

**NAME**

surface\_intercept - evaluate surface intercept

**VERSION**

surface\_intercept Version D19990113

**PARAMETERS**

**surface\_intercept** uses an IRAF-compatible parameter interface. A template parameter file is in */proj/axaf/simul/lib/uparm/surface\_intercept.par*.

**surf\_no** *surface number*

This parameter specifies the surface number for the optic (normally 1 for the paraboloid of a Wolter type-I optic). This is used to specify which surface to read from the "gi" file.

**input** *input file*

This parameter specifies the name of the file/stream for the input `bpipe`. If the filename is the string `stdin`, it reads UNIX standard input.

**output** *output file*

This parameter specifies the name of the file/stream for the output `bpipe`. If the filename is the string `stdin`, it write to UNIX standard output.

**logfile** *log file*

This parameter specifies the name of the file to contain the raytrace summary and error messages. If no `logfile` parameter is provided, a warning message is issued and the default value (`surfn.lis`, where *n* is the number of the current surface) will be used.

**gi\_filename** *OSAC-style 'gi' file*

This parameter specifies the name of the OSAC-style 'gi' file containing the optic prescription. The `surf_no`'th surface is used.

**dfm\_type** *deformation type*

This parameter specifies the type of deformation to be applied:

- 0      Fourier-Legendre only. `dfm_filename` specifies a `cogen`-generated Fourier-Legendre deformation file.
- 1      Spline only. `dfm_filename` specifies a file containing `transfit`-generated spline deformation coefficients.
- 2      Combination of `transfit`-generated spline and `cogen`-generated Fourier-Legendre deformations. `dfm_filename` specifies the spline data and `dfm2_filename` specifies the Fourier-Legendre data.

**dfm\_filename** *deformation coefficients file*

This parameter specifies the name of a file containing the deformation coefficients. The file contains either Fourier-Legendre 'cogen' coefficients, or spline coefficients.

By convention, the following file suffixes are used:

.DFR

ascii file containing `cogen`-generated Fourier-Legendre 'cogen' coefficients.

.SPL

ascii file containing `transfit`-generated spline coefficients.

`.BSPL`

binary file containing `transfit`-generated spline coefficients.

`.spl`

binary file containing `transfit`-generated spline coefficients. New format incorporating precomputed rms amplitude value.

**dfm2\_filename** *deformation coefficients file #2*

This parameter specifies the name of a file containing the second set of deformation coefficients to be used in the case that `dfm_type` = 2. Currently the file must contain `cogen`-generated Fourier-Legendre coefficients.

**dfm\_scale** *deformation scale parameter*

This parameter scales the deformation (and the theta and z derivatives) by a constant. Normally it should be set to 1.0 (no scaling)

**dfm2\_scale** *deformation scale parameter #2*

This parameter scales the second set of deformations (and their theta and z derivatives) by a constant. Normally it should be set to 1.0 (no scaling)

**theta0** *clocking angle*

This parameter specifies the reference clocking angle for the optic. `theta0` is in degrees; a positive clocking rotates the optic deformation such that a fiducial mark on the +X axis (SAO raytrace coordinates) moves towards the +Y axis.

The SAO raytrace coordinates are:

Z-axis

parallel to the nominal optical axis. Z increases from the mirror towards the focal plane.

Y-axis

"up"

X-axis

completes a right-handed coordinate system.

**theta02** *clocking angle #2*

This parameter specifies the reference clocking angle for the 2nd set of deformation parameters in the case that `dfm_type` = 2. `theta02` is in degrees; a positive clocking rotates the optic deformation such that a fiducial mark on the +X axis (SAO raytrace coordinates) moves towards the +Y axis.

**do\_osac\_reflect** *yes|no*

Boolean flag indicating the handling of reflection from the surface.

yes

`surface_intercept` locates the ray-surface intercept. If the ray intercepts within the bounds of the current optic, the ray is moved to the surface intercept location on the optic and reflected: The ray direction vector is reflected and the outgoing ray polarization state is reflected using the Fresnel equations and the complex dielectric constant which was read in from the ".gi" file.

NOTE: this setting is incompatible with the `reflect` module.

`no`

`surface_intercept` locates the ray-surface intercept but does **not** reflect the ray. The ray position The ray will be reflected by another ray-reflection module.

NOTE: this setting is required when using the `reflect` module.

If the ray intercepts ahead of the surface (outside the forward boundary), the ray is considered to have a left stop error and is discarded. If the ray intercepts aft of the aft boundary the treatment of the ray depends on the setting of the `onlygoodrays` flag.

**onlygoodrays** `yes|no`

Boolean flag indicating the handle of "ghost" rays, *i.e.*, rays which have not reflected from one or more of the optics encountered so far.

`yes`

Only those rays which have reflected off each optic (so far) will be passed along the pipeline.

`no`

Ghost rays will also be passed. Ghost rays include nonreflected rays and rays which miss at least one optical surface. The position and direction of the ghost ray correspond to the ray position/direction just after the last successful reflection.

Rays which are marked bad by `surface_intercept` for other reasons (e.g., left stop errors, negative pathlength errors, ...) are not considered to be ghost rays and are not passed.

**help** `yes|no`

Print out a simple help message and exit.

**version:** `yes|no`

Print out `surface_intercept`'s version and exit.

**debug** `list`

A list of debug flags. The presently supported flags are:

`save-history`

increase the extent of all `BPipe` data fields by one; this allows a history of data values to be accumulated.

This option was previously called `extend_all`; this is deprecated, but still supported.

## DESCRIPTION

**surface\_intercept** reads `BPipe` format rays finds the intercept point of the ray with a deformed surface.

If the parameter `do_osac_reflect` = `yes` and the ray hit within the bounds of the surface, **surface\_intercept** will also reflect the ray.

If the ray hits with `Z` coordinate less than the specified `Z` for the edge at smaller `Z`, the ray is absorbed ("left-stop error").

If the parameter `onlygoodrays` = `yes` and the ray intercepts with `Z` greater than the `Z` for the edge at larger `Z` (*i.e.*, the ray missed the surface), the ray is considered invalid and removed from the stream.

If the parameter `onlygoodrays` = `no` and the ray intercepts with `Z` greater than the `Z` for the edge at

larger Z (i.e., the ray missed the surface), the appropriate ghost ray flag is set in the bpipe ray.

## History

The `surface_intercept` module is currently based on the `SAOdrat` module. We expect to migrate towards a pure C/C++ implementation; with time, the dependence on the `OSAC` libraries is expected diminish.

The `SAOdrat` module is built atop the `OSAC` (Optical Surface Analysis Code) library. `OSAC` was developed for Goddard Space Flight Center by the Perkin-Elmer corporation under NASA contract NAS5-25802; four revisions were developed and programmed for Goddard Space Flight Center by Paul Glenn of Bauer Associates. The current version level for `OSAC` is Version 7.0. A fuller description of the `OSAC` suite of programs is provided in the *Optical Surface Analysis Code (OSAC) Version 7.0 User's Manual, 2nd edition, August 11, 1993*, available from the Goddard Space Flight Center.

The `SAOsac` version of `OSAC` library is based on Version 5.0 of the `OSAC` suite, revised and updated to incorporate the modifications and bug fixes from Versions 6 and 7 of the `OSAC` suite.

The `SAOdrat` module is modeled in part on the `drat` program of `OSAC`. `drat` is a stand-alone program which performs a raytrace of a (possibly deformed) optic. In `OSAC`, the trace is initialized by `geosac`, and an invocation of `drat` is run for each surface to be traced. The first invocation of `drat` generates the rays on a "ring and spoke" pattern within an annulus located at plane  $Z = 0$  in the `OSAC` standard coordinate system (STD). A final invocation of `drat` locates the focus and moves the ray to a specified plane. The separate invocations of `drat` communicate via `ascii` ray data files.

Because of the limitations `OSAC`, a modified version of `OSAC`, dubbed `frSAOsac`, was developed by members the Smithsonian Astrophysical Observatory AXAF Mission Support Team (SAO/MST) Simulations and Analysis group together with members SAO Central Engineering. The chief problems to be addressed by the initial version of `frSAOsac` were:

- 1 the maximum number of rays was limited to approximately 100000; this is too small for many simulations.
- 2 the intermediate `ascii` rayfiles can be enormous; the disk i/o associated with reading and writing the files imposes too much of a performance hit.
- 3 the deformations handled by `OSAC` (specified by Fourier-Legendre coefficients) are not capable of handling the deformations produced by the mechanical modeling of the AXAF optics.
- 4 the monolithic nature of `drat` made it hard to modify.

## Design

It was decided to set up the raytrace system based on the UNIX filter paradigm: the `drat` functionality was handled by a set of programs communicating via UNIX pipes. A program reads a ray from its standard input stream, manipulates it and writes it to its standard output stream. The use of pipes avoids the necessity of reading and writing large `ascii` data files.

The programs are completely independent from each other, making program development as well as validation and verification much easier. Programs can be added or removed from the pipeline, greatly adding to the flexibility.

The `OSAC` deformed surface raytrace capability was factored into a set of programs

`frRaygen`

Generate rays on an input ray bundle annulus. The source can be a finite distance or at infinite ( $1 \times 10^{30}$ ) distance. Rays can be generated on a (ring and spoke) grid pattern of up to 500 rings and 500 spokes. Rays can also be generated with a random distribution within the input ray

bundle annulus.

`frSAOdrat`

Trace rays through a (possibly deformed) optic. The `frSAOdrat` module was based based on `drat`, with modifications to communicate via binary streams read from standard input and written to standard output.

`frSAOfocus`

Determine the Gaussian focus; locate the best focus location and transport rays to the focal plane.

The modules were converted to use the `BPipe` interface and as of version 1.0, `frSAOdrat` was renamed to `SAOdrat` and `frSAOfocus` was renamed to `SAOfocus`.

## OTHER

This version of `SAOdrat` makes extensive use of routines from the NASA Goddard Space Flight Center code `OSAC`, "Optical Surface Analysis Code".

This version of `SAOdrat` also makes use of spline evaluation routines from the Numerical Algorithms Group (NAG) library Data Approximation Subroutine Library (`DASL`).

## COPYRIGHT & LICENSE

Portions of this software are copyright 2006 Smithsonian Astrophysical Observatory and released under the GNU General Public License. You may find a copy at <http://www.fsf.org/copyleft/gpl.html>

However, it's not clear which parts are redistributable, as this is based upon `OSAC` code, which has an indeterminate copyright. Thus, don't redistribute this code!

## AUTHORS

*DTN Dan Nguyen (SAO/Chandra X-ray Center)*

*MDF Mark Freeman (SAO/Central Engineering)*

*TJG Terry Gaetz (SAO/AXAF Mission Support Team)*

*DMG David Grumm (SAO/AXAF Mission Support Team)*