.8081456, -1.4266626e-2, 1.8008161e-2).
coef1_disp(lasfn15, 3.4174343, -1.5504887e-2, 3.6536079e-2).
coef1_disp(lasfn18, 3.5278149, -1.7049614e-2, 4.2895039e-2).
coef1_disp(lasfn30, 3.1731314, -1.4823958e-2, 2.6862762e-2).
coef1_disp(lasfn31, 3.4322240, -1.2790848e-2, 3.5133497e-2).
coef1_disp(lasfn9, 3.2994326, -1.1680436e-2, 4.0133103e-2).
coef1_disp(lasf11, 3.1634123, -1.4427452e-2, 2.8384126e-2).
coef1_disp(lasf13, 3.3331229, -1.3161988e-2, 3.5541296e-2).
coef1_disp(lasf3, 3.1796761, -1.5760989e-2, 3.0065588e-2).
coef1_disp(lasf32, 3.1385520, -1.5378166e-2, 3.4981894e-2).
coef1_disp(lasf33, 3.1542132, -1.1481374e-2, 3.4684427e-2).
coef1_disp(lasf8, 3.1513867, -1.2595765e-2, 3.6908088e-2).
coef1_disp(lf1, 2.4217647, -8.5906079e-3, 1.7651245e-2).
coef1_disp(lf2, 2.4682847, -8.6227186e-3, 1.9045004e-2).
coef1_disp(lf3, 2.4479180, -8.5152303e-3, 1.8685135e-2).
coef1_disp(lf4, 2.4364980, -8.6900852e-3, 1.8427867e-2).

```
coef1_disp(lf5,     2.4441760,  -8.3059695e-3,   1.9000697e-2).
coef1_disp(lf6,     2.4043241,  -8.4381264e-3,   1.7623826e-2).
coef1_disp(lf7,     2.4259431,  -8.6137761e-3,   1.8353273e-2).
coef1_disp(lf8,     2.3966602,  -8.2907298e-3,   1.7199116e-2).
coef1_disp(lgsk2,   2.4750760,  -5.4304528e-3,   1.3893210e-2).
coef1_disp(llf1,    2.3505162,  -8.5306451e-3,   1.5750853e-2).
coef1_disp(llf2,    2.3299214,  -8.5444433e-3,   1.5031394e-2).
coef1_disp(llf3,    2.3880339,  -9.1010943e-3,   1.5789160e-2).
coef1_disp(llf4,    2.3895058,  -8.6358875e-3,   1.6448035e-2).
coef1_disp(llf6,    2.3047007,  -8.5161517e-3,   1.4319368e-2).
coef1_disp(llf7,    2.3525384,  -8.8030406e-3,   1.5703390e-2).
coef1_disp(pk1,     2.2347576,  -1.0003049e-2,   1.0000850e-2).
coef1_disp(pk2,     2.2770533,  -1.0532010e-2,   1.0188354e-2).
coef1_disp(pk3,     2.2977022,  -1.0331615e-2,   1.0757237e-2).
coef1_disp(pk50,    2.2851698,  -9.5978829e-3,   9.9950907e-3).
coef1_disp(pk51,    2.3095538,  -5.7099342e-3,   9.5910979e-3).
coef1_disp(psk2,    2.4266341,  -1.0593625e-2,   1.2590958e-2).
coef1_disp(psk3,    2.3768193,  -1.0146514e-2,   1.2167148e-2).
coef1_disp(psk50,   2.3946348,  -9.6851585e-3,   1.1457938e-2).
coef1_disp(psk52,   2.5342699,  -1.0342368e-2,   1.2636367e-2).
coef1_disp(psk53,   2.5852417,  -9.4290947e-3,   1.4074470e-2).
coef1_disp(sf156,   3.0549693,  -1.2858387e-2,   4.0081007e-2).
coef1_disp(sf16,    3.1206868,  -1.3279387e-2,   4.1905574e-2).
coef1_disp(sfn64,   2.8125953,  -1.1916007e-2,   3.1041260e-2).
coef1_disp(sf1,     2.8458754,  -9.8260548e-3,   3.2192965e-2).
coef1_disp(sf10,    2.8784725,  -1.0565453e-2,   3.3279420e-2).
coef1_disp(sf11,    3.0539614,  -1.1580432e-2,   3.9199816e-2).
coef1_disp(sf12,    2.6385867,  -9.4879851e-3,   2.4984154e-2).
coef1_disp(sf13,    2.9177579,  -1.1483287e-2,   3.3825845e-2).
coef1_disp(sf14,    2.9826955,  -1.1720091e-2,   3.5994978e-2).
coef1_disp(sf15,    2.7898291,  -1.0260526e-2,   2.9707118e-2).
coef1_disp(sf16,    2.6312546,  -8.8042107e-3,   2.5282256e-2).
coef1_disp(sf17,    2.6432169,  -8.8860441e-3,   2.5631865e-2).
coef1_disp(sf18,    2.8577802,  -9.4889259e-3,   3.3023767e-2).
coef1_disp(sf19,    2.6942327,  -9.4285222e-3,   2.6806270e-2).
coef1_disp(sf2,     2.6361862,  -9.0087536e-3,   2.5179779e-2).
coef1_disp(sf3,     2.9144630,  -9.7237692e-3,   3.4855288e-2).
coef1_disp(sf4,     2.9605971,  -9.3495013e-3,   3.7404384e-2).
coef1_disp(sf5,     2.7105646,  -9.1211994e-3,   2.7760538e-2).
coef1_disp(sf50,    2.6574144,  -9.5545655e-3,   2.5081498e-2).
coef1_disp(sf51,    2.6785087,  -1.1867542e-2,   2.3037272e-2).
coef1_disp(sf52,    2.7576175,  -9.7069196e-3,   2.9008907e-2).
coef1_disp(sf53,    2.8788944,  -1.0038296e-2,   3.3232802e-2).
coef1_disp(sf54,    2.9157333,  -9.4607589e-3,   3.5929046e-2).
coef1_disp(sf55,    2.9810972,  -1.0106105e-2,   3.7725376e-2).
coef1_disp(sf56,    3.0510040,  -9.4149325e-3,   4.1729775e-2).
coef1_disp(sf57,    3.2578469,  -1.4544868e-2,   4.2028938e-2).
coef1_disp(sf58,    3.4782654,  -1.0766912e-2,   5.8676907e-2).
coef1_disp(sf59,    3.6049456,  -1.7579501e-2,   5.4777275e-2).
coef1_disp(sf6,     3.1195007,  -1.0902580e-2,   4.1330651e-2).
coef1_disp(sf61,    2.9458531,  -9.2723542e-3,   3.7513268e-2).
coef1_disp(sf62,    2.7369014,  -9.1210145e-3,   2.8718145e-2).
coef1_disp(sf63,    2.9393620,  -9.6548019e-3,   3.6674842e-2).
coef1_disp(sf7,     2.6129703,  -8.9265027e-3,   2.4553061e-2).
coef1_disp(sf8,     2.7594675,  -9.3696887e-3,   2.9328240e-2).
coef1_disp(sf9,     2.6563905,  -8.9916333e-3,   2.6099377e-2).
coef1_disp(skn18,   2.6376216,  -1.0518989e-2,   1.6589824e-2).
coef1_disp(sk1,     2.5491516,  -9.2996196e-3,   1.5424606e-2).
```

coef1_disp(sk10,	2.5881711,	-9.3042169e-3,	1.6075770e-2).
coef1_disp(sk11,	2.4098938,	-9.5183518e-3,	1.2805486e-2).
coef1_disp(sk12,	2.4674397,	-9.3595371e-3,	1.3921727e-2).
coef1_disp(sk13,	2.4934144,	-9.2031432e-3,	1.4259619e-2).
coef1_disp(sk14,	2.5300639,	-1.0126279e-2,	1.4483568e-2).
coef1_disp(sk15,	2.5901210,	-9.9415459e-3,	1.5735505e-2).
coef1_disp(sk16,	2.5846319,	-1.1059422e-2,	1.4856282e-2).
coef1_disp(sk19,	2.5594250,	-9.3541373e-3,	1.5497881e-2).
coef1_disp(sk2,	2.5395065,	-8.8026103e-3,	1.5691213e-2).
coef1_disp(sk20,	2.3975952,	-9.6663991e-3,	1.2508188e-2).
coef1_disp(sk3,	2.5470242,	-1.0118918e-2,	1.4639317e-2).
coef1_disp(sk4,	2.5585228,	-9.8824951e-3,	1.5151820e-2).
coef1_disp(sk5,	2.4876635,	-1.0442251e-2,	1.3736058e-2).
coef1_disp(sk51,	2.5855045,	-9.6112749e-3,	1.4803167e-2).
coef1_disp(sk52,	2.6378898,	-1.1047170e-2,	1.6634360e-2).
coef1_disp(sk55,	2.5852191,	-1.2089769e-2,	1.4721771e-2).
coef1_disp(sk6,	2.5596842,	-9.1217460e-3,	1.5610313e-2).
coef1_disp(sk7,	2.5420185,	-9.7351561e-3,	1.4884507e-2).
coef1_disp(sk8,	2.5516208,	-9.3738374e-3,	1.5451158e-2).
coef1_disp(sk9,	2.5592131,	-9.0020732e-3,	1.6280492e-2).
coef1_disp(sskn5,	2.6971175,	-1.0516627e-2,	1.8053262e-2).
coef1_disp(sskn8,	2.5670599,	-9.8663081e-3,	1.7078368e-2).
coef1_disp(ssk1,	2.5680835,	-9.0625067e-3,	1.6608680e-2).
coef1_disp(ssk2,	2.5832910,	-9.0240848e-3,	1.7028671e-2).
coef1_disp(ssk3,	2.5587299,	-9.0062517e-3,	1.6793145e-2).
coef1_disp(ssk4,	2.5707849,	-9.2577764e-3,	1.6170751e-2).
coef1_disp(ssk50,	2.5696487,	-9.0907229e-3,	1.6655170e-2).
coef1_disp(ssk51,	2.5256456,	-8.8479177e-3,	1.6074734e-2).
coef1_disp(ssk52,	2.6963923,	-1.0347971e-2,	1.8618508e-2).
coef1_disp(tifn5,	2.4757539,	-1.0282285e-2,	2.0102692e-2).
coef1_disp(tif1.	2.2473124,	-8.9044058e-3,	1.2493525e-2).
coef1_disp(tif2,	2.3062438,	-9.3513887e-3,	1.4218213e-2).
coef1_disp(tif3,	2.3488389,	-1.0357251e-2,	1.4639213e-2).
coef1_disp(tif4,	2.4498259,	-1.0128610e-2,	1.8753684e-2).
coef1_disp(tif6,	2.5388379,	-1.0552711e-2,	2.2004734e-2).
coef1_disp(tik1,	2.1573978,	-8.4004189e-3,	1.0457582e-2).
coef1_disp(tisf1,	2.7091574,	-1.2750242e-2,	2.7097845e-2).
coef1_disp(ubk7,	2.2715621,	-9.8571566e-3,	1.0808515e-2).
coef1_disp(uk50,	2.2858030,	-8.9733347e-3,	1.1798712e-2).
coef1_disp(zkn7,	2.2447173,	-9.4165837e-3,	1.1533424e-2).
coef1_disp(zk1,	2.3157951,	-8.7493905e-3,	1.2329645e-2).
coef1_disp(zk5,	2.3151352,	-7.7924593e-3,	1.3029385e-2).

%coef2_disp(Nam1,	A3,	A4,	A5).
coef2_disp(bafn10,	5.8969718e-4,	-2.0288303e-5,	2.8521978e-6).
coef2_disp(bafn11,	5.1935441e-4,	-1.4074936e-5,	2.2229689e-6).
coef2_disp(bafn6,	3.4558633e-4,	2.5330454e-6,	1.1900884e-6).
coef2_disp(baf12,	5.8312064e-4,	-1.7556026e-5,	2.9471099e-6).
coef2_disp(baf13,	6.2762461e-4,	-1.8947499e-5,	3.2030090e-6).
coef2_disp(baf3,	5.0168527e-4,	-1.4413749e-5,	2.0771351e-6).
coef2_disp(baf4,	6.0603872e-4,	-1.8920312e-5,	2.7375194e-6).
coef2_disp(baf5,	4.1464804e-4,	-7.2876766e-6,	1.4129143e-6).
coef2_disp(baf50,	4.4606478e-4,	6.1200859e-6,	1.7483664e-6).
coef2_disp(baf51,	5.2893109e-4,	-7.0867424e-6,	2.1797509e-6).
coef2_disp(baf52,	6.9794119e-4,	-3.8819033e-5,	3.5656310e-6).
coef2_disp(baf53,	5.8138876e-4,	-1.8249855e-5,	2.5746426e-6).
coef2_disp(baf54,	4.3867795e-4,	-2.3936865e-6,	1.5014011e-6).

coef2_disp(baf8, 4.6995284e-4, -7.3102793e-6, 1.7799691e-6).
coef2_disp(baf9, 6.9291183e-4, -3.7950550e-5, 3.2253161e-6).
coef2_disp(bak1, 2.6798268e-4, -6.1946101e-6, 6.2209005e-7).
coef2_disp(bak2, 2.4273540e-4, -6.3916988e-6, 5.1229141e-7).
coef2_disp(bak4, 3.2955714e-4, -1.2153641e-5, 1.0333767e-6).
coef2_disp(bak5, 2.0949226e-4, -5.5580737e-7, 2.9765122e-7).
coef2_disp(bak50, 1.8153631e-4, 4.6618943e-6, 2.0448919e-8).
coef2_disp(bak6, 3.7095536e-4, -1.7661400e-5, 1.3821042e-6).
coef2_disp(balf3, 1.6375720e-4, 1.5027176e-5, -1.7953510e-7).
coef2_disp(balf4, 4.3046688e-4, -2.0489836e-5, 1.5924415e-6).
coef2_disp(balf5, 2.5936745e-4, -1.5064670e-7, 6.0410634e-7).
coef2_disp(balf50, 4.3452084e-4, -1.6552131e-5, 1.6658805e-6).
coef2_disp(balf51, 3.0677273e-4, -1.9515136e-6, 8.5689514e-7).
coef2_disp(balf6, 3.0040260e-4, -2.2119056e-6, 8.0013234e-7).
coef2_disp(balf8, 3.9742058e-4, -1.3041618e-5, 1.4386564e-6).
coef2_disp(balkn3, 2.0622098e-4, -3.9599217e-6, 4.4844819e-7).
coef2_disp(balk1, 1.9908305e-4, -2.0306838e-6, 3.1429703e-7).
coef2_disp(basf1, 6.8618153e-4, -8.1530554e-6, 3.3013474e-6).
coef2_disp(basf10, 7.3535957e-4, -1.3407390e-5, 3.6962325e-6).
coef2_disp(basf12, 9.3379945e-4, -3.9996373e-5, 5.6638955e-6).
coef2_disp(basf13, 1.0753528e-3, -5.4046905e-5, 7.2739552e-6).
coef2_disp(basf14, 1.1920911e-3, -5.2304151e-5, 8.6551492e-6).
coef2_disp(basf2, 9.7702423e-4, -2.8680692e-5, 5.7931396e-6).
coef2_disp(basf5, 6.3221574e-4, -1.7533060e-5, 2.8739893e-6).
coef2_disp(basf50, 1.1036516e-3, -3.8659027e-5, 6.0798447e-6).
coef2_disp(basf51, 1.2219537e-3, -6.0256399e-5, 6.6193721e-6).
coef2_disp(basf52, 8.9302284e-4, -3.3448747e-5, 4.4506802e-6).
coef2_disp(basf54, 2.0284464e-3, -1.4304533e-4, 1.5393060e-5).
coef2_disp(basf55, 1.9412734e-3, -1.5776742e-4, 1.4562956e-5).
coef2_disp(basf56, 9.9655176e-4, -3.8518520e-5, 5.9357564e-6).
coef2_disp(basf57, 6.6532162e-4, -1.4148425e-5, 3.3387062e-6).
coef2_disp(basf6, 7.7376428e-4, -2.8857539e-5, 4.2797530e-6).
coef2_disp(basf64, 7.7721356e-4, -1.6483738e-5, 4.5370859e-6).
coef2_disp(bk1, 2.2595365e-4, -1.0729053e-5, 7.2832778e-7).
coef2_disp(bk10, 1.6782840e-4, -5.5328684e-6, 3.4747416e-7).
coef2_disp(bk3, 1.8512559e-4, -7.0180588e-6, 4.2385691e-7).
coef2_disp(bk6, 1.5598074e-4, 1.1623640e-6, 1.2318050e-7).
coef2_disp(bk7, 2.0816965e-4, -7.6472538e-6, 4.9240991e-7).
coef2_disp(bk8, 5.6741756e-5, 1.2149716e-5, -4.2771477e-7).
coef2_disp(fk1, 2.1463624e-4, -1.2671989e-5, 6.8154684e-7).
coef2_disp(fk3, 1.2613174e-4, -8.2960642e-7, 1.3180567e-7).
coef2_disp(fk5, 1.4560516e-4, -5.2843067e-6, 3.4588010e-7).
coef2_disp(fk51, 1.3129343e-4, -7.4179516e-6, 4.5815122e-7).
coef2_disp(fk52, 1.0085829e-4, -2.2241781e-6, 1.9197508e-7).
coef2_disp(fk54, 8.8845082e-5, -4.1267464e-6, 2.4392914e-7).
coef2_disp(fn11, 1.2504217e-3, -8.9130731e-5, 1.0480983e-5).
coef2_disp(f1, 7.8346482e-4, -1.0331153e-5, 4.2751979e-6).
coef2_disp(f13, 8.1271690e-4, -1.6852152e-5, 4.5166052e-6).
coef2_disp(f14, 7.1365720e-4, -1.5356429e-5, 3.5458118e-6).
coef2_disp(f15, 7.7800910e-4, -1.9727183e-5, 3.8039205e-6).
coef2_disp(f2, 8.6924972e-4, -2.4011704e-5, 4.5365169e-6).
coef2_disp(f3, 7.1828923e-4, -8.3757309e-6, 3.5549137e-6).
coef2_disp(f4, 7.3789911e-4, -9.5060664e-6, 3.8257675e-6).
coef2_disp(f5, 7.0608713e-4, -1.3731195e-5, 3.5479149e-6).
coef2_disp(f6, 9.6357866e-4, -2.9653611e-5, 5.4103450e-6).
coef2_disp(f7, 9.1209596e-4, -2.5955976e-5, 5.1185012e-6).
coef2_disp(f8, 6.3621526e-4, -8.3230722e-6, 2.9472755e-6).
coef2_disp(f9, 7.5168710e-4, -1.6255461e-5, 4.0551837e-6).

coef2_disp(kf1, 3.4494190e-4, -7.6053118e-6, 1.2660303e-6).
coef2_disp(kf3, 3.6515757e-4, -1.9089812e-5, 1.4345201e-6).
coef2_disp(kf50, 3.8601862e-4, -1.3379593e-5, 1.3359542e-6).
coef2_disp(kf6, 2.3958981e-4, 2.0310603e-6, 6.0813527e-7).
coef2_disp(kf9, 2.7803535e-4, -4.9998960e-7, 7.7105911e-7).
coef2_disp(kzfn1, 3.0149478e-4, 2.8868678e-6, 6.7799805e-7).
coef2_disp(kzfn2, 3.3438756e-4, -7.1799896e-6, 9.1964556e-7).
coef2_disp(kzfsn2, 3.3475551e-4, -1.1613973e-5, 1.0939788e-6).
coef2_disp(kzfsn4, 5.4759655e-4, -1.1717987e-5, 2.0042905e-6).
coef2_disp(kzfsn5, 7.4780873e-4, -1.7341165e-5, 3.4427318e-6).
coef2_disp(kzfsn7, 9.6006315e-4, -3.3307832e-5, 5.8217650e-6).
coef2_disp(kzfsn9, 4.7129902e-4, -1.1358998e-5, 1.6946777e-6).
coef2_disp(kzfs1, 6.0860367e-4, -2.0613741e-5, 2.3500631e-6).
coef2_disp(kzfs6, 3.6347288e-4, -1.5994850e-6, 1.1023685e-6).
coef2_disp(kzfs8, 1.0970513e-3, -2.4165632e-5, 5.6994640e-6).
coef2_disp(kzf6, 3.4833226e-4, -9.9354090e-6, 1.1227905e-6).
coef2_disp(k10, 1.3879745e-4, 7.8358808e-6, 8.4344620e-8).
coef2_disp(k11, 1.8263096e-4, -1.4105398e-6, 3.3096328e-7).
coef2_disp(k3, 3.4828095e-4, -2.1976876e-5, 1.3941592e-6).
coef2_disp(k4, 1.8966377e-4, 1.4584031e-6, 2.7687697e-7).
coef2_disp(k5, 2.0765657e-4, -2.1314913e-6, 3.2131234e-7).
coef2_disp(k50, 2.2550894e-4, -5.4163445e-6, 4.3498997e-7).
coef2_disp(k51, 1.8637660e-4, -2.5979987e-6, 4.9365998e-7).
coef2_disp(k7, 1.8421156e-4, -2.0450534e-6, 3.4814749e-7).
coef2_disp(lafn10, 7.4367535e-4, -1.2921245e-5, 2.7404039e-6).
coef2_disp(lafn21, 6.1250913e-4, -9.1760698e-6, 1.7034755e-6).
coef2_disp(lafn23, 3.9610891e-4, 1.9565557e-6, 1.0029477e-6).
coef2_disp(lafn24, 5.3038456e-4, -1.3346974e-6, 1.2064345e-6).
coef2_disp(lafn28, 5.5949356e-4, -1.0007074e-5, 1.1559175e-6).
coef2_disp(lafn7, 1.1147255e-3, -2.3979989e-5, 6.2175926e-6).
coef2_disp(lafn8, 8.2846469e-4, -2.2108582e-5, 3.9458540e-6).
coef2_disp(laf11a, 1.5748398e-3, -6.3407414e-5, 1.0044908e-5).
coef2_disp(laf13, 9.6185256e-4, -1.6231095e-5, 4.9181335e-6).
coef2_disp(laf2, 6.3171596e-4, -7.5645417e-6, 2.3202213e-6).
coef2_disp(laf20, 4.5883886e-4, -5.0275096e-6, 1.6410791e-6).
coef2_disp(laf22a, 1.3030739e-3, -6.2689126e-5, 8.5989688e-6).
coef2_disp(laf25, 1.0345598e-3, -4.3008448e-5, 5.4917491e-6).
coef2_disp(laf26, 8.8284489e-4, -2.2042475e-5, 4.5665081e-6).
coef2_disp(laf3, 5.7232017e-4, -1.3181988e-5, 2.0252639e-6).
coef2_disp(laf9, 2.5559645e-3, -1.5721379e-4, 1.8886019e-5).
coef2_disp(lakl12, 3.2407686e-4, -8.4118624e-7, 3.7652891e-7).
coef2_disp(lakl21, 3.1464516e-4, -1.0814561e-5, 7.9828892e-7).
coef2_disp(lakn12, 2.5893372e-4, 7.6556580e-6, -3.7061383e-8).
coef2_disp(lakn13, 3.2856524e-4, 3.2051557e-6, 3.0925067e-7).
coef2_disp(lakn14, 3.6867908e-4, -6.9826459e-6, 7.0273582e-7).
coef2_disp(lakn22, 2.3622943e-4, 8.1681533e-6, -1.3539324e-7).
coef2_disp(lakn6, 2.4908012e-4, 7.0505584e-7, 2.7087570e-7).
coef2_disp(lakn7, 2.4373786e-4, 1.7117573e-6, 2.1575860e-7).
coef2_disp(lak10, 5.4506921e-4, -1.7297314e-5, 1.7993601e-6).
coef2_disp(lak11, 3.2476320e-4, -5.1327015e-6, 5.0461740e-7).
coef2_disp(lak16a, 4.8892724e-4, -1.1811627e-5, 1.1363503e-6).
coef2_disp(lak20, 3.1872622e-4, 6.6319967e-6, 3.1014063e-7).
coef2_disp(lak21, 1.7291765e-4, 7.7882807e-6, -1.1063827e-7).
coef2_disp(lak23, 3.4286906e-4, -8.1287792e-6, 7.1596070e-7).
coef2_disp(lak28, 4.4238875e-4, -1.6906632e-6, 7.3784132e-7).
coef2_disp(lak31, 3.6816151e-4, -8.6283328e-6, 7.5649349e-7).
coef2_disp(lak33, 3.5357770e-4, 7.8428250e-6, 4.6780278e-8).
coef2_disp(lak8, 4.3841844e-4, -1.2951193e-5, 1.1580631e-6).

coef2_disp(lak9,	5.6748764e-4,	-3.2899281e-5,	2.0438076e-6).
coef2_disp(lasfn15,	1.0424971e-3,	-1.8067825e-6,	3.8393637e-6).
coef2_disp(lasfn18,	1.9248178e-3,	-7.5388918e-5,	1.3032008e-5).
coef2_disp(lasfn30,	6.9283981e-4,	-1.2265479e-5,	1.7263354e-6).
coef2_disp(lasfn31,	8.4763112e-4,	4.5551843e-6,	1.6550517e-6).
coef2_disp(lasfn9,	1.3263988e-3,	4.7438783e-6,	7.8507188e-6).
coef2_disp(lasf11,	6.0299606e-4,	6.6029672e-6,	1.4484996e-6).
coef2_disp(lasf13,	1.4177373e-3,	-5.2019105e-5,	7.2348123e-6).
coef2_disp(lasf3,	9.1823007e-4,	-1.8296849e-5,	4.4652393e-6).
coef2_disp(lasf32,	2.4848158e-3,	-1.7333919e-4,	1.9596905e-5).
coef2_disp(lasf33,	1.2974344e-3,	-2.7569572e-5,	7.8461878e-6).
coef2_disp(lasf8,	1.5820869e-3,	-5.1101128e-5,	1.0662006e-5).
coef2_disp(lf1,	5.8415074e-4,	-1.6884537e-5,	2.7259662e-6).
coef2_disp(lf2,	6.2971872e-4,	-1.5592113e-5,	3.0502534e-6).
coef2_disp(lf3,	4.6787852e-4,	1.3422259e-6,	2.0021368e-6).
coef2_disp(lf4,	5.6173085e-4,	-1.1219778e-5,	2.7012778e-6).
coef2_disp(lf5,	5.4129697e-4,	-4.1973115e-6,	2.3742897e-6).
coef2_disp(lf6,	4.7507875e-4,	-4.0602970e-6,	2.0556661e-6).
coef2_disp(lf7,	5.3951938e-4,	-6.9790993e-6,	2.4087417e-6).
coef2_disp(lf8,	4.5298946e-4,	-3.8995694e-6,	1.9005178e-6).
coef2_disp(lgsk2,	2.2990560e-4,	-1.6868474e-6,	4.3959703e-7).
coef2_disp(llf1,	4.2811388e-4,	-6.9875718e-6,	1.7175517e-6).
coef2_disp(llf2,	4.0265729e-4,	-7.6000030e-6,	1.5659130e-6).
coef2_disp(llf3,	4.0516750e-4,	-5.4507043e-6,	1.5992768e-6).
coef2_disp(llf4,	4.5166611e-4,	-7.4107160e-6,	1.8684107e-6).
coef2_disp(llf6,	3.5400942e-4,	-5.6239322e-6,	1.3019147e-6).
coef2_disp(llf7,	4.8840166e-4,	-1.4611711e-5,	2.1125553e-6).
coef2_disp(pk1,	1.3066001e-4,	-9.5958149e-7,	1.5899017e-7).
coef2_disp(pk2,	2.9001564e-4,	-1.9602856e-5,	1.0967718e-6).
coef2_disp(pk3,	2.1255152e-4,	-8.9190321e-6,	5.7854334e-7).
coef2_disp(pk50,	1.5661848e-4,	-4.3866422e-6,	3.1438773e-7).
coef2_disp(pk51,	1.4455285e-4,	-5.1413770e-6,	3.5994942e-7).
coef2_disp(psk2,	1.7407923e-4,	-3.2622544e-7,	1.9916158e-7).
coef2_disp(psk3,	1.1916606e-4,	6.4250627e-6,	-1.7478706e-7).
coef2_disp(psk50,	1.7638709e-4,	-3.4507987e-6,	2.9704663e-7).
coef2_disp(psk52,	3.8218429e-4,	-2.8742342e-5,	1.6748094e-6).
coef2_disp(psk53,	2.7924791e-4,	-1.2779218e-5,	9.4769182e-7).
coef2_disp(sf156,	2.3376489e-3,	-1.0682651e-4,	2.1446850e-5).
coef2_disp(sf16,	2.7505904e-3,	-1.5173832e-4,	2.5644487e-5).
coef2_disp(sfn64,	1.4083879e-3,	-4.4978076e-5,	1.1419083e-5).
coef2_disp(sf1,	1.7491597e-3,	-7.8964252e-5,	1.1858651e-5).
coef2_disp(sf10,	2.0551378e-3,	-1.1396226e-4,	1.6340021e-5).
coef2_disp(sf11,	2.9462812e-3,	-2.0371019e-4,	2.7633569e-5).
coef2_disp(sf12,	1.1970650e-3,	-5.5728265e-5,	7.8814553e-6).
coef2_disp(sf13,	2.5277439e-3,	-1.7332899e-4,	2.1465274e-5).
coef2_disp(sf14,	2.9250972e-3,	-2.1913665e-4,	2.6700784e-5).
coef2_disp(sf15,	1.9137570e-3,	-1.2468626e-4,	1.5187223e-5).
coef2_disp(sf16,	1.0126871e-3,	-2.7861159e-5,	5.8927698e-6).
coef2_disp(sf17,	1.0760709e-3,	-3.3523570e-5,	6.4052431e-6).
coef2_disp(sf18,	1.7143328e-3,	-7.4064352e-5,	1.2078466e-5).
coef2_disp(sf19,	1.2120420e-3,	-4.5516530e-5,	7.5592211e-6).
coef2_disp(sf2,	1.1171914e-3,	-4.0112089e-5,	6.6254840e-6).
coef2_disp(sf3,	2.0259158e-3,	-1.0348282e-4,	1.4959114e-5).
coef2_disp(sf4,	1.8634691e-3,	-6.5403181e-5,	1.3765657e-5).
coef2_disp(sf5,	1.2739656e-3,	-4.7889342e-5,	8.0562028e-6).
coef2_disp(sf50,	1.5350870e-3,	-9.1737353e-5,	9.9741416e-6).
coef2_disp(sf51,	2.2651418e-3,	-2.0072278e-4,	1.6380215e-5).
coef2_disp(sf52,	1.6834427e-3,	-8.6832245e-5,	1.1266792e-5).

coef2_disp(sf53, 2.0364401e-3, -1.1763293e-4, 1.5672506e-5).
coef2_disp(sf54, 1.6830114e-3, -4.6451087e-5, 1.2104532e-5).
coef2_disp(sf55, 2.1692831e-3, -9.4743970e-5, 1.6404455e-5).
coef2_disp(sf56, 1.9779903e-3, -4.8421498e-5, 1.5734251e-5).
coef2_disp(sf57, 5.2295853e-3, -4.6931979e-4, 4.4359036e-5).
coef2_disp(sf58, 4.2207315e-3, -2.2895268e-4, 4.0847905e-5).
coef2_disp(sf59, 8.0837909e-3, -7.6975589e-4, 7.9262505e-5).
coef2_disp(sf6, 3.1800214e-3, -2.1953184e-4, 2.6671014e-5).
coef2_disp(sf61, 1.6807292e-3, -3.3978727e-5, 1.2154392e-5).
coef2_disp(sf62, 1.2307943e-3, -3.7780898e-5, 7.8617786e-6).
coef2_disp(sf63, 1.7881803e-3, -5.2282393e-5, 1.2867869e-5).
coef2_disp(sf7, 9.7700411e-4, -2.6551022e-5, 5.4909672e-6).
coef2_disp(sf8, 1.4385871e-3, -5.8543435e-5, 9.3241989e-6).
coef2_disp(sf9, 1.0141982e-3, -2.3235110e-5, 5.8174500e-6).
coef2_disp(skn18, 3.4939073e-4, -1.0028109e-5, 1.1802616e-6).
coef2_disp(sk1, 2.7409985e-4, -1.7393911e-6, 4.0037912e-7).
coef2_disp(sk10, 2.2083748e-4, 3.5467529e-6, 2.6143582e-7).
coef2_disp(sk11, 2.4249380e-4, -7.5561129e-6, 5.9784011e-7).
coef2_disp(sk12, 2.0838047e-4, 1.5107581e-7, 2.4301769e-7).
coef2_disp(sk13, 2.9676678e-4, -9.3746109e-6, 7.4341245e-7).
coef2_disp(sk14, 1.5266041e-4, 6.8001252e-6, -6.7666095e-8).
coef2_disp(sk15, 1.9944840e-4, 6.3298328e-6, 1.9631497e-9).
coef2_disp(sk16, 2.2377211e-4, -4.9029910e-7, 2.8445925e-7).
coef2_disp(sk19, 2.3655893e-4, 1.4330022e-6, 2.9327826e-7).
coef2_disp(sk2, 1.8518895e-4, 8.4796564e-6, -7.0710768e-8).
coef2_disp(sk20, 2.5565265e-4, -9.6947284e-6, 6.8512885e-7).
coef2_disp(sk3, 2.9976361e-4, -9.8865231e-6, 8.2251299e-7).
coef2_disp(sk4, 2.1134478e-4, 3.4130130e-6, 1.2673355e-7).
coef2_disp(sk5, 1.6392687e-4, 4.7463374e-6, -4.9303610e-8).
coef2_disp(sk51, 2.8628731e-4, -8.4638718e-6, 7.9967143e-7).
coef2_disp(sk52, 3.0997141e-4, -3.5480823e-6, 5.7899721e-7).
coef2_disp(sk55, 2.4958755e-4, -3.2759191e-6, 3.6957062e-7).
coef2_disp(sk6, 3.0347388e-4, -6.5261354e-6, 6.9039819e-7).
coef2_disp(sk7, 1.7370120e-4, 6.3461730e-6, -5.3316579e-8).
coef2_disp(sk8, 3.5159357e-4, -1.1852989e-5, 9.7956516e-7).
coef2_disp(sk9, 1.8861769e-4, 1.3679471e-5, -3.0071065e-7).
coef2_disp(sskn5, 6.4060677e-4, -3.9493178e-5, 3.2567627e-6).
coef2_disp(sskn8, 5.0554082e-4, -2.1403288e-5, 2.4582991e-6).
coef2_disp(ssk1, 2.5292407e-4, 5.2540756e-6, 3.5912155e-7).
coef2_disp(ssk2, 2.5881635e-4, 6.7648572e-6, 3.4731723e-7).
coef2_disp(ssk3, 4.3401638e-4, -1.2949572e-5, 1.5042044e-6).
coef2_disp(ssk4, 2.7742702e-4, 1.2686469e-7, 4.5044790e-7).
coef2_disp(ssk50, 3.7688285e-4, -9.5448507e-6, 1.2959815e-6).
coef2_disp(ssk51, 2.8025567e-4, 1.1591526e-7, 7.3266529e-7).
coef2_disp(ssk52, 4.6286000e-4, -1.3476750e-5, 1.6107565e-6).
coef2_disp(tifn5, 1.2335546e-3, -8.5902987e-5, 1.0257463e-5).
coef2_disp(tif1, 4.2650638e-4, -2.1564809e-5, 2.6364065e-6).
coef2_disp(tif2, 6.1537921e-4, -3.9493915e-5, 4.7363335e-6).
coef2_disp(tif3, 1.1021697e-3, -9.3622430e-5, 8.2094068e-6).
coef2_disp(tif4, 1.1999618e-3, -8.8610291e-5, 9.8139193e-6).
coef2_disp(tif6, 2.0230960e-3, -1.8345806e-4, 2.3097036e-5).
coef2_disp(tik1, 2.1822593e-4, -5.5063640e-6, 5.4469060e-7).
coef2_disp(tisf1, 2.4436836e-3, -2.1244237e-4, 2.3518712e-5).
coef2_disp(ubk7, 1.4068151e-4, 1.3041796e-6, 4.8615933e-8).
coef2_disp(uk50, 1.4952379e-4, 3.4334254e-6, 5.9393094e-8).
coef2_disp(zkn7, 2.7117600e-5, 1.7553283e-5, -7.0564114e-7).
coef2_disp(zk1, 2.6311112e-4, -8.2854201e-6, 7.3735801e-7).
coef2_disp(zk5, 2.7439704e-4, -5.9213789e-6, 6.3841709e-7).

Appendix 22: Facts Base of the Refractive Index of the Optical Glass

/*

This base of facts contains the refractive index of the optical glasses, at some wavelengths usually used.

The parameters are:

Glass_name1:	Name of the optical glass
Nl:	Refractive index of the optical glass at the line l
Nc':	Refractive index of the optical glass at the line c'
N632:	Refractive index of the optical glass at 632 nano-meter
N589:	Refractive index of the optical glass at 589 nano-meter.
N587	Refractive index of the optical glass at 587 nano-meter.
N546	Refractive index of the optical glass at 546 nano-meter.

*/

%indice2_refract(Glass_name1, Nl, Nc', N632, N589, N587, N546).

indice2_refract(bafn10,1.66579,1.66646,1.66708,1.66991,1.67003,1.67341).
indice2_refract(bafn11,1.66260,1.66325,1.66386,1.66660,1.66672,1.67000).
indice2_refract(bafn6,1.58536,1.58594,1.58647,1.58889,1.58900,1.59189).
indice2_refract(baf12,1.63509,1.63575,1.63638,1.63918,1.63930,1.64266).
indice2_refract(baf13,1.66450,1.66519,1.66585,1.66879,1.66892,1.67245).
indice2_refract(baf3, 1.57893,1.57952,1.58008,1.58256,1.58267,1.58565).
indice2_refract(baf4, 1.60153,1.60217,1.60278,1.60550,1.60562,1.60889).
indice2_refract(baf5, 1.60361,1.60419,1.60474,1.60718,1.60729,1.61021).
indice2_refract(baf50,1.67816,1.67888,1.67955,1.68259,1.68273,1.68637).
indice2_refract(baf51,1.64792,1.64860,1.64924,1.65211,1.65224,1.65569).
indice2_refract(baf52,1.60469,1.60530,1.60588,1.60847,1.60859,1.61170).
indice2_refract(baf53,1.66578,1.66645,1.66708,1.66990,1.67003,1.67341).
indice2_refract(baf54,1.66258,1.66323,1.66385,1.66660,1.66672,1.67001).
indice2_refract(baf8, 1.61978,1.62041,1.62099,1.62362,1.62374,1.62690).
indice2_refract(baf9, 1.63927,1.63991,1.64050,1.64316,1.64328,1.64647).
indice2_refract(bak1, 1.56949,1.56997,1.57041,1.57241,1.57250,1.57487).
indice2_refract(bak2,1.53721,1.53765,1.53806,1.53988, 1.53996,1.54212).
indice2_refract(bak4,1.56576,1.56625,1.56670,1.56874, 1.56883,1.57125).
indice2_refract(bak5,1.55383,1.55429,1.55472,1.55663, 1.55671,1.55897).
indice2_refract(bak50,1.56476,1.56523,1.56568,1.56765,1.56774,1.57007).
indice2_refract(bak6,1.57136,1.57185,1.57230,1.57435, 1.57444,1.57687).
indice2_refract(balf3,1.56810,1.56862,1.56910,1.57126, 1.57135,1.57392).
indice2_refract(balf4,1.57632,1.57683,1.57731,1.57947, 1.57957,1.58214).
indice2_refract(balf5,1.54432,1.54480,1.54526,1.54730, 1.54739,1.54982).
indice2_refract(balf50,1.58549,1.58603,1.58654,1.58883,1.58893,1.59166).
indice2_refract(balf51,1.57062,1.57114,1.57163,1.57384,1.57393,1.57656).
indice2_refract(balf6,1.58569,1.58623,1.58672,1.58894, 1.58904,1.59169).
indice2_refract(balf8,1.55036,1.55088,1.55136,1.55351, 1.55361,1.55618).
indice2_refract(balkn3,1.51586,1.51628,1.51667,1.51841,1.51849,1.52054).
indice2_refract(balk1,1.52375,1.52418,1.52458,1.52634,1.52642,1.52851).
indice2_refract(basf1,1.62133,1.62207,1.62277,1.62592,1.62606,1.62987).

indice2_refract(basf10,1.64527,1.64604,1.64676,1.65002,1.65016,1.65410).
indice2_refract(basf12,1.66495,1.66574,1.66648,1.66983,1.66998,1.67403).
indice2_refract(basf13,1.69229,1.69312,1.69391,1.69745,1.69761,1.70190).
indice2_refract(basf14,1.69383,1.69475,1.69561,1.69951,1.69968,1.70442).
indice2_refract(basf2,1.65903,1.65988,1.66068,1.66430,1.66446,1.66885).
indice2_refract(basf5,1.59902,1.59969,1.60031,1.60311,1.60323,1.60660).
indice2_refract(basf50,1.70449,1.70539,1.70623,1.71003,1.71020,1.71480).
indice2_refract(basf51,1.71813,1.71901,1.71984,1.72356,1.72373,1.72823).
indice2_refract(basf52,1.69673,1.69753,1.69828,1.70166,1.70181,1.70587).
indice2_refract(basf54,1.72961,1.73065,1.73163,1.73607,1.73627,1.74169).
indice2_refract(basf55,1.69391,1.69483,1.69570,1.69963,1.69981,1.70459).
indice2_refract(basf56,1.65190,1.65272,1.65350,1.65699,1.65715,1.66139).
indice2_refract(basf57,1.64687,1.64759,1.64828,1.65134,1.65147,1.65516).
indice2_refract(basf6, 1.66284,1.66358,1.66428,1.66741,1.66755,1.67133).
indice2_refract(basf64,1.69576,1.69659,1.69736,1.70084,1.70100,1.70520).
indice2_refract(bk1, 1.50763,1.50802,1.50839,1.51002,1.51009,1.51201).
indice2_refract(bk10, 1.49552,1.49589,1.49624,1.49776,1.49782,1.49960).
indice2_refract(bk3, 1.49594,1.49632,1.49668,1.49824,1.49831,1.50014).
indice2_refract(bk6, 1.52851,1.52893,1.52932,1.53105,1.53113,1.53317).
indice2_refract(bk7, 1.51432,1.51472,1.51509,1.51673,1.51680,1.51872).
indice2_refract(bk8, 1.51764,1.51804,1.51842,1.52008,1.52015,1.52210).
indice2_refract(fk1, 1.46853,1.46888,1.46920,1.47063,1.47069,1.47236).
indice2_refract(fk3, 1.46232,1.46267,1.46300,1.46444,1.46450,1.46619).
indice2_refract(fk5, 1.48535,1.48569,1.48601,1.48743,1.48749,1.48914).
indice2_refract(fk51, 1.48480,1.48508,1.48534,1.48651,1.48656,1.48794).
indice2_refract(fk52, 1.48424,1.48453,1.48480,1.48600,1.48605,1.48747).
indice2_refract(fk54, 1.43552,1.43576,1.43598,1.43696,1.43700,1.43815).
indice2_refract(fn11, 1.61593,1.61672,1.61746,1.62081,1.62096,1.62502).
indice2_refract(f1, 1.62074,1.62154,1.62230,1.62573,1.62588,1.63004).
indice2_refract(f13, 1.61730,1.61810,1.61884,1.62222,1.62237,1.62646).
indice2_refract(f14, 1.59676,1.59749,1.59818,1.60126,1.60140,1.60513).
indice2_refract(f15, 1.60094,1.60168,1.60237,1.60551,1.60565,1.60945).
indice2_refract(f2, 1.61503,1.61582,1.61656,1.61989,1.62004,1.62408).
indice2_refract(f3, 1.60806,1.60883,1.60955,1.61279,1.61293,1.61685).
indice2_refract(f4, 1.61164,1.61242,1.61315,1.61644,1.61659,1.62058).
indice2_refract(f5, 1.59874,1.59948,1.60017,1.60328,1.60342,1.60718).
indice2_refract(f6, 1.63108,1.63191,1.63269,1.63620,1.63636,1.64062).
indice2_refract(f7, 1.62021,1.62102,1.62177,1.62521,1.62536,1.62953).
indice2_refract(f8, 1.59102,1.59173,1.59239,1.59538,1.59551,1.59912).
indice2_refract(f9, 1.61565,1.61641,1.61711,1.62031,1.62045,1.62431).
indice2_refract(kf1, 1.53723,1.53774,1.53821,1.54032,1.54041,1.54293).
indice2_refract(kf3, 1.51169,1.51215,1.51257,1.51446,1.51454,1.51678).
indice2_refract(kf50, 1.52776,1.52825,1.52871,1.53079,1.53088,1.53335).
indice2_refract(kf6, 1.51443,1.51491,1.51535,1.51733,1.51742,1.51978).
indice2_refract(kf9, 1.52035,1.52084,1.52129,1.52332,1.52341,1.52583).
indice2_refract(kzfn1, 1.54781,1.54834,1.54883,1.55105,1.55115,1.55379).
indice2_refract(kzfn2, 1.52633,1.52683,1.52729,1.52935,1.52944,1.53188).
indice2_refract(kzfsn2,1.55521,1.55571,1.55618,1.55827,1.55836,1.56082).
indice2_refract(kzfsn4,1.60924,1.60990,1.61052,1.61328,1.61340,1.61669).
indice2_refract(kzfsn5,1.64920,1.64998,1.65071,1.65397,1.65412,1.65804).
indice2_refract(kzfsn7,1.67523,1.67608,1.67688,1.68048,1.68064,1.68498).
indice2_refract(kzfsn9,1.59471,1.59532,1.59589,1.59844,1.59856,1.60159).
indice2_refract(kzfs1, 1.60894,1.60960,1.61021,1.61298,1.61310,1.61639).
indice2_refract(kzfs6, 1.58827,1.58886,1.58941,1.59186,1.59197,1.59487).
indice2_refract(kzfs8, 1.71434,1.71530,1.71621,1.72029,1.72047,1.72540).
indice2_refract(kzf6, 1.52370,1.52420,1.52466,1.52673,1.52682,1.52927).
indice2_refract(k10, 1.49867,1.49910,1.49950,1.50129,1.50137,1.50349).
indice2_refract(k11, 1.49764,1.49804,1.49841,1.50006,1.50013,1.50207).

indice2_refract(k3, 1.51556,1.51598,1.51638,1.51815,1.51823,1.52032).
indice2_refract(k4, 1.51620,1.51664,1.51705,1.51887,1.51895,1.52111).
indice2_refract(k5, 1.51982,1.52024,1.52064,1.52241,1.52249,1.52458).
indice2_refract(k50, 1.51992,1.52035,1.52074,1.52249,1.52257,1.52464).
indice2_refract(k51, 1.50258,1.50300,1.50338,1.50510,1.50518,1.50720).
indice2_refract(k7, 1.50854,1.50895,1.50934,1.51105,1.51112,1.51314).
indice2_refract(lafn10, 1.77909,1.77994,1.78073,1.78427,1.78443,1.78868).
indice2_refract(lafn21, 1.78332,1.78411,1.78485,1.78816,1.78831,1.79226).
indice2_refract(lafn23, 1.68485,1.68551,1.68612,1.68888,1.68900,1.69230).
indice2_refract(lafn24, 1.75242,1.75318,1.75388,1.75705,1.75719,1.76096).
indice2_refract(lafn28, 1.76843,1.76918,1.76988,1.77300,1.77314,1.77686).
indice2_refract(lafn7, 1.74319,1.74418,1.74511,1.74931,1.74950,1.75458).
indice2_refract(lafn8, 1.72995,1.73078,1.73155,1.73504,1.73520,1.73940).
indice2_refract(laf11a, 1.74996,1.75105,1.75207,1.75672,1.75693,1.76258).
indice2_refract(laf13, 1.76946,1.77041,1.77130,1.77533,1.77551,1.78037).
indice2_refract(laf2, 1.73905,1.73983,1.74056,1.74386,1.74400,1.7796).
indice2_refract(laf20, 1.67824,1.67891,1.67954,1.68235,1.68248,1.68585).
indice2_refract(laf22a, 1.77561,1.77658,1.77749,1.78161,1.78179,1.78677).
indice2_refract(laf25, 1.77863,1.77952,1.78035,1.78411,1.78427,1.78878).
indice2_refract(laf26, 1.74044,1.74131,1.74213,1.74581,1.74597,1.75040).
indice2_refract(laf3, 1.71252,1.71323,1.71389,1.71687,1.71700,1.72055).
indice2_refract(laf9, 1.78695,1.78821,1.78940,1.79480,1.79504,1.80166).
indice2_refract(lak112,1.67415,1.67475,1.67530,1.67779,1.67790,1.68084).
indice2_refract(lak121,1.63719,1.63772,1.63821,1.64038,1.64048,1.64304).
indice2_refract(lakn12,1.67419,1.67478,1.67533,1.67779,1.67790,1.68083).
indice2_refract(lakn13,1.68958,1.69020,1.69078,1.69339,1.69350,1.69660).
indice2_refract(lakn14,1.69297,1.69358,1.69415,1.69669,1.69680,1.69980).
indice2_refract(lakn22,1.64760,1.64816,1.64869,1.65103,1.65113,1.65391).
indice2_refract(lakn6, 1.63913,1.63967,1.64017,1.64240,1.64250,1.64514).
indice2_refract(lakn7, 1.64821,1.64875,1.64926,1.65150,1.65160,1.65426).
indice2_refract(lak10, 1.71568,1.71637,1.71701,1.71987,1.72000,1.72340).
indice2_refract(lak11, 1.65480,1.65536,1.65588,1.65820,1.65830,1.66104).
indice2_refract(lak16a,1.72921,1.72989,1.73053,1.73338,1.73350,1.73688).
indice2_refract(lak20, 1.68944,1.69008,1.69068,1.69337,1.69349,1.69669).
indice2_refract(lak21, 1.63724,1.63776,1.63825,1.64040,1.64050,1.64304).
indice2_refract(lak23, 1.66527,1.66584,1.66636,1.66871,1.66882,1.67159).
indice2_refract(lak28, 1.73985,1.74056,1.74122,1.74416,1.74429,1.74778).
indice2_refract(lak31, 1.69296,1.69356,1.69413,1.69662,1.69673,1.69968).
indice2_refract(lak33, 1.74962,1.75031,1.75096,1.75385,1.75398,1.75740).
indice2_refract(lak8, 1.70898,1.70962,1.71022,1.71289,1.71300,1.71616).
indice2_refract(lak9, 1.68716,1.68777,1.68835,1.69089,1.69100,1.69401).
indice2_refract(lasfn15,1.87118,1.87225,1.87326, 1.87780,1.87800,1.88347).
indice2_refract(lasfn18,1.90522,1.90652,1.90773,1.91324,1.91348,1.92016).
indice2_refract(lasfn30,1.79799,1.79881,1.79958, 1.80303,1.80318,1.80730).
indice2_refract(lasfn31,1.87429,1.87529,1.87624, 1.88048,1.88067,1.88577).
indice2_refract(lasfn9, 1.84256,1.84376,1.84489,1.85003,1.85026,1.85651).
indice2_refract(lasf11, 1.79624,1.79710,1.79790,1.80150,1.80166,1.80597).
indice2_refract(lasf13, 1.84856,1.84964,1.85065,1.85524,1.85544,1.86099).
indice2_refract(lasf3, 1.80210,1.80303,1.80390,1.80783,1.80801,1.81273).
indice2_refract(lasf32, 1.79581,1.79701,1.79814,1.80326,1.80349,1.80974).
indice2_refract(lasf33, 1.79908,1.80015,1.80117,1.80575,1.80596,1.81153).
indice2_refract(lasf8, 1.79996,1.80113,1.80222,1.80719,1.80741,1.81345).
indice2_refract(lf1, 1.56910,1.56973,1.57032,1.57297,1.57309,1.57629).
indice2_refract(lf2, 1.58495,1.58562,1.58625,1.58909,1.58921,1.59263).
indice2_refract(lf3, 1.57805,1.57869,1.57930,1.58203,1.58215,1.58544).
indice2_refract(lf4, 1.57433,1.57498,1.57559,1.57833,1.57845,1.58175).
indice2_refract(lf5, 1.57723,1.57789,1.57851,1.58132,1.58144,1.58482).
indice2_refract(lf6, 1.56339,1.56401,1.56459,1.56720,1.56732,1.57047).

indice2_refract(lf7, 1.57090,1.57155,1.57216,1.57489,1.57501,1.57830).
indice2_refract(lf8, 1.56060,1.56120,1.56177,1.56432,1.56444,1.56750).
indice2_refract(lgsk2, 1.58311,1.58356,1.58399,1.58591,1.58599,1.58828).
indice2_refract(llf1, 1.54457,1.54513,1.54566,1.54804,1.54814,1.55099).
indice2_refract(llf2, 1.53729,1.53783,1.53834,1.54062,1.54072,1.54344).
indice2_refract(llf3, 1.55658,1.55714,1.55767,1.56003,1.56013,1.56295).
indice2_refract(llf4, 1.55768,1.55827,1.55881,1.56127,1.56138,1.56433).
indice2_refract(llf6, 1.52845,1.52897,1.52945,1.53162,1.53172,1.53431).
indice2_refract(llf7, 1.54523,1.54579,1.54633,1.54873,1.54883,1.55170).
indice2_refract(pk1, 1.50146,1.50183,1.50218,1.50371,1.50378,1.50558).
indice2_refract(pk2, 1.51576,1.51615,1.51652,1.51814,1.51821,1.52011).
indice2_refract(pk3, 1.52292,1.52332,1.52369,1.52535,1.52542,1.52736).
indice2_refract(pk50, 1.51824,1.51861,1.51895,1.52047,1.52054,1.52232).
indice2_refract(pk51, 1.52646,1.52680,1.52711,1.52849,1.52855,1.53019).
indice2_refract(psk2, 1.56597,1.56641,1.56682,1.56865,1.56873,1.57088).
indice2_refract(psk3, 1.54965,1.55008,1.55048,1.55224,1.55232,1.55440).
indice2_refract(psk50, 1.55499,1.55539,1.55577,1.55746,1.55753,1.55951).
indice2_refract(psk52, 1.60028,1.60073,1.60115,1.60302,1.60310,1.60530).
indice2_refract(psk53, 1.61717,1.61764,1.61808,1.62005,1.62014,1.62247).
indice2_refract(sfl56, 1.77606,1.77740,1.77867,1.78444,1.78470,1.79179).
indice2_refract(sfl6, 1.79609,1.79750,1.79884,1.80491,1.80518,1.81265).
indice2_refract(sfn64, 1.69909,1.70014,1.70114,1.70565,1.70585,1.71135).
indice2_refract(sf1, 1.71032,1.71141,1.71245,1.71715,1.71736,1.72311).
indice2_refract(sf10, 1.72085,1.72200,1.72309,1.72803,1.72825,1.73430).
indice2_refract(sf11, 1.77599,1.77734,1.77862,1.78446,1.78472,1.79190).
indice2_refract(sf12, 1.64271,1.64359,1.64441,1.64814,1.64831,1.65285).
indice2_refract(sf13, 1.73304,1.73424,1.73538,1.74054,1.74077,1.74710).
indice2_refract(sf14, 1.75358,1.75486,1.75606,1.76157,1.76182,1.76859).
indice2_refract(sf15, 1.69221,1.69326,1.69425,1.69875,1.69895,1.70445).
indice2_refract(sf16, 1.64056,1.64143,1.64225,1.64595,1.64611,1.65061).
indice2_refract(sf17, 1.64453,1.64541,1.64624,1.65000,1.65017,1.65474).
indice2_refract(sf18, 1.71436,1.71548,1.71653,1.72130,1.72151,1.72734).
indice2_refract(sf19, 1.66090,1.66182,1.66269,1.66662,1.66680,1.67158).
indice2_refract(sf2, 1.64210,1.64297,1.64379,1.64752,1.64769,1.65222).
indice2_refract(sf3, 1.73242,1.73360,1.73471,1.73977,1.74000,1.74620).
indice2_refract(sf4, 1.74730,1.74853,1.74969,1.75496,1.75520,1.76167).
indice2_refract(sf5, 1.66661,1.66756,1.66846,1.67252,1.67270,1.67764).
indice2_refract(sf50, 1.64892,1.64983,1.65068,1.65455,1.65473,1.65944).
indice2_refract(sf51, 1.65441,1.65532,1.65618,1.66007,1.66025,1.66499).
indice2_refract(sf52, 1.68200,1.68301,1.68397,1.68833,1.68852,1.69384).
indice2_refract(sf53, 1.72096,1.72210,1.72318,1.72808,1.72830,1.73430).
indice2_refract(sf54, 1.73318,1.73437,1.73549,1.74057,1.74080,1.74703).
indice2_refract(sf55, 1.75366,1.75493,1.75612,1.76156,1.76180,1.76847).
indice2_refract(sf56, 1.77605,1.77739,1.77866,1.78444,1.78470,1.79180).
indice2_refract(sf57, 1.83651,1.83808,1.83957,1.84636,1.84666,1.85504).
indice2_refract(sf58, 1.90550,1.90738,1.90914,1.91725,1.91761,1.92765).
indice2_refract(sf59, 1.93928,1.94132,1.94325,1.95210,1.95250,1.96349).
indice2_refract(sf6, 1.79609,1.79750,1.79883,1.80491,1.80518,1.81265).
indice2_refract(sf61, 1.74297,1.74420,1.74535,1.75060,1.75084,1.75728).
indice2_refract(sf62, 1.67512,1.67609,1.67701,1.68115,1.68134,1.68639).
indice2_refract(sf63, 1.74061,1.74182,1.74296,1.74817,1.74840,1.75477).
indice2_refract(sf7, 1.63439,1.63524,1.63603,1.63964,1.63980,1.64418).
indice2_refract(sf8, 1.68250,1.68351,1.68445,1.68874,1.68893,1.69416).
indice2_refract(sf9, 1.64878,1.64967,1.65050,1.65429,1.65446,1.65907).
indice2_refract(skn18, 1.63505,1.63561,1.63613,1.63844,1.63854,1.64129).
indice2_refract(sk1, 1.60699,1.60751,1.60799,1.61016,1.61025,1.61282).
indice2_refract(sk10, 1.61949,1.62001,1.62050,1.62270,1.62280,1.62541).
indice2_refract(sk11, 1.56101,1.56146,1.56188,1.56376,1.56384,1.56605).

indice2_refract(sk12, 1.58015,1.58062,1.58107,1.58304,1.58313,1.58547).
indice2_refract(sk13, 1.58873,1.58922,1.58968,1.59172,1.59181,1.59423).
indice2_refract(sk14, 1.60007,1.60056,1.60101,1.60302,1.60311,1.60548).
indice2_refract(sk15, 1.61973,1.62025,1.62073,1.62289,1.62299,1.62555).
indice2_refract(sk16, 1.61727,1.61777,1.61824,1.62032,1.62041,1.62286).
indice2_refract(sk19, 1.61018,1.61070,1.61118,1.61333,1.61342,1.61597).
indice2_refract(sk2, 1.60414,1.60465,1.60513,1.60729,1.60738,1.60994).
indice2_refract(sk20, 1.55684,1.55729,1.55770,1.55955,1.55963,1.56181).
indice2_refract(sk3, 1.60567,1.60617,1.60664,1.60872,1.60881,1.61127).
indice2_refract(sk4, 1.60954,1.61005,1.61052,1.61263,1.61272,1.61521).
indice2_refract(sk5, 1.58619,1.58666,1.58710,1.58904,1.58913,1.59142).
indice2_refract(sk51, 1.61778,1.61827,1.61874,1.62081,1.62090,1.62336).
indice2_refract(sk52, 1.63505,1.63561,1.63613,1.63844,1.63854,1.64128).
indice2_refract(sk55, 1.61725,1.61776,1.61823,1.62032,1.62041,1.62287).
indice2_refract(sk6, 1.61046,1.61098,1.61147,1.61365,1.61375,1.61634).
indice2_refract(sk7, 1.60418,1.60468,1.60514,1.60720,1.60729,1.60973).
indice2_refract(sk8, 1.60787,1.60839,1.60888,1.61107,1.61117,1.61378).
indice2_refract(sk9, 1.61069,1.61122,1.61172,1.61395,1.61405,1.61670).
indice2_refract(sskn5, 1.65456,1.65517,1.65575,1.65833,1.65844,1.66152).
indice2_refract(sskn8, 1.61400,1.61459,1.61514,1.61761,1.61772,1.62067).
indice2_refract(ssk1, 1.61375,1.61430,1.61481,1.61710,1.61720,1.61993).
indice2_refract(ssk2, 1.61878,1.61933,1.61986,1.62220,1.62230,1.62509).
indice2_refract(ssk3, 1.61123,1.61180,1.61234,1.61473,1.61484,1.61770).
indice2_refract(ssk4, 1.61427,1.61480,1.61531,1.61755,1.61765;1.62032).
indice2_refract(ssk50, 1.61442,1.61498,1.61550,1.61785,1.61795,1.62075).
indice2_refract(ssk51, 1.60022,1.60075,1.60126,1.60351,1.60361,1.60629).
indice2_refract(ssk52, 1.65455,1.65517,1.65574,1.65832,1.65844,1.66152).
indice2_refract(tifn5, 1.58867,1.58944,1.59016,1.59341,1.59356,1.59751).
indice2_refract(tif1, 1.50818,1.50865,1.50910,1.51109,1.51118,1.51356).
indice2_refract(tif2, 1.52911,1.52966,1.53017,1.53246,1.53256,1.53531).
indice2_refract(tif3, 1.54382,1.54442,1.54499,1.54754,1.54765,1.55072).
indice2_refract(tif4, 1.57945,1.58017,1.58085,1.58393,1.58406,1.58779).
indice2_refract(tif6, 1.61080,1.61168,1.61252,1.61633,1.61650,1.62118).
indice2_refract(tik1, 1.47621,1.47660,1.47697,1.47862,1.47869,1.48063).
indice2_refract(tisf1, 1.66667,1.66772,1.66870,1.67319,1.67339,1.67889).
indice2_refract(ubk7, 1.51433,1.51472,1.51509,1.51673,1.51680,1.51872).
indice2_refract(uk50, 1.51993,1.52035,1.52074,1.52249,1.52257,1.52464).
indice2_refract(zkn7, 1.50592,1.50633,1.50671,1.50840,1.50847,1.51045).
indice2_refract(zk1, 1.53036,1.53080,1.53122,1.53307,1.53315,1.53534).
indice2_refract(zk5, 1.53084,1.53130,1.53173,1.53366,1.53375,1.53605).

/*
This base of facts contains the refractive index of the optical glasses, at the following lines f, f', g, h and i.

The parameters are:

Glass_name1:	Name of the optical glass
Nf:	Refractive index of the optical glass at the line f
Nf':	Refractive index of the optical glass at the line f'
Ng:	Refractive index of the optical glass at the line g
Nh:	Refractive index of the optical glass at the line h.
Ni	Refractive index of the optical glass at the line i.

*/

```
%indice3_refract(Nam1,Nf,Nf', Ng, Nh,Ni).
indice3_refract(bafn10,1.68001,1.68084,1.68803,1.69486,1.70690).
indice3_refract(bafn11,1.67637,1.67717,1.68411,1.69066,1.70217).
indice3_refract(bafn6,1.59752,1.59823,1.60435,1.61016,1.62036).
indice3_refract(baf12,1.64924,1.65007,1.65728,1.66415,1.67636).
indice3_refract(baf13,1.67937,1.68025,1.68783,1.69507,1.70792).
indice3_refract(baf3, 1.59147,1.59221,1.59857,1.60460,1.61524).
indice3_refract(baf4, 1.61532,1.61613,1.62318,1.62990,1.64182).
indice3_refract(baf5, 1.61590,1.61662,1.62279,1.62862,1.63880).
indice3_refract(baf50,1.69350,1.69440,1.70219,1.70959,1.72267).
indice3_refract(baf51,1.66244,1.66329,1.67067,1.67769,1.69008).
indice3_refract(baf52,1.61779,1.61856,1.62521,1.63154,1.64272).
indice3_refract(baf53,1.68000,1.68083,1.68801,1.69482,1.70679).
indice3_refract(baf54,1.67641,1.67721,1.68416,1.69072,1.70221).
indice3_refract(baf8, 1.63305,1.63383,1.64054,1.64691,1.65810).
indice3_refract(baf9, 1.65269,1.65347,1.66023,1.66663,1.67785).
indice3_refract(bak1,1.57943,1.58000,1.58488,1.58940, 1.59715).
indice3_refract(bak2,1.54625,1.54677,1.55117,1.55525, 1.56222).
indice3_refract(bak4,1.57590,1.57648,1.58145,1.58609, 1.59407).
indice3_refract(bak5,1.56332,1.56386,1.56850,1.57280, 1.58015).
indice3_refract(bak50,1.57455,1.57510,1.57987,1.58429,1.59183).
indice3_refract(bak6,1.58154,1.58213,1.58713,1.59180, 1.59983).
indice3_refract(balf3,1.57890,1.57952,1.58488,1.58990, 1.59856).
indice3_refract(balf4,1.58711,1.58773,1.59307,1.59806, 1.60670).
indice3_refract(balf5,1.55452,1.55511,1.56018,1.56492, 1.57313).
indice3_refract(balf50,1.59695,1.59761,1.60333,1.60869,1.61803).
indice3_refract(balf51,1.58164,1.58227,1.58776,1.59290,1.60184).
indice3_refract(balf6,1.59681,1.59745,1.60296,1.60812, 1.61706).
indice3_refract(balf8,1.56118,1.56181,1.56721,1.57230, 1.58116).
indice3_refract(balkn3,1.52447,1.52496,1.52914,1.53301,1.53962).
indice3_refract(balk1,1.53252,1.53302,1.53729,1.54125,1.54800).
indice3_refract(basf1,1.63740,1.63836,1.64671,1.65476,1.66925).
indice3_refract(basf10,1.66188,1.66287,1.67150,1.67980,1.69473).
indice3_refract(basf12,1.68204,1.68306,1.69194,1.70052,1.71603).
indice3_refract(basf13,1.71038,1.71145,1.72090,1.73004,1.74669).
indice3_refract(basf14,1.71384,1.71504,1.72563,1.73596,1.75498).
indice3_refract(basf2,1.67757,1.67869,1.68844,1.69791,1.71514).
indice3_refract(basf5,1.61323,1.61406,1.62136,1.62833,1.64074).
indice3_refract(basf50,1.72388,1.72504,1.73515,1.74492,1.76259).
```

indice3_refract(basf51,1.73712,1.73825,1.74810,1.75758,1.77463).
indice3_refract(basf52,1.71384,1.71485,1.72362,1.73201,1.74698).
indice3_refract(basf54,1.75251,1.75390,1.76614,1.77817,1.80054).
indice3_refract(basf55,1.71409,1.71530,1.72597,1.73641,1.75572).
indice3_refract(basf56,1.66979,1.67086,1.68023,1.68931,1.70579).
indice3_refract(basf57,1.66242,1.66334,1.67134,1.67901,1.69273).
indice3_refract(basf6,1.67876,1.67970,1.68790,1.69577,1.70985).
indice3_refract(basf64,1.71347,1.71452,1.72367,1.73248,1.74836).
indice3_refract(bk1, 1.51566,1.51612,1.51998,1.52355,1.52962).
indice3_refract(bk10, 1.50296,1.50337,1.50690,1.51014,1.51562).
indice3_refract(bk3, 1.50360,1.50403,1.50767,1.51101,1.51666).
indice3_refract(bk6, 1.53706,1.53754,1.54166,1.54546,1.55193).
indice3_refract(bk7, 1.52238,1.52283,1.52669,1.53024,1.53626).
indice3_refract(bk8, 1.52581,1.52627,1.53019,1.53380,1.53994).
indice3_refract(fk1, 1.47552,1.47591,1.47924,1.48230,1.48746).
indice3_refract(fk3, 1.46939,1.46978,1.47315,1.47626,1.48150).
indice3_refract(fk5, 1.49227,1.49266,1.49593,1.49894,1.50401).
indice3_refract(fk51,1.49056,1.49088,1.49365,1.49619, 1.50048).
indice3_refract(fk52,1.49018,1.49051,1.49337,1.49601, 1.50045).
indice3_refract(fk54,1.44034,1.44061,1.44291,1.44501, 1.44855).
indice3_refract(fn11,1.63309,1.63412,1.64320,1.65211, 1.66870).
indice3_refract(f1, 1.63827,1.63932,1.64851,1.65741,1.67356).
indice3_refract(f13, 1.63457,1.63560,1.64465,1.65341,1.66929).
indice3_refract(f14, 1.61249,1.61343,1.62160,1.62946,1.64361).
indice3_refract(f15, 1.61695,1.61790,1.62623,1.63426,1.64871).
indice3_refract(f2, 1.63208,1.63310,1.64202,1.65063,1.66621).
indice3_refract(f3, 1.62461,1.62560,1.63423,1.64256,1.65760).
indice3_refract(f4, 1.62848,1.62948,1.63828,1.64678,1.66215).
indice3_refract(f5, 1.61461,1.61556,1.62380,1.63174,1.64604).
indice3_refract(f6, 1.64909,1.65017,1.65963,1.66879,1.68545).
indice3_refract(f7, 1.63779,1.63885,1.64809,1.65704,1.67332).
indice3_refract(f8, 1.60622,1.60713,1.61500,1.62257,1.63615).
indice3_refract(f9, 1.63194,1.63292,1.64140,1.64959,1.66440).
indice3_refract(kf1, 1.54781,1.54842,1.55371,1.55869,1.56739).
indice3_refract(kf3, 1.52110,1.52164,1.52627,1.53060,1.53807).
indice3_refract(kf50, 1.53814,1.53874,1.54391,1.54877,1.55722).
indice3_refract(kf6, 1.52434,1.52492,1.52984,1.53446,1.54250).
indice3_refract(kf9, 1.53052,1.53110,1.53616,1.54093,1.54922).
indice3_refract(kzfn1,1.55891,1.55955,1.56509,1.57031,1.57941).
indice3_refract(kzfn2,1.53659,1.53718,1.54224,1.54699,1.55521).
indice3_refract(kzfsn2,1.56552,1.56610,1.57111,1.57578,1.58381).
indice3_refract(kzfsn4,1.62309,1.62390,1.63085,1.63745,1.64904).
indice3_refract(kzfsn5,1.66571,1.66668,1.67512,1.68319,1.69760).
indice3_refract(kzfsn7,1.69353,1.69462,1.70412,1.71330,1.72993).
indice3_refract(kzfsn9,1.60747,1.60820,1.61456,1.62055,1.63103).
indice3_refract(kzfs1, 1.62276,1.62356,1.63048,1.63702,1.64850).
indice3_refract(kzfs6, 1.60048,1.60118,1.60723,1.61292,1.62285).
indice3_refract(kzfs8, 1.73516,1.73640,1.74726,1.75777,1.77680).
indice3_refract(kzf6, 1.53400,1.53460,1.53969,1.54446,1.55273).
indice3_refract(k10, 1.50756,1.50807,1.51243,1.51649,1.52350).
indice3_refract(k11, 1.50578,1.50624,1.51019,1.51385,1.52011).
indice3_refract(k3, 1.52435,1.52485,1.52913,1.53311,1.53992).
indice3_refract(k4, 1.52524,1.52576,1.53017,1.53428,1.54132).
indice3_refract(k5, 1.52860,1.52910,1.53338,1.53735,1.54412).
indice3_refract(k50, 1.52861,1.52910,1.53331,1.53721,1.54385).
indice3_refract(k51, 1.51106,1.51154,1.51564,1.51945,1.52596).
indice3_refract(k7, 1.51700,1.51748,1.52159,1.52540,1.53189).
indice3_refract(lafn10,1.79694,1.79798,1.80699,1.81552,1.83054).

indice3_refract(lafn21,1.79992,1.80088,1.80915,1.81693,1.83043).
indice3_refract(lafn23,1.69871,1.69952,1.70646,1.71289,1.72434).
indice3_refract(lafn24,1.76826,1.76918,1.77706,1.78446,1.79732).
indice3_refract(lafn28,1.78403,1.78493,1.79264,1.79984,1.81224).
indice3_refract(lafn7, 1.76464,1.76592,1.77713,1.78798,1.80766).
indice3_refract(lafn8, 1.74763,1.74867,1.75772,1.76638,1.78179).
indice3_refract(laf11a,1.77383,1.77527,1.78793,1.80028,1.82295).
indice3_refract(laf13, 1.78996,1.79117,1.80179,1.81201,1.83036).
indice3_refract(laf2, 1.75568,1.75666,1.76510,1.77309,1.78717).
indice3_refract(laf20, 1.69240,1.69322,1.70033,1.70704,1.71877).
indice3_refract(laf22a,1.79663,1.79788,1.80886,1.81948,1.83879).
indice3_refract(laf25, 1.79762,1.79874,1.80846,1.81775,1.83433).
indice3_refract(laf26, 1.75909,1.76019,1.76977,1.77897,1.79543).
indice3_refract(laf3, 1.72747,1.72834,1.73585,1.74293,1.75529).
indice3_refract(laf9, 1.81495,1.81666,1.83180,1.84675,1.87469).
indice3_refract(lak12,1.68649,1.68720,1.69322,1.69882,1.70839).
indice3_refract(lak121,1.64791,1.64852,1.65367,1.65844,1.66654).
indice3_refract(lakn12,1.68647,1.68717,1.69320,1.69881,1.70840).
indice3_refract(lakn13,1.70258,1.70333,1.70975,1.71573,1.72600).
indice3_refract(lakn14,1.70554,1.70626,1.71237,1.71804,1.72773).
indice3_refract(lakn22,1.65925,1.65992,1.66563,1.67093,1.68000).
indice3_refract(lakn6,1.65022,1.65085,1.65625,1.66127,1.66982).
indice3_refract(lakn7,1.65935,1.65998,1.66540,1.67042,1.67898).
indice3_refract(lak10,1.72996,1.73079,1.73784,1.74444,1.75587).
indice3_refract(lak11,1.66630,1.66696,1.67256,1.67775,1.68662).
indice3_refract(lak16a,1.74338,1.74419,1.75114,1.75762,1.76874).
indice3_refract(lak20,1.70289,1.70367,1.71033,1.71657,1.72729).
indice3_refract(lak21,1.64790,1.64850,1.65366,1.65844,1.66657).
indice3_refract(lak23,1.67693,1.67759,1.68328,1.68855,1.69756).
indice3_refract(lak28,1.75451,1.75535,1.76257,1.76931,1.78090).
indice3_refract(lak31,1.70531,1.70601,1.71200,1.71755,1.72701).
indice3_refract(lak33,1.76400,1.76482,1.77187,1.77843,1.78966).
indice3_refract(lak8, 1.72222,1.72298,1.72944,1.73545,1.74574).
indice3_refract(lak9, 1.69979,1.70051,1.70667,1.71240,1.72219).
indice3_refract(lasfn15,1.89424,1.89560,1.90746, 1.91881,1.93900).
indice3_refract(lasfn18,1.93345,1.93515,1.95007, 1.96462,1.99133).
indice3_refract(lasfn30,1.81530,1.81631,1.82496, 1.83309,1.84719).
indice3_refract(lasfn31,1.89576,1.89702,1.90793, 1.91824,1.93627).
indice3_refract(lasfn9,1.86899,1.87059,1.88467,1.89844, _).
indice3_refract(lasf11,1.81435,1.81541,1.82452,1.83314,1.84821).
indice3_refract(lasf13,1.87194,1.87333,1.88547,1.89713,1.91806).
indice3_refract(lasf3,1.82199,1.82317,1.83335,1.84309, 1.86048).
indice3_refract(lasf32,1.82225,1.82385,1.83805,1.85206,1.87828).
indice3_refract(lasf33,1.82261,1.82403,1.83646,1.84856, _).
indice3_refract(lasf8,1.82550,1.82705,1.84062,1.85389, 1.87832).
indice3_refract(lf1, 1.58256,1.58335,1.59025,1.59684,1.60858).
indice3_refract(lf2, 1.59935,1.60020,1.60762,1.61473,1.62746).
indice3_refract(lf3, 1.59188,1.59270,1.59980,1.60660,1.61874).
indice3_refract(lf4, 1.58824,1.58906,1.59620,1.60305,1.61528).
indice3_refract(lf5, 1.59146,1.59230,1.59964,1.60667,1.61924).
indice3_refract(lf6, 1.57663,1.57741,1.58419,1.59067,1.60220).
indice3_refract(lf7, 1.58476,1.58558,1.59271,1.59953,1.61172).
indice3_refract(lf8, 1.57350,1.57426,1.58085,1.58713,1.59829).
indice3_refract(lgsk2,1.59271,1.59326,1.59801,1.60245,1.61005).
indice3_refract(llf1, 1.55655,1.55725,1.56332,1.56910,1.57931).
indice3_refract(llf2, 1.54876,1.54942,1.55521,1.56070,1.57036).
indice3_refract(llf3. 1.56845,1.56915,1.57515,1.58084,1.59089).
indice3_refract(llf4, 1.57009,1.57082,1.57713,1.58313,1.59375).

```
indice3_refract(llf6,  1.53935,1.53999,1.54546,1.55064,1.55972).
indice3_refract(llf7,  1.55731,1.55802,1.56415,1.56998,1.58027).
indice3_refract(pk1,  1.50898,1.50940,1.51298,1.51627,1.52182).
indice3_refract(pk2,  1.52372,1.52417,1.52798,1.53148,1.53742).
indice3_refract(pk3,  1.53104,1.53150,1.53538,1.53896,1.54503).
indice3_refract(pk50,1.52570,1.52612,1.52968,1.53294, 1.53846).
indice3_refract(pk51,1.53333,1.53372,1.53704,1.54010, 1.54528).
indice3_refract(psk2,1.57498,1.57549,1.57982,1.58382, 1.59061).
indice3_refract(psk3,1.55836,1.55885,1.56303,1.56689, 1.57342).
indice3_refract(psk50,1.56327,1.56374,1.56771,1.57138,1.57758).
indice3_refract(psk52,1.60950,1.61002,1.61447,1.61858,1.62555).
indice3_refract(psk53,1.62693,1.62749,1.63222,1.63660,1.64407).
indice3_refract(sfl56,1.80615,1.80801,1.82461,1.84128,  _).
indice3_refract(sfl6,  1.82780,1.82976,1.84731,1.86497,  _).
indice3_refract(sfn64,1.72238,1.72380,1.73637,1.74882,  _).
indice3_refract(sf1,  1.73462,1.73610,1.74916,1.76199,1.78577).
indice3_refract(sf10,  1.74648,1.74805,1.76197,1.77578,  _).
indice3_refract(sf11,  1.80645,1.80834,1.82518,1.84211,  _).
indice3_refract(sf12,  1.66187,1.66302,1.67315,1.68301,1.70110).
indice3_refract(sf13,  1.75988,1.76153,1.77621,1.79084,  _).
indice3_refract(sf14,  1.78229,1.78407,1.79988,1.81574,  _).
indice3_refract(sf15,  1.71546,1.71687,1.72939,1.74174,1.76486).
indice3_refract(sf16,  1.65954,1.66068,1.67069,1.68043,1.69818).
indice3_refract(sf17,  1.66384,1.66500,1.67521,1.68514,1.70329).
indice3_refract(sf18,  1.73903,1.74053,1.75380,1.76685,1.79108).
indice3_refract(sf19,  1.68110,1.68232,1.69302,1.70345,1.72255).
indice3_refract(sf2,  1.66123,1.66238,1.67249,1.68233,1.70029).
indice3_refract(sf3,  1.75866,1.76026,1.77444,1.78844,1.81456).
indice3_refract(sf4,  1.77468,1.77636,1.79121,1.80589,1.83333).
indice3_refract(sf5,  1.68750,1.68876,1.69985,1.71068,1.73055).
indice3_refract(sf50,  1.66884,1.67004,1.68061,1.69093,1.70989).
indice3_refract(sf51,  1.67445,1.67566,1.68630,1.69670,1.71597).
indice3_refract(sf52,  1.70448,1.70585,1.71789,1.72970,1.75155).
indice3_refract(sf53,  1.74635,1.74789,1.76161,1.77515,1.80047).
indice3_refract(sf54,  1.75955,1.76117,1.77546,1.78959,  _).
indice3_refract(sf55,  1.78193,1.78366,1.79908,1.81438,  _).
indice3_refract(sf56,  1.80614,1.80800,1.82448,1.84091,  _).
indice3_refract(sf57,  1.87205,1.87425,1.89391,1.91363,1.95148).
indice3_refract(sf58,  1.94816,1.95084,1.97486,1.99923,  _).
indice3_refract(sf59,  1.98605,1.98900,2.01559,2.04279,  _).
indice3_refract(sf6,  1.82775,1.82970,1.84705,1.86436,1.89716).
indice3_refract(sf61,  1.77026,1.77193,1.78677,1.80147,  _).
indice3_refract(sf62,  1.69646,1.69774,1.70909,1.72018,1.74056).
indice3_refract(sf63,  1.76761,1.76927,1.78393,1.79845,  _).
indice3_refract(sf7,  1.65288,1.65399,1.66372,1.67317,1.69036).
indice3_refract(sf8,  1.70460,1.70594,1.71772,1.72926,1.75049).
indice3_refract(sf9,  1.66822,1.66939,1.67966,1.68965,1.70788).
indice3_refract(skn18,1.64658,1.64724,1.65290,1.65819,1.66731).
indice3_refract(sk1,  1.61775,1.61837,1.62365,1.62856,1.63697).
indice3_refract(sk10,1.63043,1.63106,1.63642,1.64141, 1.64996).
indice3_refract(sk11,1.57028,1.57081,1.57530,1.57946, 1.58654).
indice3_refract(sk12,1.58996,1.59051,1.59529,1.59971. 1.60725).
indice3_refract(sk13,1.59888,1.59946,1.60442,1.60902, 1.61689).
indice3_refract(sk14,1.61003,1.61059,1.61541,1.61988, 1.62748).
indice3_refract(sk15,1.63046,1.63108,1.63631,1.64118, 1.64949).
indice3_refract(sk16,1.62756.1.62814,1.63312,1.63774, 1.64559).
indice3_refract(sk19,1.62087,1.62149,1.62672,1.63159, 1.63993).
indice3_refract(sk2,  1.61486,1.61547,1.62073,1.62562,1.63398).
```

```
indice3_refract(sk20,1.56598,1.56650,1.57093,1.57502, 1.58198).
indice3_refract(sk3,  1.61600,1.61659,1.62163,1.62630,1.63428).
indice3_refract(sk4,  1.62000,1.62059,1.62569,1.63042,1.63850).
indice3_refract(sk5,  1.59581,1.59635,1.60100,1.60529,1.61259).
indice3_refract(sk51,1.62807,1.62866,1.63368,1.63835, 1.64633).
indice3_refract(sk52,1.64655,1.64721,1.65284,1.65808, 1.66705).
indice3_refract(sk55,1.62757,1.62816,1.63314,1.63775, 1.64559).
indice3_refract(sk6,  1.62134,1.62196,1.62731,1.63227,1.64078).
indice3_refract(sk7,  1.61440,1.61498,1.61995,1.62455,1.63240).
indice3_refract(sk8,  1.61880,1.61942,1.62479,1.62979,1.63837).
indice3_refract(sk9,  1.62182,1.62246,1.62795,1.63306,1.64185).
indice3_refract(sskn5,1.66750,1.66825,1.67471,1.68080,1.69143).
indice3_refract(sskn8,1.62641,1.62713,1.63336,1.63925,1.64959).
indice3_refract(ssk1,1.62520,1.62586,1.63152,1.63681, 1.64595).
indice3_refract(ssk2,1.63048,1.63116,1.63696,1.64240, 1.65180).
indice3_refract(ssk3,1.62325,1.62395,1.62995,1.63559, 1.64539).
indice3_refract(ssk4,1.62547,1.62611,1.63163,1.63677, 1.64561).
indice3_refract(ssk50,1.62617,1.62685,1.63268,1.63816,1.64765).
indice3_refract(ssk51,1.61147,1.61212,1.61770,1.62292,1.63196).
indice3_refract(ssk52,1.66749,1.66824,1.67468,1.68072,1.69122).
indice3_refract(tifn5,1.60538,1.60639,1.61529,1.62405, 1.64042).
indice3_refract(tif1,  1.51820,1.51878,1.52384,1.52866,1.53726).
indice3_refract(tif2,  1.54070,1.54138,1.54734,1.55309,1.56356).
indice3_refract(tif3,  1.55680,1.55757,1.56433,1.57091,1.58300).
indice3_refract(tif4,   1.59521,1.59616,1.60452,1.61274,1.62804).
indice3_refract(tif6,  1.63070,1.63194,1.64308,1.65448,     _ ).
indice3_refract(tik1,  1.48436,1.48483,1.48880,1.49249,1.49882).
indice3_refract(tisf1, 1.68997,1.69141,1.70417,1.71698,1.74164).
indice3_refract(ubk7,1.52237,1.52282,1.52667,1.53022, 1.53623).
indice3_refract(uk50,1.52858,1.52907,1.53327,1.53715, 1.54377).
indice3_refract(zkn7,1.51423,1.51470,1.51869,1.52238, 1.52864).
indice3_refract(zk1,  1.53955,1.54008,1.54457,1.54875,1.55590).
indice3_refract(zk5,  1.54049,1.54104,1.54579,1.55023,1.55784).
```