SolTrace

Copyright © 2010 National Renewable Energy Laboratory

Table of contents

Introduction 3
Methodology 4
Basic Use 7
Defining the Sun 8
Geometry 9
Optical Properties 18
Tracing 21
Visualization 24
Exporting Data 26
Scripting 27
SolTrace Functions 30
Scripting Examples 33
References 33

Introduction

SolTrace is a software tool developed at the National Renewable Energy Laboratory (NREL) to model solar power optical systems and analyze their performance. Although originally intended for solar optical applications, the code can also be used to model and characterize general optical systems. The creation of the code evolved out of a need to model more complex optical systems than could be modeled with existing tools such as OPTDSH (Steele et al., 1991) and CIRCE (Ratzel and Boughton, 1987).

The code utilizes a ray-tracing methodology (Spencer and Murty, 1962). The user selects a given number of rays to be traced. Each ray is traced through the system while encountering various optical interactions. Some of these interactions are probabilistic in nature (e.g. selection of sun angle from sun angular intensity distribution) while others are deterministic (e.g. calculation of ray intersection with an analytically described surface and resultant redirection.) Such a code has the advantage over codes based on convolution of moments in that it replicates real photon interactions and therefore can provide accurate results for complex systems that cannot be modeled otherwise. The disadvantage is longer processing time. Accuracy increases with the number of rays traced and larger ray numbers means more processing time. Also, complex geometries translate into longer run times. However, the required number of rays is also a function of the desired result. For example, fewer rays (and therefore less time) are needed to determine relative changes in optical efficiency for different sun angles on a given solar concentrator than say are needed to accurately assess the flux distribution on the receiver of that same concentrator. Thus the responsibility is on the user to use the code wisely and efficiently.

The application is written in C++ and uses the cross-platform wxWidgets toolkit, allowing it to run on Windows, Linux/Unix, and Mac OS X operating systems. The core tracing procedures are separated from the graphical user interface, and allow for the incorporation of the calculation code into other systems and applications. SolTrace scales exceptionally well to run on computers with multiple processors, as each individual ray can be computed independently of the others. Computers with 8 processors will generally experience close to an 8x speedup over a single processor, but this trend can be limited by various factors including the specific nature of a particular geometry or other computer hardware limitations (memory, etc).

Created with the Personal Edition of HelpNDoc: From Word to ePub or Kindle eBook: A Comprehensive Guide

Methodology

In SolTrace, an optical system is organized into “stages” within a global coordinate system. A stage is loosely defined as a section of the optical geometry which, once a ray exits the stage, will not be re-entered by the ray on the remainder of it’s path through the system. A complete system geometry may consist of one or more stages. It is incumbent on the user to define the stage geometry accordingly. The motivation behind the stage concept is to employ efficient tracing and therefore save processing time and allow for a modular representation of a system. The other significant benefit of stages is that they can also be saved and employed in other system geometries without the need for recalculating element positions and orientations.

A stage is comprised of “elements”. Each element consists of a surface, an optical interaction type, an aperture shape and, if appropriate, a set of optical properties. The location and orientation of stages are defined within the global coordinate system whereas the location and orientation of elements are specified within the coordinate system of the particular stage in which they are defined. Stages can be one of two types: optical or virtual. An optical stage is defined as one that physically interacts with the rays. Conversely, a virtual stage is defined as one that does not physically interact with the rays. The virtual stage is useful for determining ray locations and directions at various positions along the optical path without physically affecting ray trajectory. Elements defined within a virtual stage therefore have no optical properties because they do not interact with the rays. Optical stages consist of elements which interact with the rays potentially altering their trajectories. These elements have optical properties and interaction types associated with them. Beyond this, optical and virtual stages are identical in how they are defined and used. Stages can be duplicated and moved around as groups of elements and then saved for use in other system geometries.

SolTrace uses three right-handed coordinate systems: the global coordinate system, the stage coordinate system and the element coordinate system. These are illustrated in Figure 1a. Each element in a stage has a local coordinate system (i.e. location and

orientation) defined relative to the stage coordinate system. Each stage has a coordinate system defined relative to the global coordinate system. As shall be described later, the direction of the sun is defined relative to the global coordinate system. Currently, the sun direction is input in either vector form, or in time-of-day, day-of-year format with latitude specified. Light rays are generated from the sun and then traced sequentially through each stage in the geometry. The position and direction of each ray in each stage is stored in memory for later processing and output.

The NREL High Flux Solar Furnace is shown as an example of a multi-stage, multi-element system in Figure 1b. Note the global (black), stage (red) and element (blue) coordinate systems. In this example, there are a total of three stages. The first stage is comprised of one flat rectangular reflective element (the heliostat). In this case, the element coordinate system lies directly on top of the stage coordinate system. The second stage is the primary concentrator consisting of twenty-five hexagonal reflective elements each having spherical curvature. One of the twenty-five individual element coordinate systems is shown to the upper right of the stage coordinate system. The third and final stage is the sample stage (a rectangular flat target) in the HFSF control room. Again, the element coordinate system occupies the same location and orientation as the stage coordinate system in this case. The details of element/stage definition are presented later.

Figures 1a-b. SolTrace Coordinate Systems

The stage and element coordinate systems are translated from the global coordinate system and stage coordinate system origins respectively and then oriented via three Euler angle rotations. These rotations are shown in Figure 2.

Figure 2. Generation of (x, y, z) system from the (/x, /y, /z) system after translation of origin.

The first rotation is by angle α about the y-axis, the second rotation by angle β about the x-axis and the last rotation by angle γ about the z-axis. After translation of the child coordinate system origin from the parent coordinate system, these three rotations completely specify the orientation of the child coordinate system within the parent. The first two rotations are automatically determined by specifying a point in the parent coordinate system toward which the z-axis of the child coordinate system is aligned. The last rotation of the child coordinate system about the z-axis is then specified. The rotation matrix that transforms coordinates from reference coordinates X_{ref} to local coordinates X_{loc} is given below.

$$X_{loc} = \begin{bmatrix} \cos\alpha\cos\gamma + \sin\alpha\sin\beta\sin\gamma & -\cos\beta\sin\gamma & -\sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma \\ \cos\alpha\sin\gamma - \sin\alpha\sin\beta\cos\gamma & \cos\beta\cos\gamma & -\sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma \\ \sin\alpha\cos\beta & \sin\beta & \cos\alpha\cos\beta \end{bmatrix} X_{ref}$$

Created with the Personal Edition of HelpNDoc: Full-featured EBook editor

Basic Use

Upon opening the SolTrace application, the main window opens with an empty project, as shown below.

Figure. Empty project upon opening SolTrace application

A SolTrace project file contains a complete set of inputs that define a model or analysis. This includes information about the sun shape and position, a set of surface optical interaction properties, and all the elements for each stage in the system. The data is stored in a plain text formatted file with the "*.stinput" extension. The results of a ray trace are not stored in this file, nor are externally defined surface types such those described by VSHOT data (discussed later). As a result, the .stinput files are generally quite small.

The icon toolbar across the top of the main window allow the user to move between various data input and output pages, shown below.

Figure. SolTrace Navigation Toolbar

The first three pages (Sun Shape, Optics, Geometry) are used to define the system. The Perform Ray Trace page is the place to configure the number of rays desired, the number of computer processors to use, and other simulation related parameters.

After a trace has completed successfully, the Visualize and Export pages are populated with the calculated ray intersection data. Several graphing options are included, as well

as the ability to save the raw intersection data in comma-separated value (CSV) format, or to export directly to Microsoft Excel on Windows computers.

Scripting is available using the dropdown menu, which allows a user to programmatically control SolTrace to configure the geometry of complex systems with hundreds of elements (i.e. a heliostat field for a power tower), or to post-process the calculated intersection points. Some example scripts are provided.

Figure. SolTrace dropdown menu.

Created with the Personal Edition of HelpNDoc: Qt Help documentation made easy

Defining the Sun

Two characteristics completely define the “sun” as the light source: the angular intensity distribution of light across the sun’s disk (referred to as the sunshape) and the sun’s position. The area in the middle-left of the window shown in Figure 5 is the first step in defining the sunshape. Three options are available. The first two are commonly used probabilistic distributions. Although the sunshape varies widely with terrestrial location, sky conditions and time and is neither truly Gaussian nor pillbox in nature, this approximation is adequate for a large class of problems. The parameter defining a Gaussian distribution for the sun’s disk is the half-angle σ. The parameter for the pillbox, being a flat distribution, is simply the half-angle width.

The third option allows the use to define the sunshape profile as a series of intensity datum points (Neumann et al., 2002). Since the sunshape is axisymmetric, only half of the profile (from the sun’s center to the edge) is required. This can be manually entered into the table seen in the lower left corner of the window. This table becomes available when the user selects "User defined" option. The number of datum points in the half profile is entered in the box to the right. The profile is entered as a set of datum points: the angular position from the sun’s center in milliradians (starting with 0) and the corresponding relative intensity at that position. The intensity values can be in any units because the code scales the distribution to a peak of unity. Once entered into the table, the profile can be stored in a file for future use and later retrieved using the Import and Export buttons. The sun shape plot automatically updates as data is entered into the shape table.

Once the sunshape has been defined, all that remains is to determine the sun direction, which is accomplished using the inputs in the upper left of the figure. One option is to

define a point in the global coordinate system such that a vector from this point to the global coordinate system origin defines the sun direction. The other option is to define a particular site latitude and time (day of year and local hour.) From this information, the sun direction is determined assuming the z-axis of the global coordinate system points due north and the y-axis points towards zenith. This is important to remember when defining the optical geometry.

By default, all sun rays cast come from the same direction, regardless of their position within the bounding box. It is possible to instead model the sun as a point-source (infinitely small source) emitting rays from a specific position within the coordinate system. To enable this option, check the "Point source at finite distance" checkbox and specify the sun's coordinate position.

Figure 5. Defining the Sun Shape and Position

Created with the Personal Edition of HelpNDoc: Streamline your documentation process with HelpNDoc's HTML5 template

Geometry

Defining System Geometry

Once the sun is defined, the user can proceed to the next step – defining the optical geometry to analyze. The system geometry is defined on the "Geometry" page, shown below.

	En.	X-Coord.	Y-Coord.	Z-Coord.	X-AimPt.	Y-AimPt.	Z-AimPt.	Z-Rot.	Aperture	Surface	Interaction	Optics	Comment
1	☑	-1.32144	1.5748	0.337323	1.83577	0.157399	9.71938	0	h-0.8799,0,0,0,0,0,0	s-0.06892,0.06892,0,0,0,0,0	Reflection	primary facet	
2	☑	-1.32144	0.7874	0.337323	1.89885	-0.11253	9.76175	0	h-0.8799,0,0,0,0,0,0	s-0.06889,0.06889,0,0,0,0,0	Reflection	primary facet	
3	☑	-1.32144	0	0.337323	1.93805	-0.355782	9.7845	0	h-0.8799,0,0,0,0,0,0	s-0.06823,0.06823,0,0,0,0,0	Reflection	primary facet	
4	☑	-1.32144	-0.7874	0.337323	1.95002	-0.584572	9.78488	0	h-0.8799,0,0,0,0,0,0	s-0.06811,0.06811,0,0,0,0,0	Reflection	primary facet	
5	☑	-1.32144	-1.5748	0.337323	1.93356	-0.813122	9.76201	0	h-0.8799,0,0,0,0,0,0	s-0.068,0.068,0,0,0,0,0	Reflection	primary facet	
6	☑	-0.660721	1.1811	0.168661	2.12162	-0.0110794	9.69952	0	h-0.8799,0,0,0,0,0,0	s-0.06859,0.06859,0,0,0,0,0	Reflection	primary facet	
7	☑	-0.660721	0.3937	0.168661	2.1694	-0.257396	9.7377	0	h-0.8799,0,0,0,0,0,0	s-0.06811,0.06811,0,0,0,0,0	Reflection	primary facet	
8	☑	-0.660721	-0.3937	0.168661	2.19352	-0.481112	9.75228	0	h-0.8799,0,0,0,0,0,0	s-0.06835,0.06835,0,0,0,0,0	Reflection	primary facet	
9	☑	-0.660721	-1.1811	0.168661	2.19165	-0.696651	9.74098	0	h-0.8799,0,0,0,0,0,0	s-0.06806,0.06806,0,0,0,0,0	Reflection	primary facet	
10	☑	-0.660721	-1.9685	0.168661	2.1638	-0.919691	9.70397	0	h-0.8799,0,0,0,0,0,0	s-0.06833,0.06833,0,0,0,0,0	Reflection	primary facet	
11	☑	0	1.5748	0	2.31639	0.100116	9.61559	0	h-0.8799,0,0,0,0,0,0	s-0.06859,0.06859,0,0,0,0,0	Reflection	primary facet	
12	☑	0	0.7874	0	2.36754	-0.156593	9.66973	0	h-0.8799,0,0,0,0,0,0	s-0.06823,0.06823,0,0,0,0,0	Reflection	primary facet	
13	☑	0	0	0	0.239967	-0.0382958	0.970025	0	h-0.8799,0,0,0,0,0,0	s-0.06811,0.06811,0,0,0,0,0	Reflection	primary facet	
14	☑	0	-0.7874	0	2.40976	-0.592645	9.70336	0	h-0.8799,0,0,0,0,0,0	s-0.06811,0.06811,0,0,0,0,0	Reflection	primary facet	
15	☑	0	-1.5748	0	2.39667	-0.801881	9.67774	0	h-0.8799,0,0,0,0,0,0	s-0.06794,0.06794,0,0,0,0,0	Reflection	primary facet	
16	☑	0.660721	1.1811	-0.168661	2.54302	-0.0450383	9.57575	0	h-0.8799,0,0,0,0,0,0	s-0.06823,0.06823,0,0,0,0,0	Reflection	primary facet	
17	☑	0.660721	0.3937	-0.168661	2.57783	-0.28196	9.62257	0	h-0.8799,0,0,0,0,0,0	s-0.06803,0.06803,0,0,0,0,0	Reflection	primary facet	
18	☑	0.660721	-0.3937	-0.168661	2.59558	-0.494036	9.64186	0	h-0.8799,0,0,0,0,0,0	s-0.06823,0.06823,0,0,0,0,0	Reflection	primary facet	
19	☑	0.660721	-1.1811	-0.168661	2.59443	-0.696985	9.63065	0	h-0.8799,0,0,0,0,0,0	s-0.06779,0.06779,0,0,0,0,0	Reflection	primary facet	
20	☑	0.660721	-1.9685	-0.168661	2.57435	-0.908035	9.58908	0	h-0.8799,0,0,0,0,0,0	s-0.06833,0.06833,0,0,0,0,0	Reflection	primary facet	
21	☑	1.32144	1.5748	-0.337323	2.70897	0.0850036	9.45325	0	h-0.8799,0,0,0,0,0,0	s-0.06811,0.06811,0,0,0,0,0	Reflection	primary facet	
22	☑	1.32144	0.7874	-0.337323	2.74079	-0.169573	9.51507	0	h-0.8799,0,0,0,0,0,0	s-0.06811,0.06811,0,0,0,0,0	Reflection	primary facet	
23	☑	1.32144	0	-0.337323	2.76087	-0.392383	9.55075	0	h-0.8799,0,0,0,0,0,0	s-0.068,0.068,0,0,0,0,0	Reflection	primary facet	

Figure 6. Optical geometry definition input page.

The input page is shown with data already entered for the sake of example. The form can now be used to design every aspect of the optical problem to be analyzed. First, stages can be added and removed using the buttons at the top of the page. Using the Import and Export buttons, specific stages can be saved and recalled later as needed. Each stage is displayed as a tab, much like worksheets in a spreadsheet. For each stage, there is a set of stage properties, position and orientation information in the global coordinate system, as well as a table of optical elements.

Calculation Procedure for Multiple Stages

Three stages have been defined in this example which is actually that of the NREL High Flux Solar Furnace shown in Figure 1b. The first two stages are of the optical type (i.e. contain actual optical elements that interact with light). Stage 1 consists of the primary heliostat which redirects sunlight to the concentrator (Stage 2.) The last stage is a virtual stage consisting simply of a flat target for the resultant flux distribution generated at the focus of the concentrator in Stage 2. Light rays will ultimately be traced from the sun to Stage 1 then to Stage 2 and finally to Stage 3. This assumes that once rays leave a stage they cannot physically re-enter that stage again by any combination of the stages following (a reiteration of the stage definition.) The user must carefully design the optical system such that the stages meet this criteria otherwise incorrect results will be obtained.

Note the checkbox labeled 'Trace Through' which appears (already checked by default) at the top of each optical stage worksheet (with the exception of Stage 1 if it is an optical stage.) If this box is checked, it means that rays continue to be traced through the rest of the system even if they miss all the elements of this particular stage. If not checked, then rays which miss this stage completely are no longer traced through the rest of the system. More information on the tracing procedure is available in the 'Tracing' section.

Stage Position and Orientation

The first set of three inputs is the location (in x,y,z) of the stage coordinate system origin within the global coordinate system. The set of four numbers below this determines the orientation of the stage coordinate system within the global coordinate system. The first three numbers of this set define a special point within the global coordinate system. A vector from the stage origin to this point defines the z-axis of the stage coordinate system. The last degree of freedom to be defined is then the rotation of the stage coordinate system about this z-axis. This is entered as the fourth number in degrees (Z Rotation). Every stage must have these parameters defined within the global coordinate system.

Element Definition

Elements can be added and removed from the stage by using the appropriate Insert, Append, and Delete buttons located above the main element data table. Each element is defined on one row of the table, is identified by its row number and by the stage in which it exists.

It is possible that the user may want to “turn on” or “turn off” certain elements within a stage while doing different traces for comparison purposes. Rather than deleting all the data for a particular element and then having to re-enter it again later, the "En." checkbox can be used to enable and disable elements. If checked (the default), the element is included as part of the geometry, if unchecked the element is ignored.

The next three columns are the x,y,z coordinates of the element coordinate system origin within the stage coordinate system. The next three columns define the x,y,z coordinates of a special point within the stage called the aim point. A vector from the element origin to this aim point defines the z-axis of the element coordinate system. The ninth column is the rotation of the element coordinate system about it’s z-axis in degrees. This set of seven numbers describing the location and orientation of the element within the stage is exactly analogous to the set described above for the stage within the global coordinate system.

Apertures

The aperture column contains the description of the projected shape of the element opening in a plane perpendicular to the element z-axis. An example of a circular aperture is shown in Figure 8.

Figure 8. Aperture description

A variety of aperture shapes is available. The column entry is a text string that encodes the aperture description. The code format for the different apertures is shown in Table 1. The code can be entered manually as a string of text. A code begins with a lower case letter denoting the type of aperture shape, followed by a hyphen, and then a list of eight numbers separated by commas. Some aperture shapes require all eight parameters, while others may only require one. Shape parameters that are unused can be set to 0 and are ignored. The program prevents the user from entering an incorrect string code. An option to manually entering the code is to click on the "Aperture" button above the table. When clicked, a dialog pops up to help a user set the aperture for the currently selected element in the table, shown below.

Aperture Type	***Code***
Circular	c-# (# = diameter of circular aperture)
Hexagonal	h-# (# = diameter of the circle which circumscribes a hexagonal aperture)
Triangular	t-# (# = diameter of the circle which circumscribes an equilateral triangle)
Rectangular	r-$\#_1$,$\#_2$ ($\#_1$,$\#_2$ = width, height of rectangle)
Annular	a-$\#_1$,$\#_2$,$\#_3$ ($\#_1$,$\#_2$,$\#_3$ = inner radius, outer radius, included angle in degrees; $\#_1 < \#_2$, $0 < \#_3 \leq 360$)
Single Axis Curvature Section	l-$\#_1$,$\#_2$,$\#_3$ ($\#_1$,$\#_2$,$\#_3$ = distance to inner edge in x dir, distance to outer edge in x dir, length of section in y dir; $\#_1 < \#_2$)
Irregular Triangle	i-$\#_1$,$\#_2$,$\#_3$,$\#_4$,$\#_5$,$\#_6$ ($\#_1$,$\#_2$ = X,Y coordinates of 1st point; $\#_3$,$\#_4$ = X,Y coordinates of 2nd point; $\#_5$,$\#_6$ = X,Y coordinates of 3rd point)
Irregular Quadrilateral	q-$\#_1$,$\#_2$,$\#_3$,$\#_4$,$\#_5$,$\#_6$,$\#_7$,$\#_8$ ($\#_1$,$\#_2$ = X,Y coordinates of 1st point; $\#_3$,$\#_4$ = X,Y coordinates of 2nd point; $\#_5$,$\#_6$ = X,Y coordinates of 3rd point; $\#_7$,$\#_8$ = X,Y coordinates of 4th point)

Table 1. Some aperture types and corresponding codes

Figure 9. Aperture definition window

The drop-down list at the top of this sub-window contains a list of all the available aperture options. Selecting one displays a graphical description of the aperture and entry boxes for entering the relevant parameters for that aperture. Clicking on "OK" automatically enters the correct string code for the selected aperture into the aperture type column.

Surfaces

Each element's surface is defined by a text code in the "Surface" column of the table, similar to how the aperture is defined. A variety of surface options are available. Table 2 lists the surface options and respective text codes.

Surface Type	***Code***
Parabolic	p-$\#_1$,$\#_2$ ($\#_1$,$\#_2$ = 1/radii of curvature in x , y directions of a parabolic surface)
Spherical	s-$\#_1$,$\#_2$ ($\#_1$,$\#_2$ = 1/radii of curvature in x , y directions of a spherical surface)
Other (hyperboloid, ellipsoids)	o-$\#_1$,$\#_2$,$\#_3$ ($\#_1$,$\#_2$ = 1/radii of curvature in x, y directions, $\#_3$ = κ parameter for other surfaces)
Flat	f
Conical	c-$\#_1$ ($\#_1$ = half angle of conical surface)
Zernike Series	*.mon (surface described by Zernike

	series equation with coefficients in file "*.mon")
VSHOT Data Set	*.sht (surface described by VSHOT data file "*.sht")
Finite Element Data Set	*.fed (surface described by finite element data file "*.fed")
General Spencer & Murty Equation	g-$\#_1$,$\#_2$,$\#_3$,$\#_4$,$\#_5$,$\#_6$,$\#_7$,$\#_8$ ($\#_1$,$\#_2$= 1/radii of curvature in x , y directions, $\#_3$= κ , $\#_{4-8}$ = α_{1-5})
Cylinder	t-$\#_1$ ($\#_1$ = 1 / radius of curvature; use in conjunction with aperture code "l-0,0,$\#_2$" where $\#_2$ is length of cylinder)
Polynomial Series (rotationally symmetric)	*.ply (surface described by coefficients of polynomial equation in file "*.ply")
Cubic Spline Interpolation (rotationally symmetric)	*.csi (surface described by discrete data points and 1st derivative boundary conditions in file "*.csi")

Table 2. Surface types and corresponding codes

The "Zernike Series" surface type is described by

$$Z(X, Y) = \sum_{i=0}^{k} \sum_{j=0}^{i} B_{ij} X^j Y^{i-j}, \quad k = \text{order of monomial}$$

The Zernike series file format is (where N is the order and the B's are the coefficients):

N
$B_{0,0}$
$B_{1,0}$
$B_{1,1}$
$B_{2,0}$
$B_{2,1}$
$B_{2,2}$
.
.
.
$B_{N,N}$

There are a total of (N+1)(N+2)/2 coefficients for the Zernike Series.

The "General Spencer & Murty" formulation is

$$F = Z - c\rho^2/[1 + (1 - \kappa c^2\rho^2)^{1/2}] - \sum_{i=1}^{N} \alpha_i \rho^{2i} = 0,$$ where N need not exceed 5 in most cases (and in fact has been hardwired to this number.)

If terms α_i are omitted from this equation above, it reduces to the "Other" equation representing revolved conic sections. For surfaces described by "Other" or "General

Spencer & Murty", the above general equation reduces to

$$F = Z - c\,(\rho^2 + \kappa Z^2)/2 = 0, \text{ where } c = 1/\text{radius of curvature at vertex}, \ \rho^2 = X^2 + Y^2$$

κ determines the surface type according to the following:

For $\kappa < 0$ surface is a hyperboloid

For $\kappa = 0$, surface is a paraboloid

For $0 < \kappa < 1$, surface is a hemi-ellipsoid of revolution about major axis

For $\kappa = 1$, surface is a hemisphere

For $\kappa > 1$, surface is a hemi-ellipsoid of revolution about minor axis

Surfaces specified by "Spherical" , "Parabolic" or "Conical" are subsets of the more general "Other" surface type designation which in turn is a subset of the even more general "General Spencer & Murty" formulation.

The Polynomial Series option is described by the equation

$$Z(\rho) = \sum_{i=0}^{N} C_i \ \rho^i \quad \text{, where} \quad \rho = (X^2 + Y^2)^{1/2}$$

A surface described by this choice is rotationally symmetric. The polynomial series file format is (where N is the order and C's are the coefficients):

N
C_0
C_1
C_2
.
.
.
C_N

A surface described by Cubic Spline Interpolation of a data point set (ρ_i , Z_i) is also rotationally symmetric. The data required are the data points themselves and the derivatives (or slopes) at the first and last points (boundary conditions). The cubic spline interpolation file format is (where N is the number of data points):

N
ρ_1 Z_1
ρ_2 Z_2
.
.
.
ρ_N Z_N
$dZ/d\rho$ at point 1 $dZ/d\rho$ at point N

A surface described by Finite Element Data requires the specification of nodal positions in a text file. The format of the file expects a first line comment (not used), a second line

indicating the number of nodes contained in the file, and a third line onward specifying the x-y-z coordinates of each node. The coordinates should be separated by a single space. One node is specified per line. The nodes can be part of a structured or unstructured mesh. SolTrace organizes the nodes into a binary search tree to quickly identify local groups of nodes based on a ray hit location, then constructs a local Gauss-Markov ("Kriging") model of the nearest 7 nodes. The surface location and slope is computed using the GM model representation.

```
# Comment / unused
N
X1 Y1 Z1
X2 Y2 Z2
.
.
XN YN ZN
```

Special considerations for specifying the aperture and surface types exist and are listed below:

1) It was originally intended that spherical and parabolic surfaces could have different curvature in the x and y directions. Unfortunately, this was never implemented, and the framework for specifying these was never removed. Thus, for 's' and 'p' surfaces, the 2nd parameter must be equal to 1st parameter or equal to zero (see item 2 below).
2) The only aperture type allowed for single axis curvature surfaces specified by 'p-#,0' or 's-#,0' is the 'l' type aperture. Other aperture types, such as rectangular ('r' type), will not work with single axis curvature surfaces.
3) Except for VSHOT, Zernike monomial and FEA surface descriptions (all of which can be asymmetric), only rotationally symmetric surfaces, planes, trough sections and cylinders can be described by the above analytical formulations.
4) The cylindrical surface (type 't') requires the 'l' aperture to be specified with the first two parameters set equal to 0 and the third being the length of the cylinder: 'l-0,0,length'.
5) The VSHOT file format is not described as this type of file is only available through NREL testing.
6) If tracing rays into a closed surface such as a cylinder it is recommended that a "virtual" window be placed at the end (or entrance) to the cylinder. This can be accomplished with a flat circular element of optic type = 1 (i.e. refractive surface) where the refraction indices for both the front and back side of the element are both = 1 (i.e. zero refraction),the transmissivity = 1.0 (i.e. perfect transmission) and the optical errors are set to very small values (e.g. <0.001 mrad). This surface does not physically interact with the rays (i.e. does not affect their path), but corrects for numerical problems associated with ray origins being far from the entrance to the cylinder. It also provides a convenient surface to flux map the entrance and compare with the exit.

As in the case of the aperture type description, an alternative to manually entering the surface code is to click on the "Surface" button in the "Define" section above the workbook. This button becomes available when the cursor is moved to the surface type column . Doing so brings up the small sub-window shown in Figure 10. The surface type can be selected via the dropdown box at the top of the window. As different types are selected, a graphical description appears complete with the appropriate set of input

parameter boxes on the right side of the window. Entering the correct parameters and clicking on the “OK” button results in the correct code appearing as the column entry.

Figure 10. Surface definition window

Optical Properties

Each element can have a distinct set of optical parameters that define how rays interact with the surface. In the "Interaction" column, the user can select between Refraction and Reflection. Other types of interactions may become available in the future, including Aperture Stop, Diffraction via Transmission, and Diffraction via Reflection. Currently, only the reflective and refractive interaction types have been validated, even though parameters for the other optic types are available on the optical property definition window.

Optical Interaction Type	***Code***
Refraction	1
Reflection	2
Aperture Stop	3
Diffraction via Transmission	4
Diffraction via Reflection	5

Table 3. Optical interaction types and corresponding numeric codes

The optical properties are referenced by name. A SolTrace project may have many sets of optical property defined each with a unique name. These are defined on the "Define Optics" input page, and are discussed in the next section in the manual. Once the

appropriate property sets are defined, the "Optics" column on the element table can be filled in with the name of the optical properties for that element. Clicking the "Optics..." button above the table will pop up a small dialog listing all the optical property sets defined in the system, and by selecting one and clicking "OK", the name will be entered into the appropriate table cell.

For virtual stages, the interaction type and optical properties are ignored and do not need to be initialized.

Created with the Personal Edition of HelpNDoc: Maximize Your Documentation Capabilities with HelpNDoc's Project Analyzer

Optical Properties

Specifying Optical Properties

Optical properties are defined on the second input page, shown below. Each SolTrace project may have numerous optical property sets defined. Use the Add and Remove buttons to add new sets of optical properties. Each optical property set contains a separate set of parameters for the front and back of the surface, accessed by the corresponding tabs. Optical property sets can be saved and retrieved later using the Import and Export buttons.

Note that the reflectivity and transmissivity values are not used together at the same time, but each are only used depending on whether the optical interaction in a Stage Element is specified as "Reflection" or "Refraction," respectively.

Figure. Optical property setup input page.

For refractive optics, only the transmissivity and the real component of the refraction indices are relevant and used at this time. The imaginary refractive index will not be discussed. A real physical refractive component is actually constructed from two elements or surfaces. In Figure 13 for example, a plane of glass consists of two surfaces (or elements) separated by the glass media between. A ray passes from one media (air for example) through one surface (or element) to the glass media, is refracted and then passes through the other surface back to the air. The first element would be defined with the index of refraction of air on the back side and the index of refraction of glass on the front side. The second surface would be defined with the index of refraction of glass on the back side and the index of refraction of air on the front side. Surfaces other than flat would construct lenses.

Figure 13. A glass plane is actually constructed from two separate elements.

The transmissivity is the fraction of rays (0 to 1.0) that pass through an element. So, if in the above example the glass plane transmits 98% then one of the elements should be defined with a transmissivity of 0.98 and the other 1.0 or both could be defined with transmissivities of 0.99. Currently, for refractive surfaces there is no reflective component.

For reflective optics, one element is usually sufficient to model a mirror since no transmission is allowed. The relevant parameter is the reflectivity and the element still possesses both back and front side values.

For both refractive and reflective optics another set of optical parameters applies. In addition to the effects of the element surface shape on ray direction, two random errors can be included which affect ray interaction at the surface of an element. They are surface slope error and surface specularity. Both are illustrated in Figure 14 for the case of a reflective surface with Gaussian error distribution. Surface slope error is a macro feature while specularity is a micro structure effect. The total error is given by

$$\sigma_{optical} = (4\sigma_{slope}^2 + \sigma_{specularity}^2)^{1/2}$$

If the pillbox distribution is used than replace σ with the half-width of the pillbox. $\sigma_{optical}$ is in terms of the reflected vector. By Snell's Law a slope error of θ results in a reflected vector error of 2θ. Specularity error is already in terms of the reflected vector. Thus the factor of 4 on the σ_{slope} term.

Figure 14. Illustration of surface slope and surface specularity error

Select the distribution type in the box on the lower left of the window and enter appropriate values for the R.M.S. errors to the right. $\sigma_{optical}$ cannot be equal to zero otherwise an error occurs. Therefore, at least one of the two error components should be non-zero. If the user does not wish to include either one of these errors, they can be switched off later before the ray tracing begins. For now, always include some positive value for at least one of these errors.

It is possible to specify the reflectivity or transmissivity of the surface as a function of the incident angle of the impinging ray. To do this, the checkbox next to the corresponding property must be enabled, and values entered into the table in the popup dialog. Enter data corresponding to the incident angle (0 is a ray hitting normal to the surface) and the angular-dependent reflectivity or transmissivity. If this option is specified, the reflectivity or transmissivity single value in the interface is not used. An example is shown in the figure for transmissivity.

Created with the Personal Edition of HelpNDoc: HelpNDoc's Project Analyzer: Incredible documentation assistant

Tracing

Running the Trace

The inputs that determine the behavior of the raytrace simulation are summarized as follows:

- Desired number of ray intersections specifies the minimum number of rays that must hit some part of the geometry in Stage 1 before the simulation can be terminated.
- Maximum number of generated sun rays is the upper limit on the number of rays that can be generated before the simulation will terminate in a failed state. If the simulation terminates due to this number being reached, no results are reported.
- Maximum number of CPUs to utilize is the maximum number of parallel threads that can be allocated to run the simulation. The actual number of threads or CPUs utilized (note -- these may be different things depending on the computer's architecture!) will be the minimum of the number specified and the number of logical processors on the host computer.
- Seed value (-1 for random) is an integer value used to initialize the random number generator for the raytrace simulation. If a number other than (-1) is used, the repeated simulations will produce exactly the same results, provided no other inputs have changed. If the value is (-1), the results will differ according to the inherent uncertainty of the problem.
- Include sun shape specifies whether the sun should be modeled as a perfect light source (i.e., all inbound rays are exactly parallel -- checkbox=False) or whether the sun shape characteristics should be modeled according to the specifications on the Sun page (checkbox = True). The simulation is most realistic in the *checked* state.
- Include optical errors specifies whether reflected or refracted rays should be

modeled as perfect (i.e., rays leave the surface at exactly the ideal angle based on incidence -- checkbox=False) or whether optical errors specified on the Optics page should be modeled (checkbox = True). The simulation is most realistic in the *checked* state.

- Point focus system indicates that the system being modeled consists of a reflector array in stage 1 and a point-like target in stage 2, which allows SolTrace to modify the ray tracing methodology to significantly improve computatational speed for these types of systems. Examples include power tower (central receivers) or dish systems. If enabled, SolTrace reduces ray trace computation by checking only spatially adjacent elements in stage 1 for intersection before and after reflection. It is important to understand that enabling this option can reduce the accuracy of the simulation if the point-focus system might include multiple hits for a single ray on elements that are not near each other spatially.

Further Background Information

When the user specifies a certain number of rays to be traced (how this is done will be described later), rays continue to be randomly generated until that number of intersections has been realized within the elements of Stage 1. A vector is calculated which connects the origin of Stage 1 and the source (e.g. the sun) and ray locations are generated on a plane normal to this vector and within a circle which just encompasses all the projected shapes of the elements within Stage 1. This narrows down the random ray generation region and saves time. The actual application of the sun direction and the sunshape occurs after these intersections have been determined, not before as might be expected. Mathematically it makes no difference when they are applied, but functionally it is more efficient to do so in this way in order to eliminate needless ray generation and, subsequently, time. Once the requested number of rays has intersected Stage 1 somewhere, they are traced to subsequent stages. The 'Trace Through' checkbox thus never appears on the Stage 1 worksheet, because the requested ray number always intersects Stage 1 somewhere.

In certain optical systems, the optical path can fold back on itself. An example would be that of the solar furnace described in Figure 1. A heliostat (Stage 1) reflects sunlight back to a primary concentrator (Stage 2) which in turn focuses and redirects the light back to a target (Stage 3). In this case, Stage 3 could actually lie between Stages 1 and 2. SolTrace does not 'know' this fact. It only knows sequential stage numbers. After the first stage, when a ray leaves a subsequent stage it is automatically traced to the next stage (if the 'Trace Through' box is checked) regardless of whether the ray hit an element within the stage or missed all the elements completely. Thus, if the 'Trace Through' box was checked for Stage 2 and a ray was traced that actually missed all the elements of Stage 2 it would continue to be traced to the target in Stage 3 rather than exiting the system completely which is the physical reality. Since the last known ray location and direction was in Stage 1, the ray would continue to be traced to Stage 3 and would intersect on the backside of the target in Stage 3 which is not the physical reality. If the "Trace Through' box had been unchecked for Stage 2 then this ray would have been handled correctly. Upon missing all elements of Stage 2, it would have been logged as missed and not traced further.

Figure 15. The trace setup parameters

The first input box is the number of rays to be traced. Recall from previous discussions that this many rays will actually interact with elements of Stage 1. The code randomly generates rays on a plane normal to the vector linking the sun with origin of Stage 1. The rays are generated within a circle large enough to just encompass the projected images of all the elements in Stage 1 as seen from the sun. The code continues to generate rays until the requested rays have fallen on elements of Stage 1. The code tallies the total number of rays generated within the circle, both those that fell on an element and those that did not, divides this number into the area of the circle and multiplies the result by the Direct Normal Insolation value in the second input box. This results in a unit power/ray value subsequently used to calculate efficiencies and flux intensities. A seed for the Random Number Generator is internally calculated and is based of a psedorandom computation using the current CPU clock time as a driver.

The maximum number of rays to generate allows the user to limit the maximum number of rays generated, regardless of how many have actually hit Stage 1. This prohibits a trace to spin forever due to an implausibly defined geometry. To avoid hitting this limit in normal operation, it is incumbent upon the user to make sure the number is large enough.

SolTrace allocates memory as it needs it during the trace. Before the trace actually begins, SolTrace calculates the amount of memory it believes is sufficient to do the trace based on the requested ray number and on the assumption that each ray will hit each stage once. It then does one large allocation of memory. It cannot anticipate the possibility of multiple bounces of a ray or rays within a stage. If SolTrace runs out of memory, it begins to grab memory as it needs it from that point forward. This can drastically reduce execution speed. If the user can anticipate this fact, the number of initial memory blocks can be increased with this spin button thereby minimizing or even eliminating these subsequent memory calls and thus execution time.

For computers with multiple processors, SolTrace automatically dispatches threads to concurrently trace rays. By default, it dispatches as many threads as there are processors in the computer. However, the user may elect to only use one processor or two to leave others free for other tasks, or to asses performance characteristics.

Upon click the "Start new trace" button, a progress dialog will prop up that gives statistics about the current status of the ray tracing procedure. On computers with

multiple processors, each processor's progress is shown separately, along with aggregate statistics about the number of rays being traced per second and the total elapsed time.

When the trace has completed, the results are automatically loaded into the visualization and data export modules. Simply click the appropriate toolbar buttons to view graphs or export the data to CSV files or Excel.

Created with the Personal Edition of HelpNDoc: Bring your WinHelp HLP help files into the present with HelpNDoc's easy CHM conversion

Visualization

Two different plot types available within SolTrace: the 3-D ray intersection and path plot, and a 2-D flux distribution contour plot. Figure 16 is an example of a scatter plot showing ray intersections for a power tower system with many heliostat elements in Stage 1 and a single cylindrical element in Stage 2 at the focus of the heliostats. At the upper left of the plot window, the user will notice two boxes, one entitled 'Stages' and the other entitled 'Elements'. In the Stage box will be a list of the stages within the system. Initially the 'Element' box will be empty. Clicking on (or selecting) one or more of the stages within the stage list displays within the 'Element' box the list of elements for each of the selected stages. The surface type for each element is also indicated. See 'Geometry' for the alphanumeric code designation. The last pieces of information shown in the 'Element' box are the number of intersection points for each stage and for each element within each stage. Note that one ray can have multiple intersections with one or more surfaces. One can select more than one stage if desired resulting in the elements for each stage being displayed in the 'Element' box as shown. At the bottom of the 'Stage' list are two buttons whose purpose is straightforward. One selects all the stages with one click, the other unselects all stages with one click. The user can select ray intersections for specific elements to be plotted by checking the boxes next to the elements of interest. The buttons below this box also perform the same function as the previous two for the stage list.

Once the elements of interest have been selected, the user can now choose how to display the ray intersections. Directly above the stage list is a box for selecting the coordinate system to be used for the plot. If elements from multiple stages have been selected, then clearly the global coordinate system would be the logical choice. However, if a set of elements within one stage is selected, the user may wish to plot the ray intersections in the coordinate system for that stage. The same holds true if only one element was selected, the user may wish to plot the results in the coordinate system for that element. The "Final intersections only" checkbox to the right allows the user to display all ray intersections including multiple bounces or just the absorbed or final ray intersections (intersection before ray exits a stage) within a particular stage. There are three options available to the user for displaying the ray intersections. The first is displaying them all as one series or color. Another is to plot ray intersections at each individual element (regardless of stage) as separate series or colors. The last option is to plot ray intersections within each stage as separate series or colors. Ray paths for individual rays can also be shown as lines connecting ray intersection points (useful for tracking ray paths through complicated multiple bounce geometries). Each ray has a number associated with it from 1 to the number of requested rays. Specific ray numbers can be plotted or a range of ray numbers (e.g. '1,2,5,6-10'). Clicking on 'Ray Numbers:' makes the text box available for entering the ray numbers of interest. Rays paths are shown in black with the exception of rays that miss the last displayed stage. The last leg of a ray path that ultimately misses the last displayed stage is shown in white as it exits.

Figure 16

Some statistical information is displayed once the plot is complete. The r.m.s. radius of the distribution from the coordinate system origin and the r.m.s deviation are displayed together with the centroid of the distribution relative to the coordinate system origin. This information is only meaningful for intersections on a flat plane.

Flux maps can be viewed on the **Flux Maps** tab. The contour plot is designed for flux distributions on a plane and therefore requires that the data be for one element with either a flat or cylindrical surface in the element coordinate system. A cylindrical element is "rolled out flat" for plotting purposes. The program will not allow a surface or contour plot if 1) more than one element is selected, 2) a coordinate system other than the element coordinate system is selected, or 3) the element surface is not flat or cylindrical. The number of bins in the x and y directions must be entered. The program calculates the minimum and maximum extents in both directions and uses these by default for plotting, however the user can enter other minimums and maximums as desired. Below the plotting buttons, some statistical information on the flux distribution is displayed. Peak flux, verage flux and the r.m.s radius of the flux distribution are displayed in both the units chosen by the user and also in suns (units of the direct normal irradiance value entered in the 'Trace' form.)

Figure. Flux contour plot with statistics.

Created with the Personal Edition of HelpNDoc: Experience the Power and Simplicity of HelpNDoc's User Interface

Exporting Data

Upon completion of a trace, the raw intersection point data can be viewed and exported on the "Data" page.

	Pos X	Pos Y	Pos Z	Cos X	Cos Y	Cos Z	Element	Stage	Ray
1	23.4694	811.306	0.58305	0.0235106	0.572321	-0.819693	4903	1	1
2	376.288	-26.5281	-1.72874	0.0235106	0.572321	-0.819693	5165	1	2
3	522.291	165.205	-4.49469	0.0235106	0.572321	-0.819693	5058	1	3
4	-1307.06	80.5741	-3.41438	0.0235106	0.572321	-0.819693	1143	1	4
5	57.0752	211.074	-3.82207	0.0235106	0.572321	-0.819693	1082	1	5
6	31.3149	165.077	-1.46726	0.0235106	0.572321	-0.819693	6208	1	6
7	176.803	145.762	3.50468	0.0235106	0.572321	-0.819693	6178	1	7
8	-29.7834	1141.94	-5.07966	0.0235106	0.572321	-0.819693	4409	1	8
9	-486.571	217.568	0.258013	0.0235106	0.572321	-0.819693	1678	1	9
10	-74.6253	281.134	1.22827	0.0235106	0.572321	-0.819693	4208	1	10
11	-198.696	-260.03	-0.0336544	0.0235106	0.572321	-0.819693	-521	1	11
12	168.915	-314.194	1.74333	0.0235106	0.572321	-0.819693	4056	1	12
13	-107.201	1265.75	-0.0735699	0.0235106	0.572321	-0.819693	2420	1	13

Figure 20. Handling raw data points

Tabular data of ray intersections and directions can be copied or loaded into a worksheet by clicking the corresponding button at the top of the table. The data shown in the table is what is saved as a CSV file or exported to Excel. The coordinate system in which the data is to be cast is selected in the upper right of the form.

When the user specifies a given ray count in the 'Trace' form, that number of rays is generated somewhere on the elements of Stage 1. Each ray is numbered from one to the requested ray number and can be identified and traced through the system by this number. The table shown in Figure 20 is a table of intersections for each ray within a particular stage. Each entry contains: 1) the x,y,z location of the ray intersection point within the stage, cast in whatever coordinate system the user chose, 2) the direction

cosines of the ray as it *entered* the stage, 3) the element number within the stage that the ray hit, 4) the associated stage number, and 5) the ray number. If a particular ray had multiple intersections within a stage then there will be multiple entries for that ray number showing the path that it took through that particular stage before exiting. If a ray missed all elements of the stage completely then there will be a zero in the element number column signifying this occurrence. If the 'trace through' for that stage was turned off, then that ray number will no longer show up in any of the following stage data lists as it is considered a lost ray. If the element number is negative for a particular ray intersection, it means that the ray in fact hit this element but was absorbed. Similarly, this ray ceases to be traced though the rest of the system and so will not show up in subsequent stage data lists.

This data is provided so that it can be processed outside of SolTrace with other plotting packages, heat transfer/radiation analysis programs, etc.

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

Scripting

The SolTrace automation capabilities use the LK scripting language developed by NREL. This is the same scripting language used by the Solar Advisor Model (SAM) tool, although several built-in functions are provided for working directly with SolTrace projects. See https://github.com/nrel/lk for general LK language documentation and refer to the scripting function reference for the most up-to-date function information.

1.1 Purpose of Scripting

Scripting can be a useful mechanism to automate SolTrace operations within the application rather than entering values manually or importing externally generated inputs. All SolTrace operations including defining sun, optics, and geometries, running traces, and exporting data can be automated by the scripting functionality. This is useful in cases of modeling large numbers of collectors, inputting complex geometries, or iterating multiple trace operations, in which manual input of parameters is not ideal.

1.2 Opening and Running a Script

A script can be opened in the main SolTrace menu by selecting the three bars at the upper left corner of the window. Select New Script to start a new script or Open Script to open an existing one. Scripts are saved as the '.lk' file type. The Script will be executed by by clicking the 'Run' button on the tool bar.

SolTrace 3.2.0 (64 bit): untitled
Sun
Optics
Geometry
Trace
Intersections
Flux maps
Data
New project Ctrl-N
New script
Open project Ctrl-O
Open script
Save Ctrl-S
Save as...
Close Ctrl-W
Quit
Y
Z
0
0
100
to zenith, Z to north.
0
4.65
0
Pillbox
User defined, # of points:
Predefined...
Load
Save
Copy
Paste
Angle (mrad)
Intensity
1
Sun shape profile
Intensity
Angle from center (mrad)

untitled
New
Open
Save
Save as
Find
Run
Functions
Help
Close
1
Ready.

1.2 Hello world!

As with any new language, the traditional first program is to print the words "Hello, world!" on the screen, and the lk version is listed below.

```
outln( 'Hello, world!' );
```

Notable features:

1. The `outln` command generates output from the script on a new line
2. The text to output is enclosed in single-quotes
3. A semincolon should follow each line of code

To run Hello world, type it into a new SolTrace script, and press the 'Run' button on the toolbar. Now we will learn some more about variables and program control in SolTrace.

Created with the Personal Edition of HelpNDoc: How to Protect Your PDFs with Encryption and Passwords

SolTrace Functions

SolTrace Functions

The functions described in this section comprise the SolTrace library of function calls. For general function calls for data manipulation, operating system access, and calculation, see the LK documentation: https://github.com/NREL/lk/blob/develop/doc/lk_guide.pdf

`dot(array[3]:a, arra[3]:b):real`
Calculates the dot product between two 3 dimensional vectors.

`euler (array[3]:origin, array[3]:aimpoint, real:zrot):array[3]`
Calculates the euler vector given an origin, aimpoint, and z rotation. The Euler angles are described in the Methodology section.

`reftoloc (array[3]:euler):array[3][3]`
Calculates the 3x3 transform matrix from euler angles for going from reference to local coordinates. This is equivalent to the matrix provided at the end of the Methodology section.

`loctoref (array[3]:euler):array[3][3]`
Calculates the 3x3 transform matrix from euler angles for going from local to reference coordinates. This is the transpose of the matrix outputted by `reftoloc`

`toloc (array[3]:posref, array[3]:cosref, array[3]:origin, array[3][3]:reftoloc, array[3]:posloc, array[3]:cosloc):void`
Transforms a reference position/cosine point description to local coordinates given the transform matrix. The cosine point describes the pointing direction of the z-axis in the respective coordinate system.

`toref (array[3]:posloc, array[3]:cosloc, array[3]:origin, array[3][3]:loctoref, array[3]:posref, array[3]:cosref):void`
Transforms a reference position/cosine point description to reference coordinates given the transform matrix. The cosine point describes the pointing direction of the z-axis in the respective coordinate system.

`transpose (array[3][3]):array[3][3]`
Transposes a 3x3 matrix.

`matvecmult (array[3][3]:m, array[3]:v):array[3]`
Multiplies a 3x3 matrix and a vector, returning the result.

`workdir (void):string`
Returns the current working directory for SolTrace.

`workdir (string:path):void`
Sets the current working directory for SolTrace.

`file_name (void):string`
Returns the current SolTrace project file name.

`save_project ({string:optional file name}):boolean`
Saves the current SolTrace project as an .stinput file.

`open_project (string:filename):boolean`
Writes a binary ray data file with the current trace results.

`clear_project (void):void`
Clears the current SolTrace project, without asking to save or close first.

`writerayfile (string:file):boolean`
Writes a binary ray data file with the current trace results.

`trace ([string:errors]):integer`
Starts a new trace operation. Returns time elapsed in milliseconds, or negative if an error occurred or was cancelled. Optionally, fills the 1st argument with error messages if passed in.

`traceopt (table:parameters):integer`
Sets various ray trace parameters. The argument is a table with keys
{rays, maxrays, cpus, seed, include_sunshape, optical_errors, point_focus}

`nintersect (void):integer`
Returns the number of ray intersections in the results.

`nintersect (integer:stagenum, integer:elementnum):integer`
Returns the number of ray intersections with a particular element.

`raydata (integer:index):array`
Returns the ray data as a 9 item array in stage coordinates

`sundata (void):table`
Returns the number of generated sun rays and the bounding box extents as a table with fields {nrays, xmin, xmax, ymin, ymax}.

`sunopt (table:parameters):void`
Sets various sun shape parameters as a field indexed table whose fields are {ptsrc, x, shape, sigma, y, halfwidth, z, useldh, lat, day, hour, userdata}

`sunopt (void):table`
Gets various sun shape parameters as a field indexed table whose fields are {ptsrc, x, shape, sigma, y,

halfwidth, z, useldh, lat, day, hour, userdata}

`addoptic (string:name):void`
Adds a new optical property set with the given name.

`clearoptics (void):void`
Deletes all of the optical property sets.

`listopics (void):array`
Returns a list of all the optical property sets.

`opticopt (string:name, integer: front or back, table:properties):void`
Set various optical properties fro a the given optic name and side (1=front, 2=back) with a table whose fields are {dist, apstrop, surfnum, difford, refl, trans, errslope, refractr, refracti, grating, reftable}

`addstage (string:name):void`
Adds a stage to the system with the given name. Sets the new stage as the currently active stage.

`clearstages (void):void`
Clears all stages from the project.

`liststages (void):array`
Returns a list of all the stage names.

`stageopt (string:stage name):table`
Set various stage properties for the given stage name with a table whose fields can include {virtual, multihit, tracethrough, x, y, z, ax, ay, az, zrot}

`stageopt (string:stage name):table`
Get stage properties for the given stage name as a table.

`activestage (string: stage name):string`
Set the currently active stage. Returns the active stage name also.

`addelement ([integer:optional number of elements]):void`
Adds one (or optionally more) elements to the active stage.

`clearelements (void):void`
Clears all the elements from teh currently active stage.

`nelements (void):integer`
Returns the number of elements in the current stage.

`elementopt (intger:element index, table:properties):void`
Set various element properties fro the specified element in the active stage. Table fields are {en, x, y, z, ax, ay, az, zrot, aper, surf, interact, optic, comment}.

`elementopt (intger:element index):table`
Gets the properties fo the specified element in the active stage.

`elementstats (intger:stage index, integer: element index, integer: x bins, integer: y bins, double:dni, boolean:final rays only, [double:minx, double:maxx, double:miny, double:maxy]):table`
Returns a table of statistical information about a particular element. The variables included in the

outputted table are:
{min_flux, bin_size_x, power_per_ray, bin_size_y, peak_flux, peak_flux_uncertainty, uniformity, ave_flux, ave_flux_uncertainty, sigma_flux, radius, num_rays, centroid, xvalues, yvalues, flux}

`rayhits (integer:stage index, integer:element index, [boolean:final only]):integer`
Returns the number of ray hits on an element.

Created with the Personal Edition of HelpNDoc: Maximize Your Productivity with a Help Authoring Tool

Scripting Examples

Scripting Examples

Located in the scripts directory in the SolTrace files, there the following scripting examples:

importgeo.lk

This script allows a user to directly import a 3D CAD file saved in the GEO format to automatically define collector geometry inputs.

statistics.lk

This script is meant to be run with the High Flux Solar Furnace project example (High Flux Solar Furnace.stinput found in the samples directory). The script outputs statistical information about the ray hits and flux distribution on the target.

system_setup_1.lk

This script provides an example of how each SolTrace module can be configured and trace can be run. The script configures sun, optics, and geometry stage/element properties and runs a ray trace.

tmy3.lk

This script reads a user provided TMY3 data set, documentation provided here:
https://www.nrel.gov/docs/fy08osti/43156.pdf

Created with the Personal Edition of HelpNDoc: News and information about help authoring tools and software

References

Steele, C.R., Balch, C.D., Jorgensen G.J., Wendelin, T. and Lewandowski, A., 1991, “Membrane Dish Analysis: A Summary of Structural and Optical Analysis Capabilities,” NREL/TP-253-3432, National Renewable Energy Laboratory, Golden, CO.

Ratzel, A.C., and Boughton, B.D., 1987, “CIRCE.001: A Computer Code for Analysis of Point-Focus Concentrators with Flat Targets,” SAND86-1866, Sandia National Laboratories, Albuquerque, NM.

Spencer, G.H., and Murty, M.V.R.K., 1962, “General Ray-Tracing Procedure,” *Journal of the Optical Society of America*, Vol, 52, June, pp.672-678.

Neumann, A., A. Witzke, S.A. Jones, G. Schmitt, "Representative Terrestrial Solar Brightness Profiles," *Journal of Solar Energy Engineering*, 124, pp. 198-204, May, 2002.

Created with the Personal Edition of HelpNDoc: Benefits of a Help Authoring Tool