The Eikonal+ Backend

Adam Hayes
The Institute of Optics
University of Rochester
402 Goergen Hall
abraunhayes@gmail.com

Daniel K. Nikolov
The Institute of Optics
University of Rochester
504 Goergen Hall

February 5, 2019

Contents

Abstract

This is the draft manual for the `Eikonal+` Backend

1 Introduction

The Eikonal+ application is divided into a frontend for most purposes and a backend, which can be controlled directly by text commands, or interfaced to other software. Almost any language or major scientific package can launch Eikonal+ as a subprocess and communicate with it. Communication with the backend is done via the stdin (standard in) and stdout (standard out) streams. Stdin is the keyboard if the backend is launched from a terminal ("from the command line").

Additionally, a SIGINT (usually CTRL-C) can be used to interrupt the backend during some calculations without crashing Eikonal+. (This is not fully implemented.)

A log file is overwritten at the start of execution.

Repetitive calculations can be run using scripts. A script can be piped to stdin when starting the backend, or one can be called during an interactive session. This can be useful in preventing time-consuming mistakes, such as when the source of a problem with the design is not known.

General notes from the original Eikonal manual:

- Always optimize axial focus before starting designing (1S1=0).
- Unless you really start from scratch it is best to correctly fill the aperture stop with off axis pencils of rays using codes PE, PS, 1PE, 1PS at the very beginning of the design.
- Never forget to run computation of parameter increments with code PARI once the optimization variables (construction parameters) have been defined with code PARA, and repeat computation frequently ever after.
- When using the Adaptive Method bring aberrations, functions or constraints to required targets one at a time, unless the residual errors are very small or the entities targeted are completely independent.
- Before using the method of constrained damped least squares (DLS) all functions and constraints must be brough to target with the Adaptive Method for the program to run efficiently.

1.1 Optical system limitations

The following limits apply to systems designed in the front end and in the back end.

Number of configurations in a lens system	5	(from 0 to 4)
Number of optimization parameters	199	
Number of targeted aberrations	199	
Actual number of constraints	117	
Number of surfaces in each configuration	100	(from 0 to 99)
Number of field points in each configuration	15	(from 0 to 14)
Number of groups in each configuration	10	
Number of design wavelengths in each configuration	3	

2 Benchmarking

2.1 Ray Tracing

Without optimization in the compiler, using *a single thread* on the Linux machine described below, the following speed values were obtained. Note that these rays covered a square grid. A circular grid would make use of only $\approx 78\%$ of these points. Currently, Eikonal+ traces rays only in square grids, but this is easy to change if the 22% improvement in speed is required.

Eikonal+ traced 144k rays / second through a Cooke Triplet system, which includes 6 "real" surfaces, or 9 surfaces including the image and reference plane and the aperture stop.

With parallelization, this speed should increase linearly with the number of simultaneous threads.

Including writing to disk on a common laptop, the speed was reduced to 39k rays / second for the same Cooke Triplet system.

System details for the tests above:

```
product          : MacBookPro9,2 (System SKU#)
operating system : Ubuntu Linux 14.04
vendor           : Apple Inc.
version          : 1.0
width            : 64 bits
*-cpu            : 0
description      : CPU
product          : Intel(R) Core(TM) i7-3520M CPU @ 2.90GHz
```

See the `RAYTRA` command for instructions on ray tracing.

3 Setup

The backend is configured during installation of the full Eikonal+ package. The Eikonal+ backend can be configured independently by editing the `EPLUSCONFIG.IKT` file in a text editor.

```
! Eikonal+ namelist format!
&DIRECTORIES
PDIRG = 'C:\EikonalLibs',
DOCS_DIR = 'C:\Eikonal\eikonal\docs'/
```

(This is a Fortran "namelist" file.)

4 Eikonal+ backend commands

4.1 Getting help

A brief help string is available when communicating directly with the backend. The syntax is 'HELP KEYWORD', where KEYWORD is a search term or command name.

4.2 Command syntax

Optional parameters are enclosed in square brackets `[ ]`, and required parameters are enclosed in angle brackets `<>`.

Generally, commands are formed from the first three letters of each word in the operation or quantity, e.g. `FIRORDPAR` is "first order parameter." Exceptions:

```
WRITE_NOTE
WRITE_LOG
WRITE_WARNING
WRITE_ERROR
WHAT
```

Old style Eikonal commands must be preceded by '$'.

```
$RT
```

`$RTGL`

`BACSYS`

Makes a *temporary* backup of the lens system in memory for the user. The backup includes more than the lens definitions and thus is different from the .DBS and .A files. This file is destroyed upon restarting Eikonal+. See also `RESSYS`.

`CD`

Changes the working directory of the back end. Note that it will fail silently if the directory is badly formed or not found. On Linux / Unix / OS X systems, use the forward slash to separate directories.

```
CD /home/adam/MyTelescope
```

Be sure to use "\" (back slash) instead of forward slash on Windows systems. Example:

```
CD "C:\Users\Adam\My Telescope"
```

For directories with spaces in them, surround the entire path with quotation marks:

```
CD "/home/adam/My Telescope"
```

`CHAREFLECSUR`

Changes the properties of a reflective surface in the specified configuration. See also `INSREFLECSUR`. Usage:
`CHAREFLECSUR <c> <sn> <r> <dp> <dn> [title]`
where `c` is the configuration number (starting from 0),
`sn` is the index of the surface to change,
`r` is the new radius of curvature,
`dp` is the new distance from the previous surface (surface `sn` - 1),
`dn` is the new distance to the next surface,
`[title]` is an optional string (no spaces, e.g. 'PRIMARY MIRROR').
Check the inserted surface using `LISSUR`.

`CHAREFRACSUR`

Changes the properties of a refractive surface in the specified configuration.

See also INSREFRACSUR. Usage:
CHAREFRACSUR <c> <sn> <r> <dp> <dn> <g> [title]
where c is the configuration number (starting from 0),
sn is the index of the surface to change,
r is the new radius of curvature,
dp is the new distance from the previous surface (surface sn - 1),
dn is the new distance to the next surface,
g is the new glass code from the catalog (8 characters max, e.g. 'BK7'... and [title] is an optional string (no spaces, e.g. 'PART-OF-CEMENTED-DOUBLET').
Check the inserted surface using LISSUR.

CHDIR

See CD.

COLLABORATOR2A

Prints a short summary of the system surfaces, glass codes, etc., followed by the 3rd and 5th order Seidel-Buchdahl aberration coefficients.

DANSYSDAT8

Produces a table of the surfaces in the lens system. This is used by the front end.

DANSYSDAT9

This is the same as DANSYSDAT8, except that the output is written to systemdata.dat in the working directory.

DUMP

Dumps all system data in binary format to the file DUMP.BIN. See also RESSYS and BAKSYS.

$EVAL

$EVAL performs a number of functions that need to be run following any change to the optical system. Some of its functions are

- Calculating the semi-apertures for new surfaces
- ...

`FFD`

Creates optical path difference (OPD) plot data for a full-field display (FFD). In the `.../examples/full_field_displays/` directory is an example input file `coords_for_ffd.txt` giving a set of polar coordinates used for both field points and pupil points.

On the first line of the input file is a pair of integers, the number of radial coordinates and the number of azimuthal coordinates, respectively. Eikonal+ will cycle through all permutations of these coordinates for field points and pupil points. Following the first line are the radial coordinates (one per line) in system units and then the azimuthal coordinates in radians.

```
7, 25
0.15951816143819091089225221066676
0.35949187362206503717137258188947
0.54504809357643057125091582559565
0.70710678118654752440084436210485
0.83840478033507095662135338736221
0.93314821587982325195673413572177
0.98719499399631239330569439839602
0
0.25132741228718345907701147066236
0.50265482457436691815402294132472
0.75398223686155037723103441198708
1.0053096491487338363080458826494
1.2566370614359172953850573533118
1.5079644737231007544620688239742
1.7592918860102842135390802946365
2.0106192982974676726160917652989
2.2619467105846511316931032359612
2.5132741228718345907701147066236
2.7646015351590180498471261772 86
3.0159289474462015089241376479483
3.2672563597333849680011491186107
3.5185837720205684270781605892 73
3.7699111843077518861551720599354
4.0212385965949353452321835305978
4.2725660088821188043091950012601
4.5238934211693022633862064719225
```

```
4.775220833456485722463217942584 8
5.026548245743669181540229413247 2
5.277875658030852640617240883909 6
5.529203070318036099694252354571 9
5.780530482605219558771263825234 3
6.031857894892403017848275295896 6
```

The output from the FFD calculation can be used in the Matlab code in the same examples directory to create a plot. The format of the output file is as follows. The first line shows the half-field of view values of the system (in *degrees*) with a comment. All remaining lines of the output file contain the fields r_{fp} (radius of field point), θ_{fp}, (polar angle of field point in *radians*) r_{pp} (radius of the pupil point), θ_{pp} (polar angle of the pupil point), d (the OPD value in wavelengths), in that order.

Below is an example of an output file.

```
0.000000000   0.000000000 HFOV OF SYSTEM FOR Y AND Z (DEGREES)
0.159518161   0.000000000   0.159518161   0.000000000  0.43902704E-02
0.159518161   0.000000000   0.159518161   0.251327412  0.65348162E-02
0.159518161   0.000000000   0.159518161   0.502654825  0.12396494E-01
```

Errors are noted in the file next to each entry:

```
0.933148216   4.021238597   0.545048094   2.764601535  0.21649860E+02 <WARNING>  R
0.933148216   4.021238597   0.545048094   3.015928947  0.21649860E+02 <WARNING>  W
```

Syntax: `FFD <optical system file> <input file> <output file>`

`FINGLA`

Looks up glass data by the name, *e.g.* "BK7". The name currently has a maximum length of 8 characters.

`GETCWD`

Prints the working directory of the back end.

`GETGENSYSSET`

Display the following settings, as appropriate.

```
Image distance solve ON
Image distance solve OFF
Objective
Common relay
Afocal relay
Telescope
Eyepiece
Autocollimator (finite)
Autocollimator (infinite)
Rotationally symmetric system
Anamorphic system
Symmetric image blur
Asymmetric image blur
Circular or elliptical pupil
Square or rectangular pupil
Strictly collinear System
Non-collinear System (explicit; dummy sfces)
Non-collinear System (implicit; hidden sfces)
Standard coordinate system
Canonical coordinate system
```

`INSFIEPOI` or `INSFLDPNT`

Adds a field point to the specified configuration. Usage:
`INSFIEPOI <"FIH" or "HFOV"> <configuration #> <y-value> <z-value>`
where `FIH` and `HFOV` indicate that the values to follow are fractional image heights or half-field-of-view values, respectively. See also `LISFIEPOI` and `REMFIEPOI`.

`INSREFLECSUR`

Inserts a reflective surface in the lens system. Usage:
`INSREFLECSUR <c> <p> <r> <dp> <dn> [title]`
where `c` is the configuration number (starting from 0),
`p` is the index of the surface after which to insert this surface (starting from 0),
`r` is the radius of curvature,
`dp` is the distance from the previous surface (surface p),
`dn` is the distance to the next surface,
`[title]` is an optional string (no spaces, e.g. 'PART-OF-CEMENTED-

DOUBLET'.
NOTES:

1. In order to propagate changes in lens thicknesses (surface positions) `$EVAL` should be run following `INSREFLECSUR`.

2. The distance from the previous surface generally should match the value of "`Dist to next`" for the previous surface in the `LISSUR` output. (See the example below.)

3. The distance to the next surface should match in sign the direction light will be traveling after leaving the new surface. The user must do this sensibly.

4. `Eikonal+` takes care of signs of the refractive indices internally.

5. Check the results with `LISSUR` to be sure that the new surface is in the correct position.

The transcript below demonstrates the use of `INSREFLECSUR` to make a two-mirror system. The system is initially a single spherical mirror (surface 1) 4.0 cm from the reference surface, reflecting light 2.0 cm back toward the reference surface. When the next surface is added after surface 1, the distance of 2.0 cm can be changed using the `<dp>` field.

```
lissur 0
 //   Type |   Curvature | Dist to next   | Refr indx | Gls code | Glass
 //-------------------------------------------------------------------
    0 SP      0.00000000      4.00000000    1.000000      0.000
    1 SP     -0.12500000     -2.00000000   -1.000000      0.000
    2 SP      0.00000000      0.00000000    1.000000      0.000
 <END>
```

To add another mirror 2.0 cm after the reference surface (physically between surfaces 0 and 1) reflecting rays back toward surface 1, insert the new mirror *after* surface 1. Below, a spherical mirror is added after surface 1 using dp = -2.0. Using a different desired value for dp would change the "`Dist to next`" value for surface 1 in the `LISSUR` output.

```
insreflecsur 0 1 8 -2 4
 <END>
lissur 0
 //   Type |   Curvature | Dist to next   | Refr indx | Gls code | Glass
 //-------------------------------------------------------------------------
    0 SP      0.00000000      4.00000000    1.000000      0.000
    1 SP     -0.12500000     -2.00000000   -1.000000      0.000
    2 SP      0.12500000      4.00000000    1.000000      0.000
    3 SP      0.00000000      0.00000000    1.000000      0.000
 <END>
```

The <dn> value was entered as 4.0 cm, which can be changed when inserting a surface after surface 2. Use $EVAL to update computed parameters. Running $EVAL again and listing the surfaces gives the following output. The $LENS command can be used to see the finished system (Figure 1).

```
$eval
  CURRENT EVALUATION:
                     W(rms)1   W(rms)2   W(rms)3    Distrn.   1Lt.Clr.   2Lt.Clr.
  Config. 0:
    fld. pt.   0   2580.0456 2917.4648 2174.9692    0.0000    0.0000    0.0000
    fld. pt.   1   2580.0458 2917.4651 2174.9694   -0.0015    0.0000    0.0000
    fld. pt.   2   2580.0460 2917.4653 2174.9696   -0.0022    0.0000    0.0000

 <END>
lissur 0
 //   Type |   Curvature | Dist to next   | Refr indx | Gls code | Glass
 //-------------------------------------------------------------------------
    0 SP      0.00000000      4.00000000    1.000000      0.000
    1 SP     -0.12500000     -2.00000000   -1.000000      0.000
    2 SP      0.12500000      4.00000000    1.000000      0.000
    3 SP      0.00000000      0.00000000    1.000000      0.000


 <END>
$lens
```

INSREFRACZERSUR

Inserts a refractive Zernike polynomial surface in the lens system. NOTE:

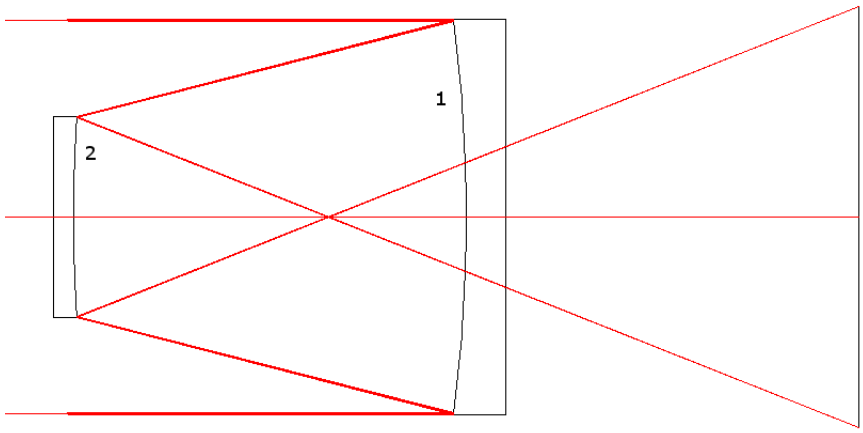

Figure 1: The two-mirror system created using INSREFLECSUR commands. The numbered surfaces correspond to the final system.

In order to propagate changes in lens thicknesses (surface positions) `$EVAL` should be run following `INSREFRACZERSUR`.
Usage:

`INSREFRACZERSUR <c> <p> <r> <dp> <dn> <su>` <$a_1, a_2, ..., a_{20}$> `<g> [title]`
where `c` is the configuration number (starting from 0),
`p` is the index of the surface after which to insert this surface (starting from 0),
`r` is the radius of curvature to first order (the sperical term),
`dp` is the distance from the previous surface (surface p),
`dn` is the distance to the next surface,
`su` is the radius over which the Zernike polynomials are defined†,
$a_1, a_2, ..., a_{20}$ are the coefficients of the Zernike polynomial ‡,
`g` is the glass code from the catalog (8 characters max, e.g. 'BK7'... and
`[title]` is an optional string (no spaces, maximum 40 characters, e.g. 'PART-OF-CEMENTED-DOUBLET'.

Check the inserted surface using `LISSUR`.

†The `<su>` parameter is the radius in the lens plane (not the radius of curvature) that will be treated as a unit circle over which the Zernike poly-

nomials are defined.

‡The numbering of the Zernike polynomials is given in Table 1. [Put the table in the common-parts directory]

INSREFRACSUR

Inserts a refractive surface in the lens system. NOTE: In order to propagate changes in lens thicknesses (surface positions) `$EVAL` should be run following `INSREFRACSUR`.

Usage:

`INSREFRACSUR <c> <p> <r> <dp> <dn> <g> [title]`

where `c` is the configuration number (starting from 0),
`p` is the index of the surface after which to insert this surface (starting from 0),
`r` is the radius of curvature,
`dp` is the distance from the previous surface (surface p),
`dn` is the distance to the next surface,
`g` is the glass code from the catalog (8 characters max, e.g. 'BK7'... and `[title]` is an optional string (no spaces, e.g. 'PART-OF-CEMENTED-DOUBLET'.

Check the inserted surface using `LISSUR`.

LISFIEPOI or LISFLDPNT

Lists field points for the specified configuration. Usage:

`LISFIEPOI <configuration #>`

The output should look similar to the following:

```
            FRCT. IMG. HGT      PUPIL SHIFT COEFF.      PUPIL EXPN. COEFF.
            Weight  Y-axis    Z-axis     Y-axis     Z-axis       Y-axis     Z-axis
0:BASIC
    fld.,  1    +  0.700000  0.000000  0.00000000  0.00000000  1.00000000  1.00000
                                       0.00000000  0.00000000  1.00000000  1.00000
                                       0.00000000  0.00000000  1.00000000  1.00000
    fld.,  2    +  0.700000  0.000000  0.03067568  0.00000000  0.96262707  0.99930
                                       0.03067568  0.00000000  0.96262707  0.99930
                                       0.03067568  0.00000000  0.96262707  0.99930
    fld.,  3    +  1.000000  0.000000 -0.00225698  0.00000000  0.79630930  0.97903
```

```
-0.00225698  0.00000000  0.79630930  0.97903
-0.00225698  0.00000000  0.79630930  0.97903
```

See also ADDFIEPOI and REMFIEPOI.

LOASYS

Loads an optical system from a file. Usage: LOASYS "<file-path>", e.g. LOASYS "/home/mary/optics/telescope.dbs". The surrounding quotation marks are required for files not in the current directory. LOASYS loads both .DBS and .A files. See SAVSYS.

OPETEXFIL

Opens a user text file. Usage: OPETEXFIL filename unit#. See CLO.

RAYTRA

Traces rays through the optical system. Usage: RAYTRA <wavelengths> <last obj point> <pupil points>, where <last obj point> is the index (starting from 0) of the last field point to be used. Field points will be used cyclically if more are requested than are currently defined. The number of rays traced is the square of the <pupil points> argument (for a circular aperture).

REMFIEPOI

Removes a field point from a configuration of the lens system. Usage: REMFIEPOI <c> <fn>, where <c> is the configuration number (starting from 0), and <fn> is the field number to remove (starting with 0). Note that after removing a field point the rest of the field points will be re-ordered with respect to an increasing fractional image height.

REMSUR

Removes a surface from a configuration of the lens system. Usage: REMSUR <c> <sn>, where <c> is the configuration number (starting from 0), and <sn> is the surface number to remove (starting with 0).

RESSYS

Restores the user's *temporary* backup of the lens system. This file is destoyed upon restarting Eikonal+. See also BAKSYS.

SAVSYS

Saves the optical system in memory to a file. Usage: SAVSYS "<file-path>", e.g. SAVSYS "/home/mary/optics/telescope.dbs". The surrounding quotation marks are required for files not in the current directory. SAVSYS works with both .DBS and .A files. See LOASYS.

SETIMADISSOL <configuration #> <'yes' or 'no'>

Turns on ('yes') or off ('no') solving for the image distance when command $EVAL is run.
configuration # is the number of the configuration to set the image-distance solve flag for. The numbering begins with 0.

SETABESYM <configuration #> <symmetry code>

where <symmetry code> is one of "SYM" and "ASYM." This command sets the appropriate flags and other values to change the ray tracing from half-pupil to full-pupil, used for rotationally symmetric ("SYM") and asymmetric ("ASYM") aberrations (blur), respectively.

SETROTSYM <configuration #> <symmetry code>

where <symmetry code> is one of "RS", "NRS", "NSRT" (rotational symmetry, non-rotational symmetry and non-symmetric ray tracing, respectively). This command sets the appropriate flags and other values to change the system's symmetry, which controls the methods used by Eikonal+ for ray tracing.

SHOCOD

Prints the complete list of Eikonal (not Eikonal+) command codes to stdout.

STASCR <filename>

Begins taking input from the script file. Use STOSCR to return to standard input. STOSCR must be in the script file, or Eikonal will hang. This is to be used in interactive mode, *not* when Eikonal is fed a script at startup. See STOSCR.

STOSCR

Stops a script in progress and returns to standard input. This is to be used in a script file that is called in interactive mode, *not* when a script is piped to Eikonal+ at startup. See STASCR.

`SURMOV` Moves (displaces, decenters) one or more surfaces. In order to

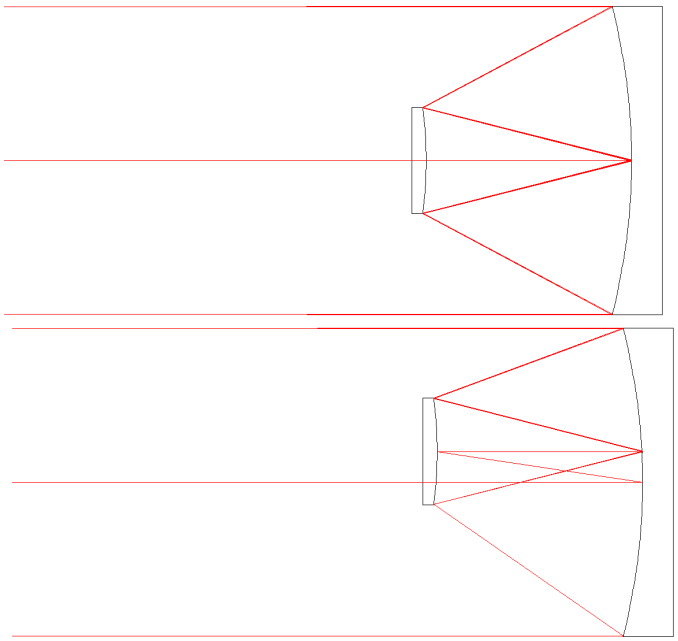

Figure 2: A two-mirror system before (top) and after (bottom) a displacement using SURMOV.

move surface n, insert the displacement with `SURMOV` before surface n. Note: surfaces cannot be moved along the optical axis. Instead, use `CHAREFRACSUR` or `CHAREFLECSUR`. *Removing and replacing tilts and displacements is not robust in Eikonal or Eikonal+. It may be helpful to write a script that builds the system, in case you need to start from scratch.* Usage:
`SURMOV <c> <n> <d> <axis>`
where `c` is the configuration number (starting from 0),
`n` is the index of the surface to move,
`d` is the distance in system units,
`axis` is one of "x," "y," or "z," the pivot axis.
Example: The following script will displace the first mirror (bottom of Figure 2) from the original system (top of Figure 2). The following script will reproduce Figure 2.

```
loasys 111113.dbs
```

```
$eval
$lens
setimadissol 0 yes
setrotsym 0 nrs
lissur 0
surmov 0 1 0.2 y
$eval
lissur 0
$lena
stoscr
```

The surface list before and after the displacement are shown below. Note that the first mirror is on the *right* and the second mirror is on the *left*.

```
//   Type |   Curvature | Dist to next   | Refr indx | Gls code | Glass
//-------------------------------------------------------------------
0 SP      0.00000000      4.09206570    1.000000      0.000
1 A1     -0.25000000     -1.33333300   -1.000000      0.000
2 A1     -0.37499981      1.33333300    1.000000      0.000
3 SP      0.00000000      0.00000000    1.000000      0.000

//   Type |   Curvature | Dist to next   | Refr indx | Gls code | Glass
//-------------------------------------------------------------------
0 SP      0.00000000      4.09206570    1.000000      0.000
1 A1     -0.25000000     -1.00000000   -1.000000      0.000
2 YD      0.20000000     -0.33333300   -1.000000      0.000
3 A1     -0.37499981      1.33333399    1.000000      0.000
4 SP      0.00000000      0.00000000    1.000000      0.000
```

`SURTIL` Tilts (rotates) one or more surfaces. In order to rotate surface n, insert the tilt with `SURTIL` before surface n. *Removing and replacing tilts and displacements is not robust in Eikonal or Eikonal+. It may be helpful to write a script that builds the system, in case you need to start from scratch.*
Usage:
`SURTIL <c> <n> <a> <p> <axis>`
where `c` is the configuration number (starting from 0),
`n` is the index of the surface to tilt,
`a` is the tilt angle in degrees,
`p` is the distance from surface n to the pivot axis and

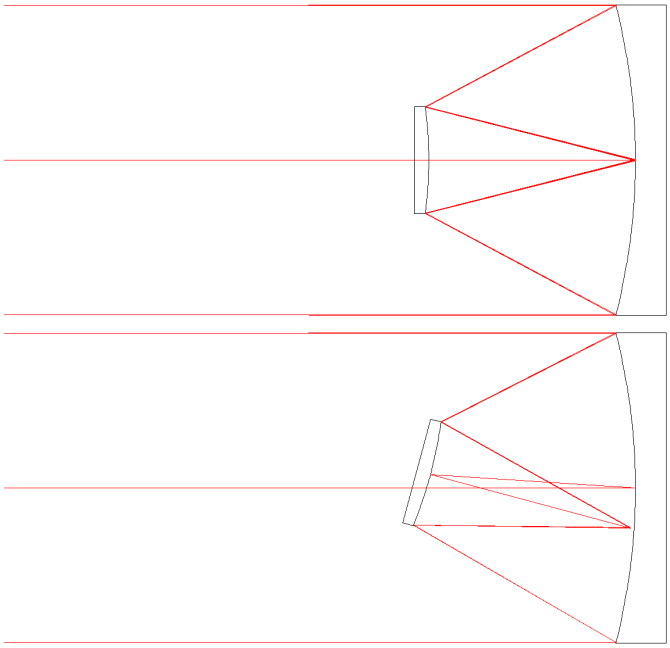

Figure 3: A two-mirror system before (top) and after (bottom) a tilt using SURTIL.

`axis` is one of "x," "y," or "z," the pivot axis.
Example: The following script will produce the tilted system (bottom of Figure 3) from the original system (top of Figure 3). The following script will reproduce Figure 3.

```
loasys 111113.dbs
$eval
$lens
setimadissol 0 yes
setrotsym 0 nrs
lissur 0
surtil 0 1 -15 0 y
$eval
lissur 0
$lena
stoscr
```

The surface list before and after the tilt are shown below. Note that the first mirror is on the *right* and the second mirror is on the *left*.

```
//   Type |   Curvature | Dist to next   | Refr indx | Gls code | Glass
//-------------------------------------------------------------------------
0 SP      0.00000000      4.09206570    1.000000      0.000
1 A1     -0.25000000     -1.33333300   -1.000000      0.000
2 A1     -0.37499981      1.33333300    1.000000      0.000
3 SP      0.00000000      0.00000000    1.000000      0.000

//   Type |   Curvature | Dist to next   | Refr indx | Gls code | Glass
//-------------------------------------------------------------------------
0 SP      0.00000000      4.09206570    1.000000      0.000
1 A1     -0.25000000     -1.00000000   -1.000000      0.000
2 YD      0.20000000     -0.33333300   -1.000000      0.000

3 A1 -0.37499981 1.33333399 1.000000 0.000

4 SP      0.00000000      0.00000000    1.000000      0.000
```

`CLO n`

Closes the file with unit number `n`.

`LISCON`

Prints a list of all configurations of the current system in memory.

`LISGLACAT <n>`

Prints a list of glasses in one of the catalogs (n) below.

- 0 Schott
- 1 Ohara
- 2 S-Ohara
- 3 Hoya
- 4 Chance and Sovirel
- 5 Miscellaneous (IR materials)

- 6 Schott (obsolete)
- 7 Schott radiation resistant
- 8 for Selfoc materials
- 9 for S-ohara (Sellmeir formula)

`LISKIN`
Prints a list of all surface types. (List kinds)

`NEWSYS`
Creates an empty lens *system* containing one configuration.

4.3 GET and SET commands

`GET` and `SET` retrieve and set a stored parameter or variable. Some quantities may be calculated by `GET` each time they are retrieved, and there will be no corresponding `SET` call. The exact syntax depends on the parameter or variable. See for example `FIRORDPAR`.

`FIRORDPAR`
Use with `GET` to access or calculate a first-order parameter, such as the numerical apertures, pupil sizes, etc. For systems that are not strictly collinear, the values returned by this function may not accurately describe the first-order properties of the lens because the system is not strictly collinear.' Syntax: `GET FIRORDPAR [configuration number] [M or E] [parameter name]`
Either none or both of [configuration number] and [parameter name] must be given. If neither are given, then a report of all applicable first-order parameters is returned. For example, `GET FIRORDPAR 0 M EXIPUPSIZ` would return the exit pupil size for configuration 0 of the system, in the meridional plane. If the system is not collinear, the meridional or equatorial plane must bespecified using `M` or `E`. If the system is collinear, then the meridional plane (`M`) must be specified.
The following first-order parameters are available.

```
GET FIRORDPAR 0 M UNI
GET FIRORDPAR 0 M EXIANGHALFIE
GET FIRORDPAR 0 M ENTANGHALFIE
GET FIRORDPAR 0 M IMASPANUMAPE
```

```
GET FIRORDPAR 0 M OBJSPANUMAPE
GET FIRORDPAR 0 M IMASPAFNUM
GET FIRORDPAR 0 M OBJSPAFNUM
GET FIRORDPAR 0 M ANGDEM
GET FIRORDPAR 0 M ANGMAG
GET FIRORDPAR 0 M LATDEM
GET FIRORDPAR 0 M LATMAG
GET FIRORDPAR 0 M EXIPUPSIZ
GET FIRORDPAR 0 M ENTPUPSIZ
GET FIRORDPAR 0 M IMAHEI
GET FIRORDPAR 0 M OBJHEI
GET FIRORDPAR 0 M SECNODPOIDIS
GET FIRORDPAR 0 M FIRNODPOIDIS
GET FIRORDPAR 0 M SECPRIPLADIS
GET FIRORDPAR 0 M FIRPRIPLADIS
GET FIRORDPAR 0 M SECPRIRAYFOCDIS
GET FIRORDPAR 0 M FIRPRIRAYFOCDIS
GET FIRORDPAR 0 M EXIPUPDIS
GET FIRORDPAR 0 M ENTPUPDIS
GET FIRORDPAR 0 M IMADIS
GET FIRORDPAR 0 M OBJDIS
GET FIRORDPAR 0 M PETSUM
GET FIRORDPAR 0 M LAGINV
GET FIRORDPAR 0 M OBJIMADIS
GET FIRORDPAR 0 M PUPPUPDIS
GET FIRORDPAR 0 M VERVERDIS
GET FIRORDPAR 0 M EQUFNUM
GET FIRORDPAR 0 M EQUFOCLEN
GET FIRORDPAR 0 M LASSUR
GET FIRORDPAR 0 M FIRSUR
GET FIRORDPAR 0 M DES
GET FIRORDPAR 0 M CONTAG
GET FIRORDPAR 0 M NAM
```

`INSTR n [filename]`

Begins taking input from unit `n`, which may or may not be a regular file. If a file name is given, then the file is opened and assigned a unit number of

`n`. Subsequent changes of the input stream do *not* close the file. Use `CLO` to close a file.

`OUTSTR <n> [filename]`
Redirects output to file unit `n`, which may or may not be a regular file. If a file name is given, then the file is opened and assigned a unit number of `n`. Subsequent changes of the output stream do *not* close the file. Use `CLO` to close a file.

The EAF entries need cleanup.

`EAF <configuration #> <"M" or "E">`
`GET`: Returns the object-space half-angle field in degrees.
`EAF <configuration #> <"M" or "E">`
`SET`: The first required parameter is a flag, either "1" or "2," for the meridional or equatorial plane, respectively. The meridional and equatorial EAF values should *both* be set by the user for a non-rotationally symmetric system; changing one does not update the other. For a rotationally symmetric system, the user *must* set the half-angle for the meridional plane. Eikonal+ will then set the same EAF for the equatorial plane. The half-angle is then given in degrees. Use `GET FIRORDPAR` to check the value of EAF.

`XFN <ME> <f-number>`
Sets the image-space f-number. The first required parameter is a flag, either "1" or "2," for the meridional or equatorial plane, respectively. The meridional and equatorial XFN values should *both* be set by the user for a non-rotationally symmetric system; changing one does not update the other. For a rotationally symmetric system, the user *must* set the f-number for the meridional plane. Eikonal+ will then set the same f-number for the equatorial plane. The f-number is then given as a float. Use `GET FIRORDPAR` to check the value of XFN.

5 Common User Procedures

5.1 Constructing a lens system

Generally, it is best to define systems by adding surfaces numbered consecutively in ascending order. As an example, take a base system with a reference surface (0), an aperture stop (1) and an image plane (2). Surfaces could be added by inserting after surface 1, then inserting after surface 2, etc.

Note: It is helpful to define a system by writing a script that can be run repeatedly until the system is set up properly. (See the command STASCR.) This will avoid situations where a mistake forces the user to start from scratch by hand.

It may be helpful to start with one of the systems in the Eikonal+ library. (More info on this to come.) Alternatively, a good starting point is to load the empty system InitLens.A. It contains only one field point, an aperture stop and the reference surface.

5.2 Changing surfaces

The command CHAREFRACSUR will change the parameters of a refractive surface. An alternative is to insert a new serface before the one that is to be changed with the new parameters and then remove the original surface. CHAREFRACSUR does both of these operations at once.

Below is an example of a sequence of insert and remove commands that will add, then change a surface. A surface of radius 10 is added. Then a replacement surface of radius 11 is added. (The simplest way to make a replacement is to add the new surface before removing the old one.) Last, the surface of radius 10 is removed. See commands INSREFRACSUR, REMSUR, LISSUR. ,,.

```
 <END>
loasys triplet.a
 READING FROM FILE: TRIPLET.A **IN WORKING DIRECTORY**

                    Current Lens System
 Cooke Triplet f/4.5
 TRIPLE      23 Apr2012    10:  47:  32
```

```
 0:BASIC


    ECP       EPD       ENP       QFL       XNP       XPD       XCP
**********    8.2156   10.4323   50.0007   -6.9498   -9.2694   43.0508
    ECH       EPS       EFN       QFN       XFN       XPS       XCH
**********  11.1113**********    4.5000    4.5000   11.6267   18.1988
    EMX       EAF       EVD       VTV       XVD       XAF       XME
    0.0000   20.0000   58.5008   15.4500    0.0000   19.1794**********
    ASS       ASD       ASN       CTC       FSS       FSD       FSN
    8.7367    0.0000    3.0000**********    0.0000    0.0000    0.0000

 WARNING:
 *) Please run code EVAL (evaluation of system)
    before using any codes related to optimization.
 <END>
lissur 0
 //   Type |   Curvature | Dist from last | Refr indx | Gls code | Glass
 //-----------------------------------------------------------------------
    0 SP      0.00000000     10.00000000    1.000000      0.000
    1 SP      0.04655190      2.00000000    1.622861    620.603   SK16
    2 SP     -0.00805802      5.26000000    1.000000      0.000
    3 AS      0.00000000      0.00000000    1.000000      0.000
    4 SP     -0.05235602      1.25000000    1.620578    617.366   F4
    5 SP      0.04545454      4.69000000    1.000000      0.000
    6 SP      0.00304044      2.25000000    1.622861    620.603   SK16
    7 SP     -0.05988024     43.05082831    1.000000      0.000
    8 SP      0.00000000      0.00000000    1.000000      0.000
 <END>
insrefracsur 0 2 10 3 4 BK7
 <END>
lissur 0
 //   Type |   Curvature | Dist from last | Refr indx | Gls code | Glass
 //-----------------------------------------------------------------------
    0 SP      0.00000000     10.00000000    1.000000      0.000
    1 SP      0.04655190      2.00000000    1.622861    620.603   SK16
    2 SP     -0.00805802      3.00000000    1.000000      0.000
    3 SP      0.10000000      4.00000000    1.518721    517.642   BK7
```

```
    4 AS      0.00000000      0.00000000    1.000000      0.000
    5 SP     -0.05235602      1.25000000    1.620578    617.366   F4
    6 SP      0.04545454      4.69000000    1.000000      0.000
    7 SP      0.00304044      2.25000000    1.622861    620.603   SK16
    8 SP     -0.05988024     43.05082831    1.000000      0.000
    9 SP      0.00000000      0.00000000    1.000000      0.000
 <END>
insrefracsur 0 2 11 3 4 BK7
 <END>
lissur 0
 //   Type |   Curvature | Dist from last | Refr indx | Gls code | Glass
 //-------------------------------------------------------------------------
    0 SP      0.00000000     10.00000000    1.000000      0.000
    1 SP      0.04655190      2.00000000    1.622861    620.603   SK16
    2 SP     -0.00805802      3.00000000    1.000000      0.000
    3 SP      0.09090909      4.00000000    1.518721    517.642   BK7
    4 SP      0.10000000      4.00000000    1.518721    517.642   BK7
    5 AS      0.00000000      0.00000000    1.000000      0.000
    6 SP     -0.05235602      1.25000000    1.620578    617.366   F4
    7 SP      0.04545454      4.69000000    1.000000      0.000
    8 SP      0.00304044      2.25000000    1.622861    620.603   SK16
    9 SP     -0.05988024     43.05082831    1.000000      0.000
   10 SP      0.00000000      0.00000000    1.000000      0.000
 <END>
remsur 0 4
          STATUS= 0:BASIC  is a Strictly collinear System

     ECP       EPD       ENP       QFL       XNP       XPD       XCP
 **********    8.2156    6.9848   19.4559  -11.4688  -10.3112    7.9872
     ECH       EPS       EFN       QFN       XFN       XPS       XCH
 **********  11.1113**********    1.7510    1.7510   10.4502    7.0814
     EMX       EAF       EVD       VTV       XVD       XAF       XME
     0.0000   20.0000   25.1772   17.1900    0.0000   21.1561**********
     ASS       ASD       ASN       CTC       FSS       FSD       FSN
     7.8953   -0.8983    4.0000**********   14.1628    7.9872    8.0000
 <END>
lissur 0
 //   Type |   Curvature | Dist from last | Refr indx | Gls code | Glass
```

```
//--------------------------------------------------------------------
   0 SP      0.00000000     10.00000000    1.000000      0.000
   1 SP      0.04655190      2.00000000    1.622861    620.603   SK16
   2 SP     -0.00805802      3.00000000    1.000000      0.000
   3 SP      0.09090909      4.00000000    1.518721    517.642   BK7
   4 AS      0.00000000      0.00000000    1.000000      0.000
   5 SP     -0.05235602      1.25000000    1.620578    617.366   F4
   6 SP      0.04545454      4.69000000    1.000000      0.000
   7 SP      0.00304044      2.25000000    1.622861    620.603   SK16
   8 SP     -0.05988024      7.98717872    1.000000      0.000
   9 SP      0.00000000      0.00000000    1.000000      0.000
<END>
```

Similarly, the command CHAREFLECSUR can be used to change a mirror surface. Refer to section for syntax and instructions.

5.3 Changing field points

After changing, adding, or removing a field point, vignetting factors (pupil shift and pupil expansion) can be recalculated using the command $VIFA.

6 Scripts

The syntax used in script mode is the same as the syntax in an interactive session. The exit command should be the last line in the script. If it is not, Eikonal+ will hang, because it is not reading from stdin during script execution.

Linux:

```
cat MYSCRIPT | /usr/local/bin/eikonalplus > OUTPUT.TXT
```

Windows using PowerShell:

```
cat MYSCRIPT | C:/Program\ Files/Eikonal/eikonalplus.exe > OUTPUT.TXT
```

Redirecting output is optional.

7 Glass Data

7.1 The Glass Codes

Each glass in the Eikonal+ catalog has a 6-digit number ‘‘nnnaaa’’ or ‘‘nnn.aaa’’ that uniquely identifies it (unless two manufacturers make glasses that are very similar. The original code does not handle this in an unambiguous way.

The code is constructed from the refractive index and the Abbe number as follows.

The first 3 digits of the code, ‘‘nnn,’’ come from the first 3 significant digits of the refractive index *after the decimal point*, e.g. if the refractive index is 1.2468, then ‘‘nnn’’ is ‘‘247’’.

The last 3 digits, ‘‘aaa,’’ are the first 3 significant digits of the Abbe number v_d, e.g. if $v_d = 25.76$, then ‘‘aaa’’ = ‘‘258’’.

For the glass SF11 with refractive index $n_d = 1.7847$ and Abbe number $v_d = 25.76$, the glass code would be ‘‘785.258’’ or ‘‘785258’’.

7.2 Searching the Glass Catalog

8 Freeform Surfaces

8.1 Surfaces calculated by Eikonal+

Eikonal+ can can define freeform surfaces using fringe Zernike polynomials up to the 21st term. (list below)

9 Appendix

Term	Fringe Zernike polynomial	Aberration type
1	1	Piston (constant)
2	$Rcos\theta$	Distortion - Tilt (x-axis)
3	$Rsin\theta$	Distortion - Tilt (y-axis)
4	$2R^2 - 1$	Defocus - Field curvature
5	$R^2cos(2\theta)$	Astigmatism, Primary (axis
6	$R^2sin(2\theta)$	Astigmatism, Primary (axis
7	$(3R^3 - 2R)cos\theta$	Coma, Primary (x-axis)
8	$(3R^3 - 2R)sin\theta$	Coma, Primary (y-axis)
9	$6R^4 - 6R^2 + 1$	Spherical Aberration, Prima
10	$R^3cos(3\theta)$	Trefoil, Primary (x-axis)
11	$R^3sin(3\theta)$	Trefoil, Primary (y-axis)
12	$(4R^4 - 3R^2)cos(2\theta)$	Astigmatism, Secondary (ax
13	$(4R^4 - 3R^2)sin(2\theta)$	Astigmatism, Secondary (ax
14	$(10R^5 - 12R^3 + 3R)cos\theta$	Coma, Secondary (x-axis)
15	$(10R^5 - 12R^3 + 3R)sin\theta$	Coma, Secondary (y-axis)
16	$20R^6 - 30R^4 + 12R^2 - 1$	Spherical Aberration, Secon
17	$R^4cos(4\theta)$	Tetrafoil, Primary (x-axis)
18	$R^4sin(4\theta)$	Tetrafoil, Primary (y-axis)
19	$(5R^5 - 4R^3)cos(3\theta)$	Trefoil, Secondary (x-axis)
20	$(5R^5 - 4R^3)sin(3\theta)$	Trefoil, Secondary (y-axis)
21	$(15R^6 - 20R^4 + 6R^2)cos(2\theta)$	Astigmatism, Tertiary (axis
22	$(15R^6 - 20R^4 + 6R^2)sin(2\theta)$	Astigmatism, Tertiary (axis
23	$(35R^7 - 60R^5 + 30R^3 - 4R)cos\theta$	Coma, Tertiary (x-axis)
24	$(35R^7 - 60R^5 + 30R^3 - 4R)sin\theta$	Coma, Tertiary (y-axis)
25	$70R^8 - 140R^6 + 90R^4 - 20R^2 + 1$	Spherical Aberration, Tertia
26	$R^5cos(5\theta)$	Pentafoil, Primary (x-axis)
27	$R^5sin(5\theta)$	Pentafoil, Primary (y-axis)
28	$(6R^6 - 5R^4)cos(4\theta)$	Tetrafoil, Secondary (x-axis)
29	$(6R^6 - 5R^4)sin(4\theta)$	Tetrafoil, Secondary (y-axis)
30	$(21R^7 - 30R^5 + 10R^3)cos(3\theta)$	Trefoil, Tertiary (x-axis)
31	$(21R^7 - 30R^5 + 10R^3)sin(3\theta)$	Trefoil, Tertiary (y-axis)
32	$(56R^8 - 105R^6 + 60R^4 - 10R^2)cos(2\theta)$	Astigmatism, Quaternary (a
33	$(56R^8 - 105R^6 + 60R^4 - 10R^2)sin(2\theta)$	Astigmatism, Quaternary (a
34	$(126R^9 - 280R^7 + 210R^5 - 60R^3 + 5R)cos\theta$	Coma, Quaternary (x-axis)
35	$(126R^9 - 280R^7 + 210R^5 - 60R^3 + 5R)sin\theta$	Coma, Quaternary (y-axis)
36	$252R^{10} - 630R^8 + 560R^6 - 210R^4 + 30R^2 - 1$	Spherical Aberration, Quate
37	$924R^{12} - 2772R^{10} + 3150R^8 - 1680R^6 + 420R^4 - 42R^2 + 1$	Spherical Aberration, Quint

Table 1: Fringe Zernike polynomials as defined for surfaces in Eikonal+.
Taken from the *CODE V Lens System Setup Reference Manual*.

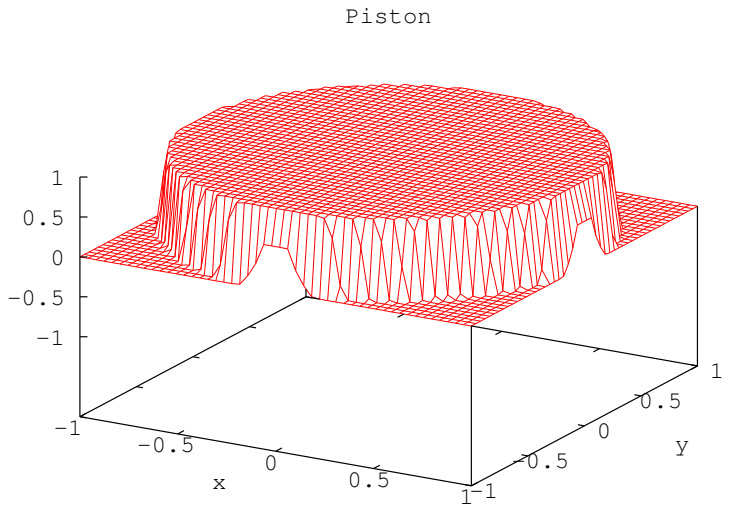

Figure 4: Piston

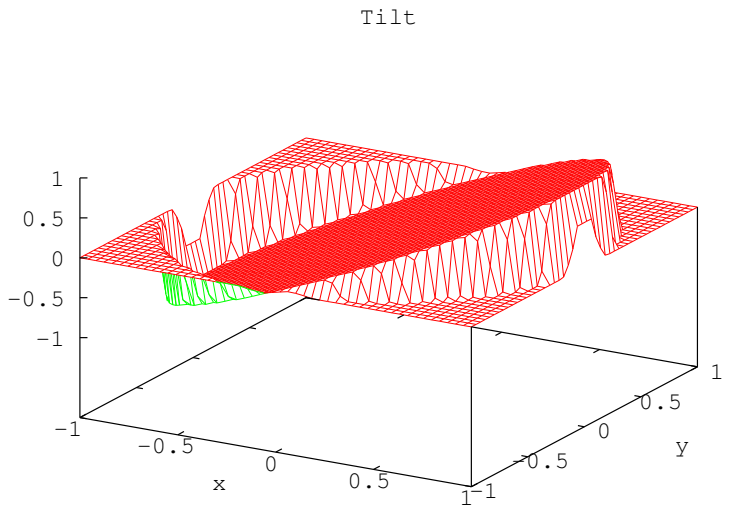

Figure 5: Tilt

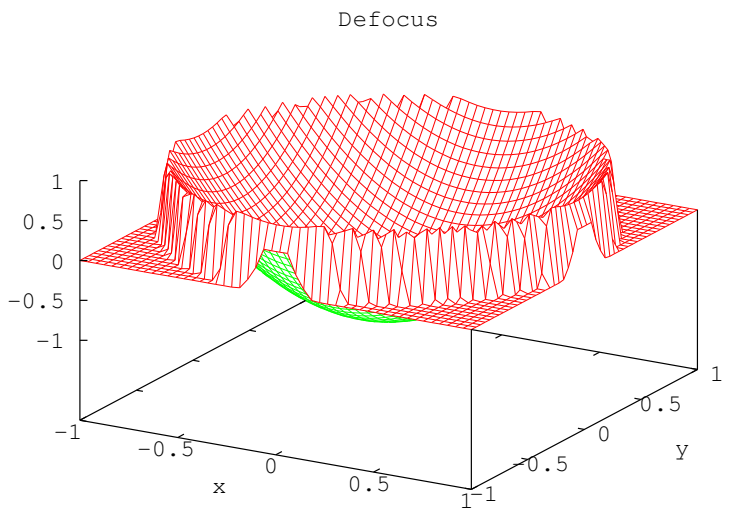

Figure 6: Defocus (field curvature)

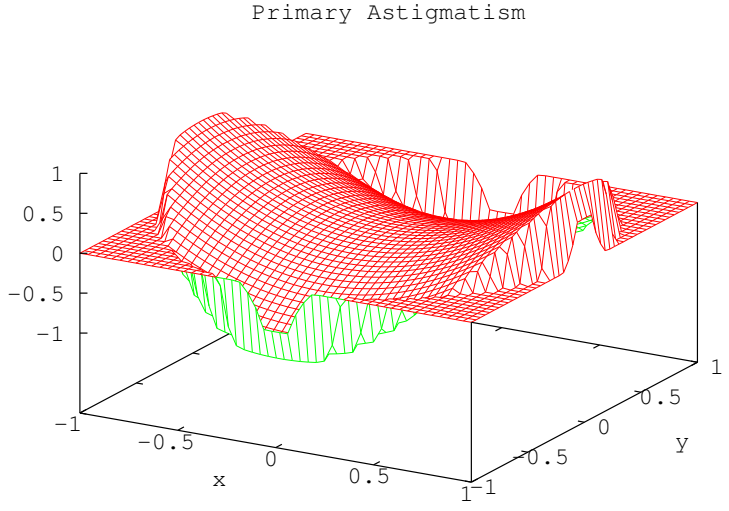

Figure 7: Primary astigmatism

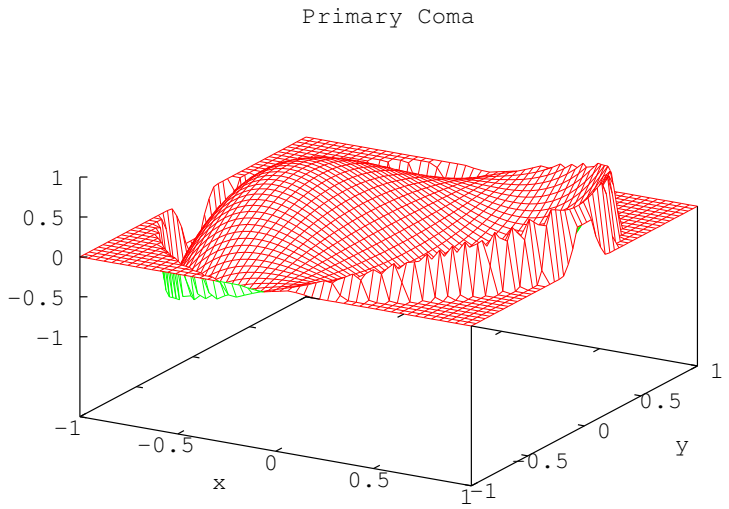

Figure 8: Primary coma

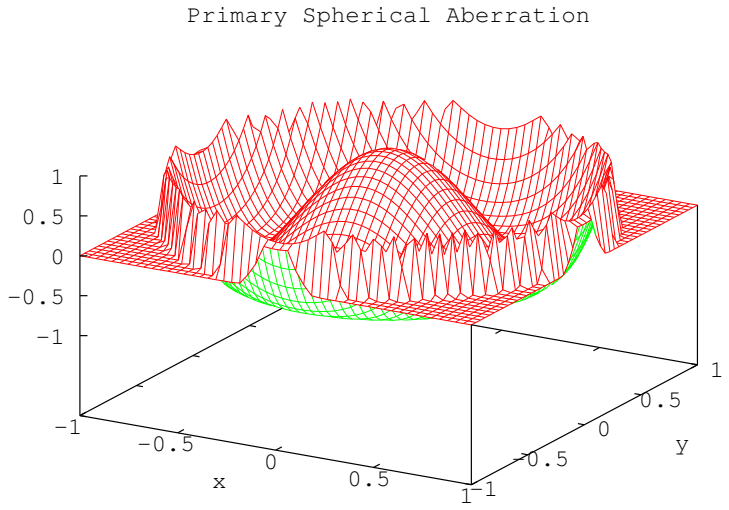

Figure 9: primary-spherical-aberration

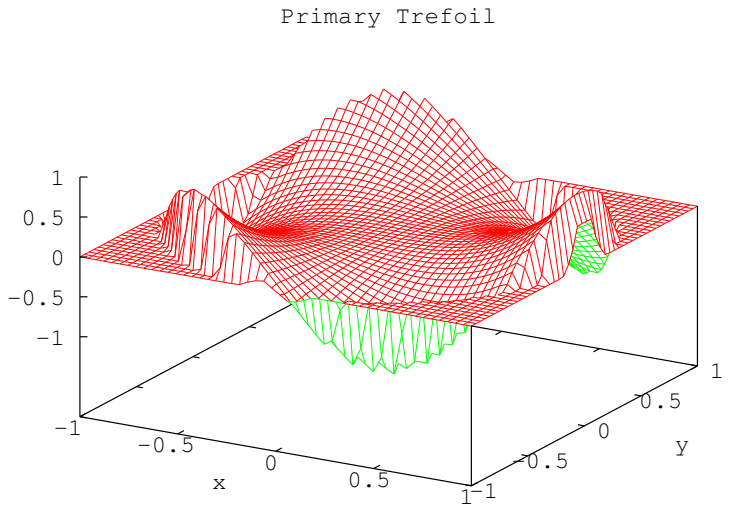

Figure 10: primary-trefoil

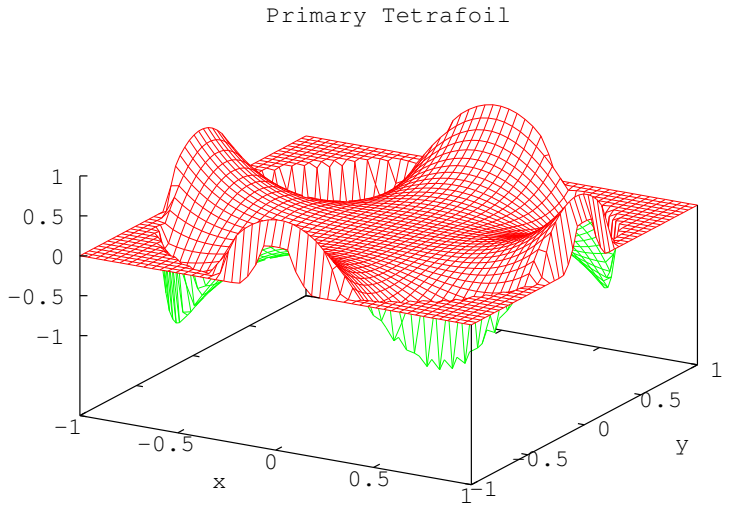

Figure 11: primary-tetrafoil

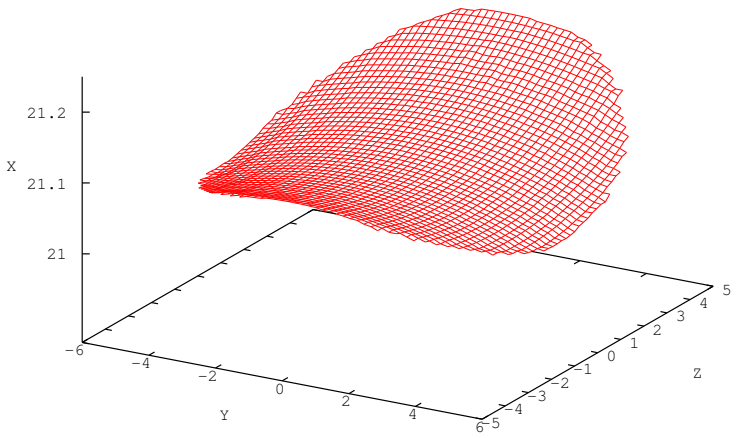

Figure 12: A freeform surface including the Zernike tilt, defocus and astigmatism terms. This surface, a set of ray-surface intersection points, was produced by Eikonal+ during ray tracing.

Index