MatLightTracer Version 1.0 Reference

**Table of contents**

[Cover Page](#_topic_CoverPage) 4

[Objective](#_topic_Objective) 7

[User Manual](#_topic_UserManual1) 10

[Downloading the MatLightTracer Toolbox](#_topic_DownloadingtheMatLightTracerTool) 13

[General Toolbox Folder Structure](#_topic_GeneralToolboxFolderStructure) 16

[Starting the Toolbox](#_topic_StartingtheToolbox) 20

[Conventions and Definitions](#_topic_ConventionsandDefinitions) 24

[Using MatLightTracer GUI](#_topic_UsingMatLightTracerGUI) 27

[Parent Window](#_topic_ParentWindow) 31

[Title Bar](#_topic_TitleBar) 34

[Menu Bar](#_topic_MenuBar) 37

[Surface/Component List Panel](#_topic_SurfaceComponentListPanel) 41

[Surface/Component Parameter Panel](#_topic_SurfaceComponentParameterPanel) 45

[System Quick Layout Panel](#_topic_SystemQuickLayoutPanel) 63

[Status Bar](#_topic_StatusBar) 66

[Children Windows](#_topic_ChildrenWindows) 69

[System Configuration Window](#_topic_SystemConfigurationWindow) 72

[Setting up a New Optical System](#_topic_SettingupaNewOpticalSystem) 79

[Opening Optical Systems from File](#_topic_OpeningOpticalSystemsfromFile) 82

[System Layout](#_topic_SystemLayout) 85

[2D System Layout Window](#_topic_2DSystemLayoutWindow) 88

[3D System Layout Window](#_topic_3DSystemLayoutWindow) 91

[Optical System Analysis](#_topic_OpticalSystemAnalysis1) 94

[Scalar Ray Trace](#_topic_ScalarRayTrace) 97

[Paraxial Analysis](#_topic_ParaxialAnalysis) 101

[Transverse Ray Aberration](#_topic_TransverseRayAberration) 104

[Longitudinal Aberration](#_topic_LongitudinalAberration) 107

[Footprint Diagram](#_topic_FootprintDiagram) 110

[Hybrid Diffraction](#_topic_HybridDiffraction) 113

[Wavefront @ Exit Pupil](#_topic_WavefrontExitPupil) 116

[FFT PSF](#_topic_FFTPSF) 119

[Pupil Apodization](#_topic_PupilApodization) 122

[Polarization](#_topic_Polarization) 125

[Polarization Ellipse Map](#_topic_PolarizationEllipseMap) 127

[Polarization Ray Trace](#_topic_PolarizationRayTrace) 130

[Polarization Aberration:](#_topic_PolarizationAberration) 134

[Amplitude Transmission Map](#_topic_AmplitudeTransmissionMap) 137

[Phase Map](#_topic_PhaseMap) 140

[Wavefront Diattenuation Map](#_topic_WavefrontDiattenuationMap) 143

[Wavefront Retardation Map](#_topic_WavefrontRetardationMap) 146

[Pulse Analysis](#_topic_PulseAnalysis) 149

[Coating](#_topic_Coating1) 152

[New Coating Catalogue](#_topic_NewCoatingCatalogue) 155

[Coating Data Editor](#_topic_CoatingDataEditor) 158

[Coating Analysis](#_topic_CoatingAnalysis1) 161

[Refractive Index Profile](#_topic_RefractiveIndexProfile) 164

[Coating Property Versus Incidence Angle](#_topic_CoatingPropertyVersusIncidenceAn) 167

[Coating Properties vs Wavelength](#_topic_CoatingPropertiesvsWavelength) 171

[Glass](#_topic_Glass) 175

[Glass Data Editor](#_topic_GlassDataEditor) 178

[New Glass Catalogue](#_topic_NewGlassCatalogue) 181

[Help](#_topic_Help) 184

[About](#_topic_About) 187

[User Manual](#_topic_UserManual) 190

[Programming and Algorithms Reference](#_topic_ProgrammingandAlgorithmsReferenc) 193

[Source Code Viewer](#_topic_SourceCodeViewer) 196

[Using MatLightTracer From Matlab Command](#_topic_UsingMatLightTracerFromMatlabCom)  199

[Loading an Optical System](#_topic_LoadinganOpticalSystem) 202

[Creating Initial Rays For Optical Systems](#_topic_CreatingInitialRaysForOpticalSys) 205

[Tracing Rays Through Optical Systems](#_topic_TracingRaysThroughOpticalSystems) 209

[Accessing Ray Tracing Results](#_topic_AccessingRayTracingResults) 214

[Programming and Algorithms Reference](#_topic_ProgrammingandAlgorithmsReferen1) 218

[Programming Approach](#_topic_ProgrammingApproach) 221

[Toolbox Structures and Classes](#_topic_ToolboxStructuresandClasses) 224

[GUI Related Class](#_topic_GUIRelatedClass) 228

[AODParentWindow: Class](#_topic_AODParentWindowClass) 231

[AODChildWindow: Class](#_topic_AODChildWindowClass) 234

[OpticalSystem: Struct](#_topic_OpticalSystemStruct) 237

[Surface: Struct](#_topic_SurfaceStruct) 242

[Glass: Struct](#_topic_GlassStruct) 248

[Coating: Struct](#_topic_CoatingStruct) 251

[Scalar Ray Tracing Module](#_topic_ScalarRayTracingModule) 255

[Coordinate Transformations](#_topic_CoordinateTransformations) 258

[Intersection Point and Surface Normal Computation](#_topic_IntersectionPointandSurfaceNorma) 261

[Refraction (or Reflection)](#_topic_RefractionorReflection) 267

[Incidence Angle](#_topic_IncidenceAngle) 270

[Paraxial and Meridional Ray Tracing Module](#_topic_ParaxialandMeridionalRayTracingM) 273

[Paraxial Ray Tracing Module](#_topic_ParaxialRayTracingModule) 276

[Meridional Ray Tracing Module](#_topic_MeridionalRayTracingModule) 280

[Polarization Ray Tracing Module](#_topic_PolarizationRayTracingModule) 284

[Jones Matrix to Polarization Ray Tracing Matrix](#_topic_JonesMatrixtoPolarizationRayTrac) 289

[Polarization Ray Tracing through Interfaces](#_topic_PolarizationRayTracingthroughInt) 293

[Polarization Properties of Optical System](#_topic_PolarizationPropertiesofOpticalS) 298

[Examples of Extending the Toolbox](#_topic_ExamplesofExtendingtheToolbox) 303

[Method 1: As part of the toolbox](#_topic_Method1Aspartofthetoolbox) 306

[Method 2: As separate tool using the existing toolbox](#_topic_Method2Asseparatetoolusingtheexi) 309

[Add New Surface Types](#_topic_AddNewSurfaceTypes) 312

[Transverse Ray Aberration Diagram](#_topic_TransverseRayAberrationDiagram) 327

[Procedures](#_topic_Procedures) 330

[Results](#_topic_Results) 340

[Longitudinal Ray Aberration Diagram](#_topic_LongitudinalRayAberrationDiagram) 343

[Procedures](#_topic_Procedures1) 346

[Results](#_topic_Results1) 350

**Cover Page**

MatLightTracer : MATLAB Light Tracing Toolbox

User Manual and

Programming & Algorithms Reference

Version 1.00

Optical System Design Research Group

Institute of Applied Physics

Friedrich Schiller University of Jena

01/06/2015

Norman G. Worku

Advisor: Prof. Dr. Herbert Gross

*Created with the Personal Edition of HelpNDoc:* [*Generate EPub eBooks with ease*](http://www.helpndoc.com/create-epub-ebooks)

**Objective**

# Objective

MatLightTracer is a GUI based optical system analysis toolbox in MATLAB, which is being designed by optical system design and simulation research group, Institute of Applied Physics, Friedrich Schiller University of Jena.

This document is divided in to two main parts, the user manual and the programming and algorithms reference, each with different objectives.

**The User Manual**

The objective of the user manual document is to give a brief user manual for using the toolbox from its graphical user interface windows. Even though the functions and classes in the toolbox are commented and well documented, this document makes the toolbox more understandable and easy for future users and developers. The user interfaces and their functionalities are described in detail. But more technical and detailed programming reference is given in the second part of the document, the programming and algorithms reference part.

**The Programming and Algorithms Reference**

The objective of this part of the document is to give a short introduction to different modules and mathematical algorithms used in the toolbox and act as programming reference to any programmer using and working on the toolbox.

Please send any comments, corrections or suggestions by email to **''Norman G. Worku <normangirma2012@gmail.com> or <norman-girma.worku@uni-jena.de>"**. They shall be recorded and included in the next versions.

*Created with the Personal Edition of HelpNDoc:* [*Free Web Help generator*](http://www.helpndoc.com)

**User Manual**

MatLightTracer : MATLAB Light Tracing Toolbox

User Manual

Version 1.0

*Created with the Personal Edition of HelpNDoc:* [*Easily create Web Help sites*](http://www.helpndoc.com/feature-tour)

**Downloading the MatLightTracer Toolbox**

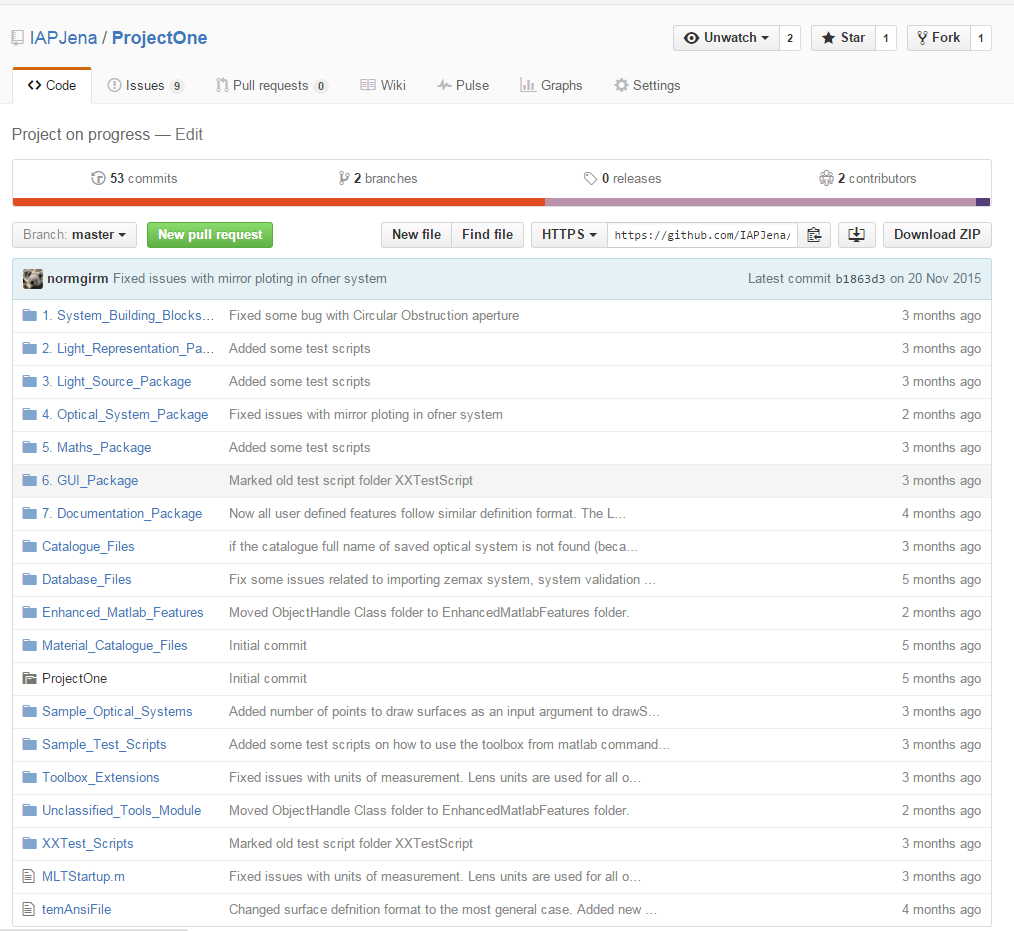
# Downloading MatLightTracer Toolbox

The MatLightTracer Toolbox can be easily downloaded from its github page

https://github.com/IAPJena/ProjectOne

In the github project folder it is possible to

* Download the updated version of the toolbox.
* Submit any issues, bugs and comments on the toolbox using the issue section.
* Contribute to the toolbox (This requires preregistration).



*Created with the Personal Edition of HelpNDoc:* [*Free Web Help generator*](http://www.helpndoc.com)

**General Toolbox Folder Structure**

# General Toolbox Folder Structure

All codes for the toolbox are organized in to different packages and folders based on their functions. The main folder for the toolbox is "MatLightTracer\_GitHub". The following table summarized the folders and packages with brief description.

|  |  |  |
| --- | --- | --- |
| **Folder (Package) Name** | **Description** | **Sub Folder (Module) Name** |
| System\_Building\_Blocks\_Package | In this package modules used to define building blocks of an optical system are included. These include Aperture, Coating, Glass, Grating, Material Catalogue, and Optical Elements (Surfaces and Components) modules. Each module contains the struct file for respective building blocks and defintion files for some building blocks. All building blocks are implemented in such a way that a user defined building block of certain kind can be added by just writing a single .m file for defining the new element (See Extension of Toolbox section in the Programing reference manual). | Optical\_Elements/Surface\_Module |
| Optical\_Elements/Component\_Module |
| Aperture\_Module |
| Glass\_Module |
| Coating\_Module |
| Material\_Catalogue\_Module |
| Grating\_Module |
| Light\_Representation\_Package | This package includes different representations of light in optical system. In this version it includes Ray, Gaussian and Harmonic\_Field based modules. Each module in turn includes classes and functions related to the corresponding light representation. | Ray\_Module |
| Gaussian\_Module |
| Complex\_Field\_Module |
| Light\_Source\_Package | Here functions used to define general electric field source are included. In the toolbox a general electric field source can be defined by specifying the spatial, spectral and polarization profile of the field generated. All the profiles are implemented as a user defined functions. It also includes a GUI function which is used to define an electric field source. | HarmonicFieldSource |
| Complex\_Field\_Spacial\_Profile\_Defintions |
| Complex\_Field\_Spectral\_Profile\_Defintions |
| Complex\_Field\_Polarization\_Profile\_Defintions |
| GaussianBeamDecomposition |
| Optical\_System\_Package | This package includes modules related to define OpticalSystem struct, optical system analysis, RayTraceResult struct, and Optical system import export functions. | OpticalSystem |
| Optical\_System\_Analysis\_Module |
| Optical\_System\_Import\_Export |
| RayTraceResult |
| Enumeration\_Module |
| Math\_Package | This folder is Maths package of the toolbox containing all mathematical functions used in the toolbox. | Fourier\_Transforms |
| Other\_Mathematical\_Functions |
| GUI\_Package | All functions related to the main GUI of the toolbox including all child windows is included here. | @ParentWindow |
| @ChildWindow |
| GeneralObjectInputGUI |
| Documentation\_Package | It contains help files, user manuals, programming reference, scientific research papers and any other files which can be used as help for the toolbox. | User\_Manual\_And\_Technical\_Reference |
| Source\_Code\_Explorer |
| Matlab\_Reference\_Books |
| Catalogue\_Files | Contains catalogue files used in the optical system. The catalogue files include Glass and Coating catalogue files. It is default folder for saving new and accessing existing catalogue files. |  |
| Database\_Files | To store any files which are used by the toolbox such as GUI Icons, Zemax Glass catalogues, Zemax sample optical systems ... |  |
| Enhanced\_Matlab\_Features | Sometimes it becomes necessary to enhance certain features of Matlab before using in the toolbox so this folder has collection of all such enhanced features. Those features can also be used outside the toolbox. |  |
| PROGRESS\_Files | FIles, functions and features which are not complete yet. |  |
| Toolbox\_Extensions | In the design of the MatLightTrracer toolbox simplifying the future extension was given great attention. This folder is dedicated to store all analysis features which are implemented as external extension of the toolbox. |  |
| Unclassified\_Tools\_Module | All functions and scripts which doesn't belong to any of the above folders are put here. |  |
| XXFiles | It contains all archived functions which are no longer used in the toolbox. |  |
|  |  |  |

*Created with the Personal Edition of HelpNDoc:* [*Generate EPub eBooks with ease*](http://www.helpndoc.com/create-epub-ebooks)

**Starting the Toolbox**

# Starting the Toolbox

**1. Using the MatLightTracer App**

For MATLAB 2012b or latest users, the toolbox is deployed as a MATLAB app, which is a single file that can be installed to App gallery of MATLAB. All issues regarding the path and working directory are handled automatically. So to start using the toolbox,

* Install the app, which is done just by opening the installable file.
* Start the toolbox by clicking on the App from the App gallery in MATLAB.
* Once installed and started the toolbox can be used from the MATLAB Command Window
  + Use the function names directly in the command window.

**Note:** All the source files and the different modules of the toolbox can be seen by browsing through the folder where the App is installed.

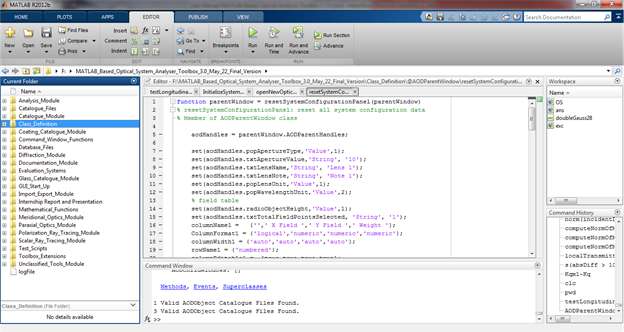
**2. Using the source code folder**

For users of earlier versions of MATLAB, the toolbox folder shall be copied to the destination computer. And then the toolbox can be used from either from the GUI or the command window as follows.

1. Using from the GUI Windows
   * Include the Toolbox main folder with all of its sub folders to the current working directory of the MATLAB.
   * Make the Toolbox main folder your current directory.
   * To open the Main GUI window of the toolbox, write 'MLTStartup' in the MATLAB command window.
   * Then use the using the GUI to set up and analyze an optical system.
2. Using form Matlab Command Window or Other Matlab scripts
   * Include the Toolbox main folder with all of its sub folders to the current working directory of the MATLAB.
   * Use the function names directly. But make sure that you read the comment section of the function before using to make sure that you are giving appropriate input parameters.

This user manual is used for the GUI based operation of the toolbox. The command based operation will be included in the future.

**Note:** As the toolbox uses 'pwd' commands to get the toolbox directory, it is must to make the toolbox folder current working folder otherwise some features may fail. Your current directory should be as shown below



*Created with the Personal Edition of HelpNDoc:* [*Free HTML Help documentation generator*](http://www.helpndoc.com)

**Conventions and Definitions**

# Starting the Toolbox

# Unit for lengths

Optical system has defined Lens and wavelength units. Everything related to optical system will be in those units. They include

* all surface parameters, optical system parameters,
* returns of functions operating on optical systems such as rayTracer results, multipleRayTracer results,…

But for other things which are not directly related to the optical system such as

* Ray objects, returns of getChiefRay, getInitialRayBundle, getMariginalRay,
* returns of getPrimaryWavelength, getSystemWavelengths, getSystemFieldPoints,

the SI units are used for both wavelength and lengths.

*Created with the Personal Edition of HelpNDoc:* [*Easily create CHM Help documents*](http://www.helpndoc.com/feature-tour)

**Using MatLightTracer GUI**

# Using MatLightTracer GUI

The MatLightTracer toolbox has its own GUI which could be used to setup and analyze optical systems. This section of the user manual describes basic steps to use the user interfaces of the toolbox. As describe in the previous section all functionalities of the toolbox can also be accessed with out using the GUI. That enables integration of the toolbox to other projects and is discussed in the last section section of the user manual.

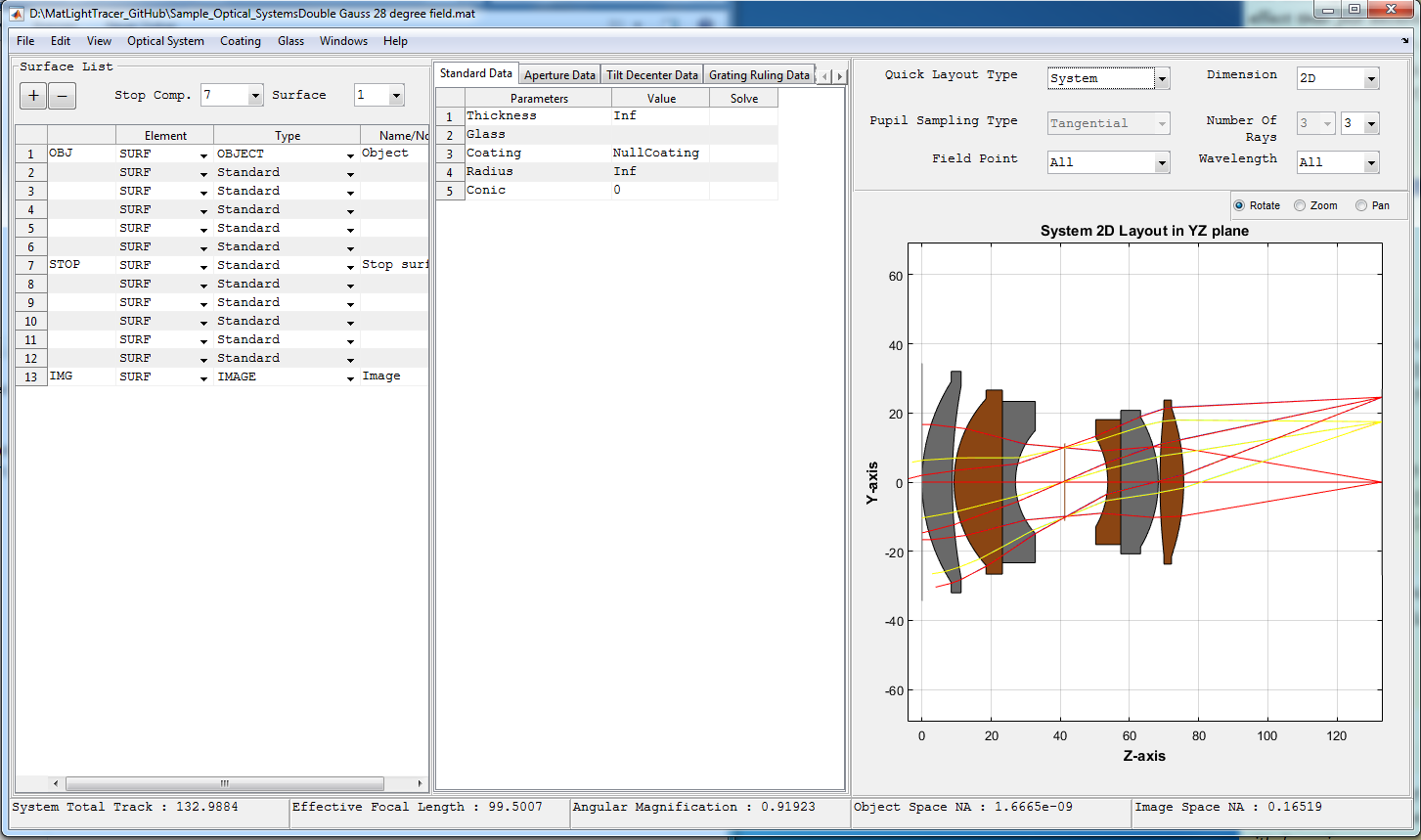
# 

*Created with the Personal Edition of HelpNDoc:* [*Easily create HTML Help documents*](http://www.helpndoc.com/feature-tour)

**Parent Window**

**Parent Window**

Writing "MLTStartup" in the command window starts the main GUI of the toolbox shown below.



The main window has the following main parts:

* **Title Bar:**
  + The full path name of the current optical system is displayed.
* **Menu Bar:**
  + It contains Menu items which are used to use different features of the toolbox.
  + For detailed description see the "Menu Bar" section below.
* **Element List panel:**
  + List of surfaces or components in our optical system. Buttons to add and remove surfaces is also inclued here. The Stop index can also be easily changed.
* **Element Parameter Panel:**
  + Here all parameters of the selected surface can be seen and changed.
* **System Quick Layout Panel:**
  + Used to show the layout of current system or just the selected component or surface. This enables quick view of the system as it is being setup.
* **Status Bar:** 
  + It is at the bottom of the window and used to display important information of the current optical system.

*Created with the Personal Edition of HelpNDoc:* [*Easily create Web Help sites*](http://www.helpndoc.com/feature-tour)

Title Bar

**Title Bar**

* + The full path name of the current optical system is displayed.



*Created with the Personal Edition of HelpNDoc:* [*Produce Kindle eBooks easily*](http://www.helpndoc.com/feature-tour/create-ebooks-for-amazon-kindle)

Menu Bar

**Menu Bar**

* + It contains Menu items which are used to use different features of the toolbox.



The general structure of the menu items in the menu bar is shown here.

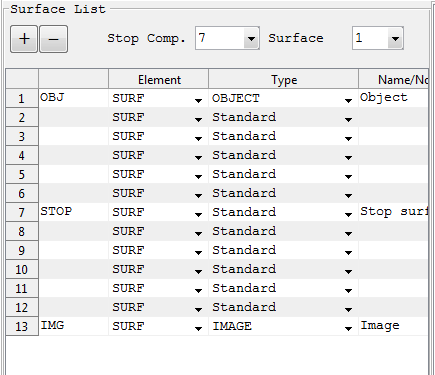
* **File:** 
  + **New:** Opens new surface editor window resetting all the parent window and closing all child windows.
  + **Open:** Opens Windows “Open File” dialog box which can be used to open previously saved optical systems by the toolbox.
  + **Save:** Saves any changes on the current optical system to the current file name or opens Windows “Save As” dialog box if the system is not yet saved.
  + **Save As:** Opens Windows “Save As” dialog box which enables saving current optical system to a .mat file in a user defined location.
  + **Close:** Closes the application.
* **Edit:** 
  + **System Configuration:** Opens the “Optical System Configuration” window.
* **View:** 
  + **2D System Layout:** Plots the two dimensional cross sectional view of rotationally symmetric optical systems.
  + **3D System Layout:** Plots the three dimensional system layout for general optical systems.
* **Optical System:**
  + **Import:**
    - **From ZMX File:** Enables importing .zmx file formats from zemax system files.
  + **Optical System Analysis:**
    - **Paraxial Analysis:** Performs paraxial analysis and displays the result in the command window.
    - **Scalar Ray Trace:** Opens single ray data entry window which enables tracing of a single ray through current optical system.
    - **Footprint Diagram:** Plots footprint diagram of a bundle of ray on a given surface of an optical system.
    - **Ray Aberration:** 
      * **Transverse Ray Aberration:** Plots the transverse ray aberration with respect to the chief ray.
      * **Longitudinal Ray Aberration:** Plots the transverse ray aberration with respect to the chief ray.
    - **Hybrid Diffraction:**
      * **Wavefront @ Exit Pupil:** Computes and displays the wavefront surface which corresponds to OPD surface at exit pupil.
      * **Pupil Apodization:** Displays graphical apodization profile of the current system.
      * **FFT PSF:** Computes Fast Fourier Transform based point spread function for a given optical system.
    - **Polarization:**
      * **Polarization Ray trace:** Opens single polarized ray data entry window which enables polarization ray tracing of a single ray through current optical system.
      * **Polarization Ellipse Map:** Shows distribution of polarization ellipse over the pupil area for a given polarized ray at a given surface.
      * **Polarization Aberration:** Displays different graphs showing the polarization aberration of the optical system.
* **Coating:** 
  + **Coating Data Editor:** Displays a window that enables to see, edit, add or remove coating from the coating catalogues used in the current optical system.
  + **Coating Analysis:** Displays different graphs to analyze any coating in the coating catalogue.
  + **Coating Catalogue:**
    - **New Coating Catalogue:** Adds new empty coating catalogue to the default catalogue file folder.
* **Glass:** 
  + **Glass Data Editor:** Displays a window that enables to see, edit, add or remove glass from the glass catalogues used in the current optical system.
  + **Glass Catalogue:**
    - **New Glass Catalogue:** Adds new empty glass catalogue to the default catalogue file folder.
    - **Import Glass Catalogue:**
      * Import glass from other formats. Currently Schott catalogue on excel format are supported but in the future more formats shall be defined.
* **Help:**
  + **About:** Short description of the toolbox.
  + **User Manual:** Open the electronic form of the user manual.
  + **Programming Reference:** Open the programming reference document for the toolbox.
  + **Source Code Explorer:** Used to view source codes of the toolbox in html format.

*Created with the Personal Edition of HelpNDoc:* [*Full-featured EBook editor*](http://www.helpndoc.com/create-epub-ebooks)

Surface/Component List Panel

**Optical Element List Panel**

* + List of optical elements (surfaces or components) in our optical system. Buttons to add and remove surfaces is also included here. The Stop index can also be easily changed.



|  |  |  |
| --- | --- | --- |
| **Column Header** | **Description** | **Remarks** |
| Element | Used to indicate the optical element is a surface or a component | SURF: For Surface  COMP: For Component |
| Type | A pop up menu to choose the type of surface or component | Currently supported  Surfaces: Standard, Even Aspheres, Odd Aspheres, Ideal Lens, Torroidal, Dummy, Kostenbauder surfaces are functional.  Components: Sequence\_Of\_Surfaces, Grating1D and Prism are included. |
| Name/Comment | To enter name and notes related to the surface. | Any text inputs are accepted and it has no real functional significance. |

**Note:**

* Object and image surfaces are always the first and the last surfaces of the system.
* To add or remove an optical element select the row indicating the index of element to be removed or to be added, then click + (Add) or -(Remove) button.
* To make an optical element a stop, select its index in the pop up menu. For components made of multiple surfaces it is necessary to select the surface index for the stop.

*Created with the Personal Edition of HelpNDoc:* [*Full-featured Kindle eBooks generator*](http://www.helpndoc.com/feature-tour/create-ebooks-for-amazon-kindle)

Surface/Component Parameter Panel

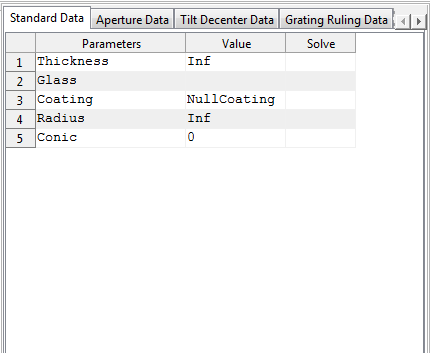
**Surface/Component Parameter Panel**

It is part of the main panel of the parent window and contains spreadsheet like data editors for entering and editing the surface data of an optical system.

It is organized in to different tabs: Standard data, Aperture Data, Tilt and Decenter Data and Others . Each of the tabs are discussed below.

**1. Standard Data**

It contains surface data which are required for conventional optical systems.



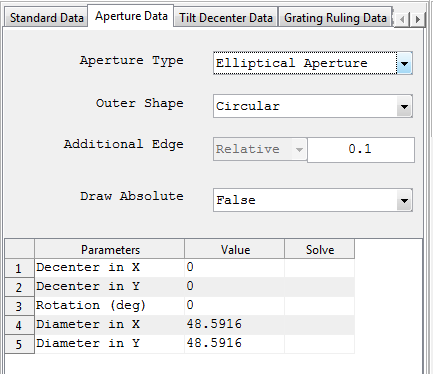
|  |  |  |
| --- | --- | --- |
| **Row Header** | **Description** | **Remarks** |
| Thickness | To enter the thickness the medium that follows the current surface. The unit of the number will be that specified in the “System Configuration” window. | All numeric values including 0 and Inf are accepted by the system. Other inputs are not allowed. But entering Inf thickness for surfaces other than the object surface and the last surface of the system will result in an invalid optical system. |
| Glass | To enter the glass name or the refractive index of the medium that follows the current surface. | Any number value is treated as refractive index of the medium. And any non-numerical text will be treated as the name of the glass from glass catalogue. If the glass name entered exists in the catalogue, it will be confirmed by changing the entered text to upper case. If the system fails to find the glass name entered then, the Glass data editor window will automatically appear to enable user select or enter the new glass. |
| Coating | Name of coating used. "None" for bare glass. | Any non-numerical text will be treated as the name of the glass from coating catalogue. If the coating name entered exists in the catalogue, it will be confirmed by changing the entered text to upper case. If the system fails to find the coating name entered then, the Coating data editor window will automatically appear to enable user select or enter the new coating. |
| Radius | Used to input the radius of curvature of the surface. The unit of the number will be that specified in the “System Configuration” window. | All numeric values including 0 and Inf are accepted by the system. Other inputs are not allowed. The surface type and radius fields are interlinked and changing one automatically affects the other. For instance, selecting plane surface for the surface type automatically sets the radius to infinity. And changing the radius to some finite value changes the surface type to Spherical if it is Plane. |
| Conic | Used to input the conic constant of the surface. | All numeric values including 0 and Inf are accepted by the system. |

NB:

* In addition to the above standard parameters which are common to all surfaces, there are also specific parameters defined for specifiec surface types. Those parameters include for example the polynómial coefficnets for the even aspher types.
* Unlike most optical design softwares, the object surface is considered to be the first surface in the toolbox. This is follows from the fact that indexing begins from index 1 in Matlab.

**2. Aperture Data**

It contains data defining surface apertures.

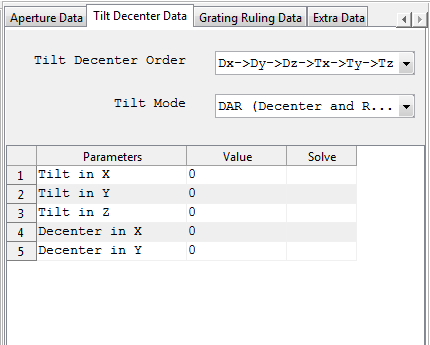


|  |  |  |
| --- | --- | --- |
| **Field Name** | **Description** | **Remarks** |
| Aperture Type | List of all supported apertures from which one can be selected. | The default value is Floating Circular Aperture |
| Outer Shape | The outer shape of aperture encompassing the real aperture. This is the one used for plotting the aperture in system layout. | For simplicity it can only be either elliptical or rectangular. other shapes are not allowed. |
| Additional Edge | The additional edge of aperture used in plotting. | Currently the relative value can be entered. That is the fraction of the aperture size added as additional edge. |
| Draw Absolute | A boolean value indicating weather to draw the surface aperture absolutely in the layout. If it is set to false, then the aperture of two surfaces separated by glass (singlets) is drawn by taking the maximum size of apertures of the two surfaces. |  |
| The following three parameters are also common to all aperture types. | |  |
| Decenter in X | Aperture Decenter in X | Only positive numerical values are allowed |
| Decenter in Y | Aperture Decenter in Y | Only positive numerical values are allowed |
| Rotation (deg) | Rotation angle of the aperture measured from the +ve y axis. | The angle is in degree. |

NB: All other parameters vary depending on the type of aperture chosen. So the documentation of each aperture type shall be refereed before using them.

**3. Tilt and Decenter Data**

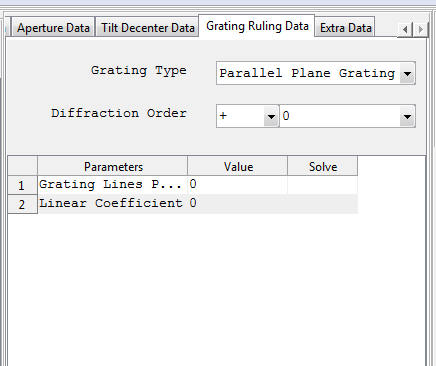
It includes tilt and decenter parameters of the surfaces.



|  |  |  |
| --- | --- | --- |
| **Row Header** | **Description** | **Remarks** |
| Tilt Mode | Pop up menu for selecting the mode of the tilt determining the reference coordinate axis after the current surface. | Currently there are three types of tilt modes supported. DAR (Decenter and Return), NAX (New Axis) and BEN (Bend Axis) |
| Tilt Decenter Order | Order in which tilt and decenter are performed. | Currently only two orders are allowed. Tilt after denceter and decenter after tilt. |
| Decenter X,Y | Surface decenter in X and Y coordinate | Non infinite numeric values are allowed. |
| Tilt X,Y,Z | Tilt angle (in deg) about the corresponding axes. | Currently they are used as successive rotation angles in degrees. |

**4. Grating Ruling Data**

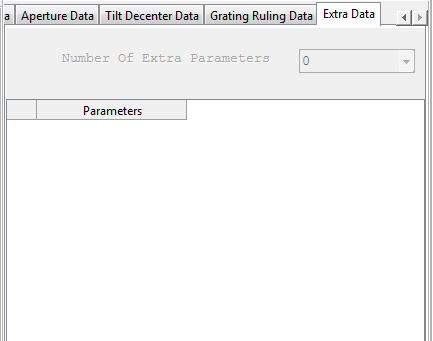
It includes grating structure data associated with surfaces which support specification of ruled grating.



|  |  |  |
| --- | --- | --- |
| **Row Header** | **Description** | **Remarks** |
| Grating Type | Type of grating ruling (Parallel Plane or Concentric Cylindrical) | Currently there are three types of tilt modes supported. DAR (Decenter and Return), NAX (New Axis) and BEN (Bend Axis) |
| Diffraction Order | Diffraction order to be considered. |  |
| Grating Line per Micrometer | Grating Line per Micrometer |  |
| Linear Coefficient | Coefficient used to specify the linear variation of the grating line density on the position of the point. |  |

**5. Extra Data**

It is used to input extra parameters of the surfaces.

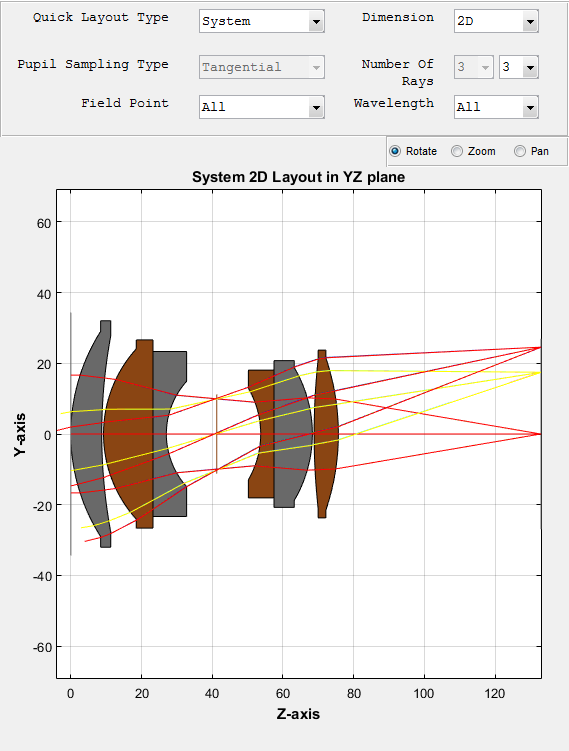


*Created with the Personal Edition of HelpNDoc:* [*Easy EBook and documentation generator*](http://www.helpndoc.com)

System Quick Layout Panel

**System Quick Layout**

* + Used to show the layout of current system or just the selected component or surface. This enables quick view of the system as it is being setup.



*Created with the Personal Edition of HelpNDoc:* [*Free EPub producer*](http://www.helpndoc.com/create-epub-ebooks)

Status Bar

**Status Bar**

* + It is at the bottom of the window and used to display important information of the current optical system.

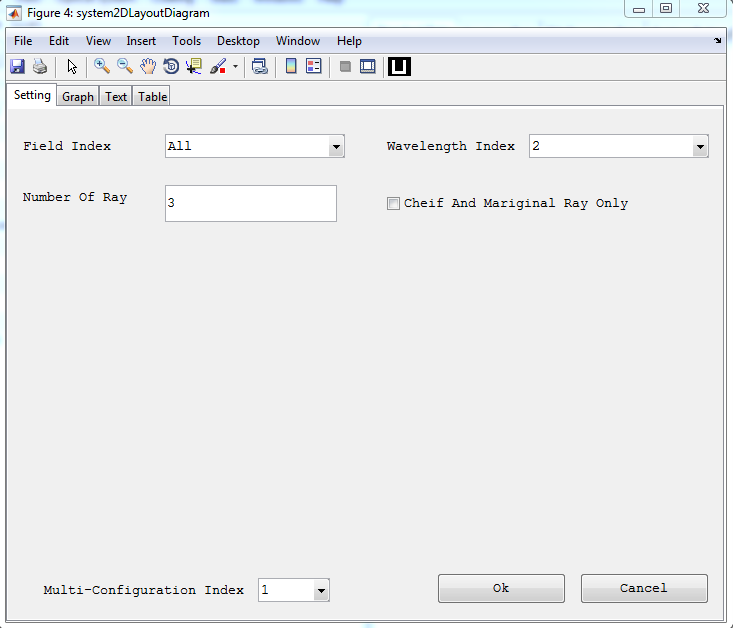


*Created with the Personal Edition of HelpNDoc:* [*Write EPub books for the iPad*](http://www.helpndoc.com/create-epub-ebooks)

**Children Windows**

**Children Windows**

Children windows are those windows which are displayed on request and are used to perform some analysis on the optical system or other optical components. All the children windows are made to have the same format in order to keep consistency throughout the toolbox. All windows have got four tabs as shown below.



* **Setting Tab:** 
  + Contains all the input settings for the analysis to be performed.
  + Almost all analysis windows have setting parameters.
* **Graph Tab:**
  + Displays the graphical results for analysis having graphical output.
  + Most analysis have graphical results.
* **Text Tab:** 
  + Displays the Text results for analysis having Text output.
  + Some analysis features such as single ray trace have text results.
* **Table Tab:**
  + Displays the tabular results for analysis having tabular output.
  + Used for displaying the x and y axis numerical values for 2D graphical results.

*Created with the Personal Edition of HelpNDoc:* [*Easily create PDF Help documents*](http://www.helpndoc.com/feature-tour)

**System Configuration Window**

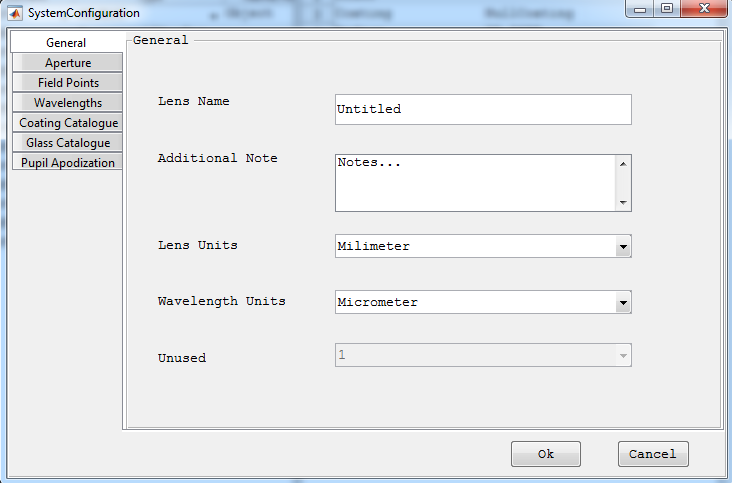
**System Configuration Window**

It contains tabbed windows for entering and editing the system configuration data of an optical system.

In the following section each tabbed windows will be briefly discussed.

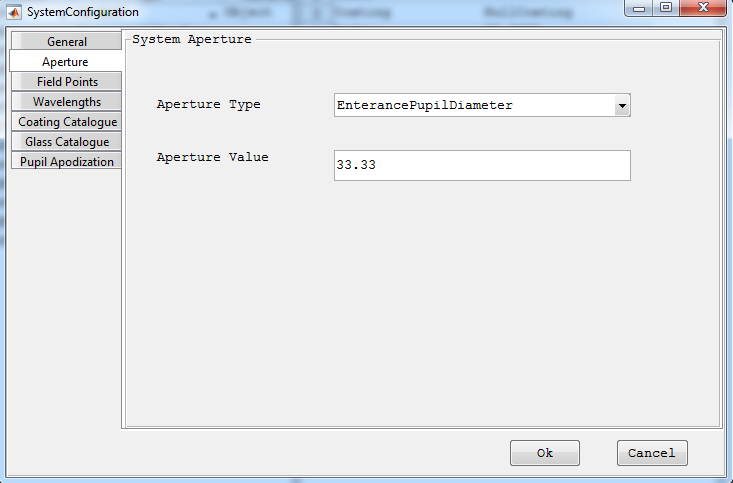
**1. General**

Here the general information relating the current optical system can be entered or modified including the units used for the lens and wavelength measurements.



**2. Aperture**

Define system aperture. System aperture can be specified by either Entrance Pupil Diameter or Object Space NA. All other methods of system aperture specifications are left for the future. For objects at finite distance both can be used whereas for infinite objects, the object space numerical aperture is not defined and so it cannot be set in the aperture window.



**3. Field Points**

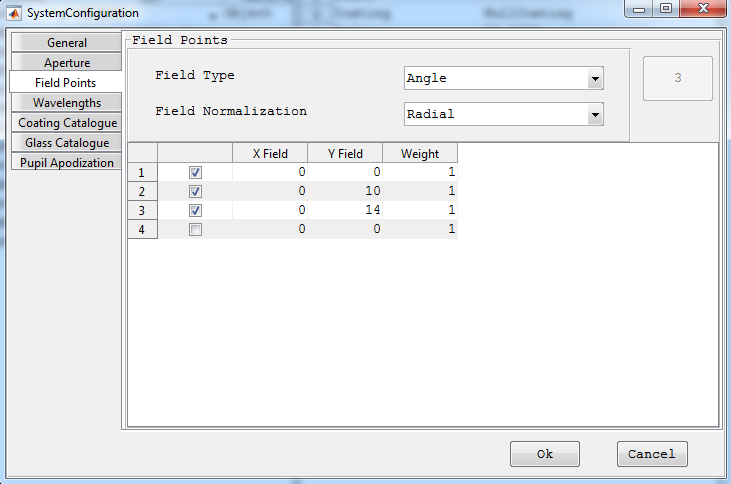
To define set of field points, which can later be used to analyze the system. Field points can be defined either by specifying the object height or by ray angles. If object heights are used to define the field points, the heights are measured in lens units. Field angles are always in degrees. The angles are measured with respect to the object space z axis and the paraxial entrance pupil position on the object space z axis. Like in Zemax, x field angles and y field angles can be converted to ray direction cosines using the following formulas:

tan (Fx) = l/n

tan (Fy) = m/n

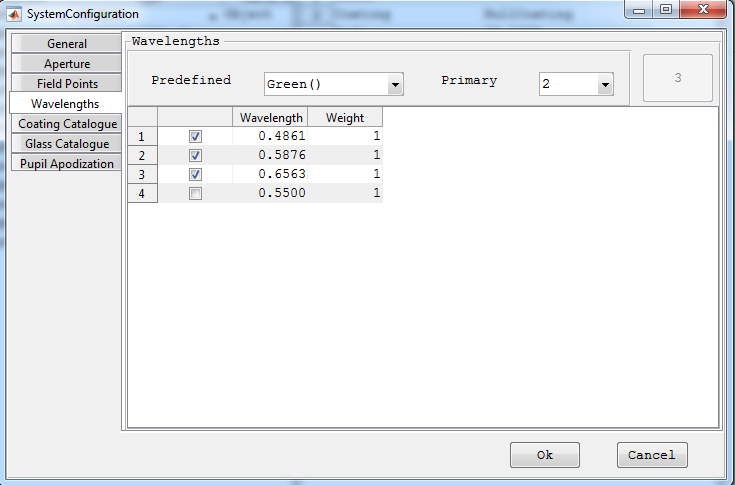
l2+m2+n2 = 1

Where l,m and n are the x,y and z direction cosines.



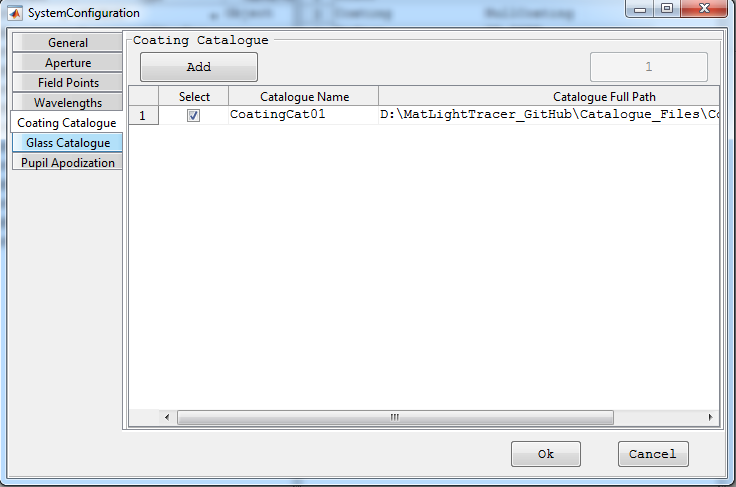
**4. Wavelengths**

To define wavelengths which the system can use for different analysis.The weight related to each wavelength values are not functional for current version. And the units of the wavelength specified are in the general tab. Predefined wavelengths can also be selected directly.



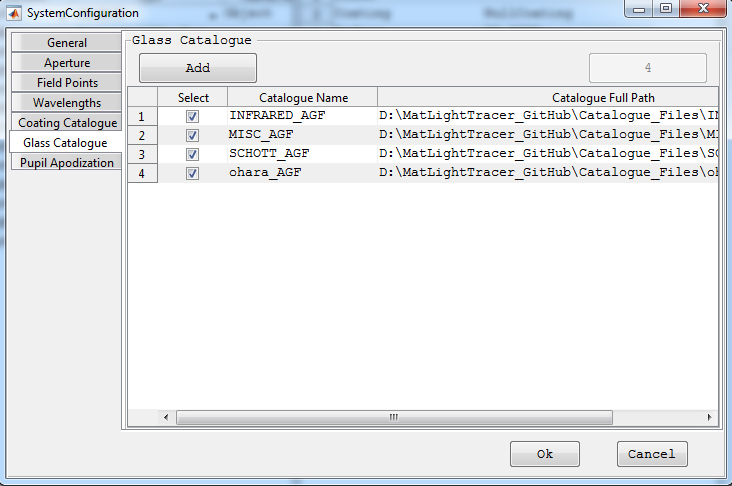
**5. Coating Catalogue**

List of all coating cataloogue used in the optical system. They can be deselected to remove from the used catalogue list. When optical system is saved those only selected ones will be saved as used coating catalogues. A new catalogue can also be added to the list. If the new catalogue added is not in the default catalogue files folder of the toolbox it will be automatically copied to that folder. Only those selected catalogues are cosidered during caoting analysis.



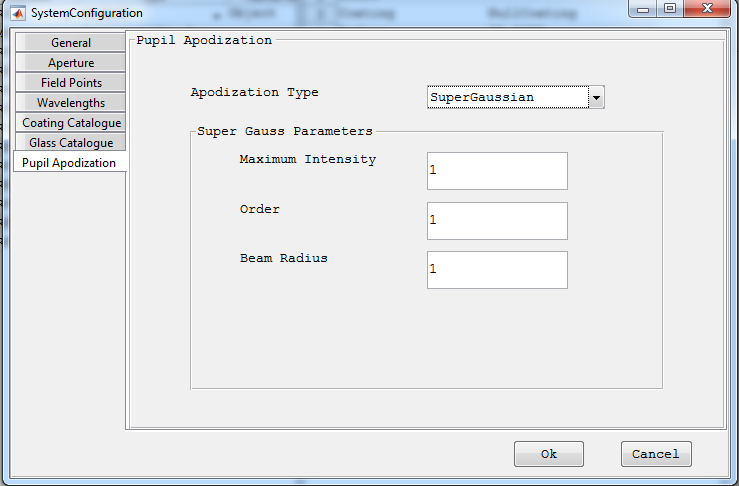
**6. Glass Catalogue**

List of all glass cataloogue used in the optical system. They can be deselected to remove from the used catalogue list. When optical system is saved those only selected ones will be saved as used glass catalogues. A new catalogue can also be added to the list. If the new catalogue added is not in the default catalogue files folder of the toolbox it will be automatically copied to that folder. Only those selected catalogues are cosidered during glass analysis.



**7. Pupil Apodization**

It is used to define the pupil apodization to be used in the system to simulate the effect of non-uniform illumination. Currently only uniform and super-gaussian profiles are supported.



*Created with the Personal Edition of HelpNDoc:* [*Easy to use tool to create HTML Help files and Help web sites*](http://www.helpndoc.com/help-authoring-tool)

**Setting up a New Optical System**

**Setting up a New Optical System**

To set up a new optical system system using the GUI of the toolbox,

1. Start the main parent GUI (see section "Starting the Toolbox" for detail steps.)
2. Start entering the element data in the element data editor of the parent GUI.
   * To add a new element:
     + Select the first column of the row where a new element is to be added.
     + Click the plus (+) button.
     + **Note:** Element can not be added in the row 1 (reserved for Object).
   * To remove an element:
     + Select the first column of the surface which is to be deleted.
     + Click the minus (-) button.
     + **Note:** First and Last elements can not be removed.
   * To set an element as Stop:
     + Select the element index in the stop comp popup menu.
3. Enter all system configuration data otherwise the default values will be used.
4. Finally Save the system.

*Created with the Personal Edition of HelpNDoc:* [*Full-featured Documentation generator*](http://www.helpndoc.com)

**Opening Optical Systems from File**

**Opening Optical Systems from File**

The toolbox supports saving an optical systems to file which can be opened when required. To open an optical system:

1. Click on File >> Open
2. Choose a valid optical system file (.mat file) from the open dialog box.
3. Click Ok then the system will be loaded to the parent GUI.

*Created with the Personal Edition of HelpNDoc:* [*Generate EPub eBooks with ease*](http://www.helpndoc.com/create-epub-ebooks)

**System Layout**

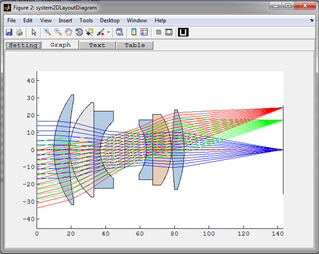
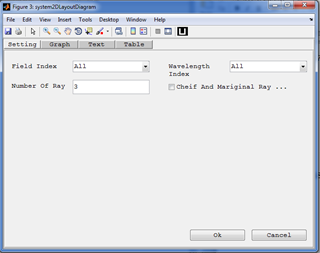
**System Layout Windows**

Display the 2D and 3D system layout windows for the given optical system.

*Created with the Personal Edition of HelpNDoc:* [*Free help authoring environment*](http://www.helpndoc.com/help-authoring-tool)

2D System Layout Window

**2D System Layout Window**



**Purpose:**

Display 2D layout diagram of the optical system. It shows the y-z cross section of the system.

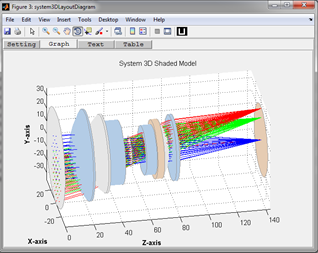
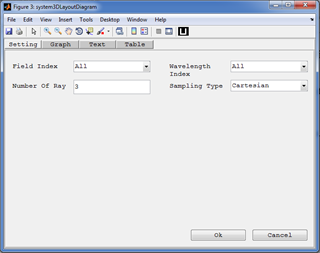
**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Field Index | Index of the field value for the rays to be displayed in the layout. | All fields can be used at the same time. |
| Wavelength Index | Index of the wavelength value for the rays to be displayed in the layout. | All wavelengths can be used at the same time. |
| Number of Rays | The number of rays to be traced per each field point and each wavelength. | Number less than 3 is not recommended. |
| Chief and Marginal Ray | Limits the number of rays to just 3 corresponding to the two marginal rays and a chief ray. |  |

*Created with the Personal Edition of HelpNDoc:* [*Free help authoring tool*](http://www.helpndoc.com/help-authoring-tool)

3D System Layout Window

**3D System Layout Window**



**Purpose:**

Display 3D layout diagram of the optical system. It shows the y-z cross section of the system.

**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Field Index | Index of the field value for the rays to be displayed in the layout. | All fields can be used at the same time. |
| Wavelength Index | Index of the wavelength value for the rays to be displayed in the layout. | All wavelengths can be used at the same time. |
| Number of Rays | The number of rays to be traced per each field point and each wavelength. | Number less than 3 is not recommended. |
| Sampling Type | Select the sampling types to be used for pupil. | Currently supported: Cartesian, Tangential, Sagital, and Random sampling. All others are treated as Cartesian. |

*Created with the Personal Edition of HelpNDoc:* [*Full-featured multi-format Help generator*](http://www.helpndoc.com/help-authoring-tool)

**Optical System Analysis**

**Optical System Analysis**

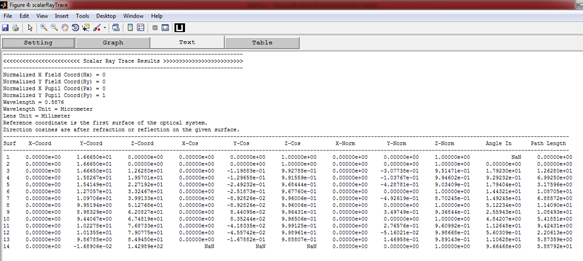
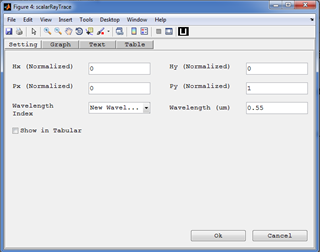
Here all analysis features of the toolbox are described.

*Created with the Personal Edition of HelpNDoc:* [*Full-featured Documentation generator*](http://www.helpndoc.com)

Scalar Ray Trace

**Scalar Ray Trace**

It is feature of the toolbox which allows tracing of a single ray through the system with or without polarization consideration. A single ray in an optical system can be defined using the relative definition specifying the normalized coordinates/angles of the field point and normalized coordinate of the pupil point where the ray crosses the entrance pupil.



**Purpose:**

To see the real scalar ray trace results for single ray.

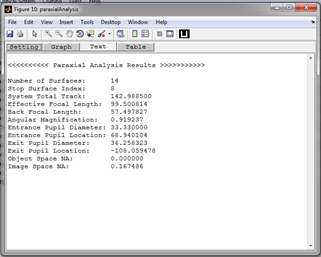
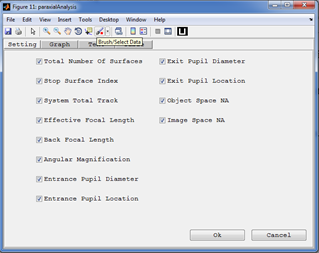
**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Hx(Normalized) ,Hy(Normalized) | Normalized values for field coordinates/angles in the object space. | Valid values [-1,1] |
| Px(Normalized) ,Py(Normalized) | Normalized values for pupil coordinates at the entrance pupil. | Valid values [-1,1] |
| Wavelength Index | Index of the wavelength value for the ray. | A single index should be selected. But new wavelengths can also be used. |
| Wavelength | The wavelength to be used for the ray trace in micrometer. | If "New Wavelength" is selected for a wavelength index, the new wavelength should be specified here in micrometer. |
| Show in Tabular | Display the result in tabular format as in Zemax. | If deselected, the ray trace results will be displayed in a text format giving the results in surface to surface sequence. |

*Created with the Personal Edition of HelpNDoc:* [*Free help authoring tool*](http://www.helpndoc.com/help-authoring-tool)

Paraxial Analysis

**Paraxial Analysis**



**Purpose:**

Performs paraxial analysis and displays the result in the command window.

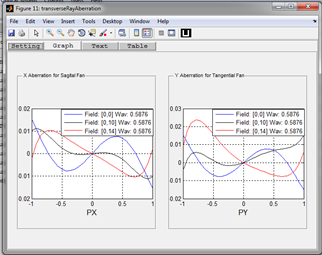
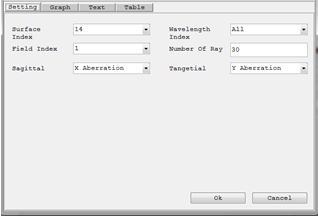
**Setting:**

Select the paraxial/gaussian parameter of the optical system to compute.

*Created with the Personal Edition of HelpNDoc:* [*Easily create PDF Help documents*](http://www.helpndoc.com/feature-tour)

Transverse Ray Aberration

**Polarization Ellipse Map**



**Purpose:**

Display graphs showing the transverse aberration (in X or Y) of sagittal or tangential rays with respect to the cheif ray at a given surface.

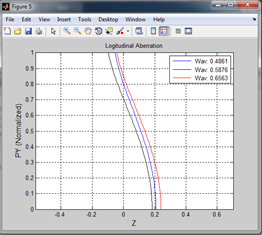
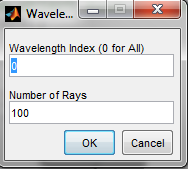
**Setting:**

|  |  |
| --- | --- |
| **Settings** | **Description** |
| Surface Index | The surface index used to compute the transverse aberration. |
| Field Index | Index of the field value for the analysis. |
| Wavelength Index | Index of the wavelength value for the analysis. |
| Number of Rays | The number of rays to be traced per each field point and each wavelength. |
| Sagittal | Selects which aberration component to plot for the sagittal fan. |
| Tangential | Selects which aberration component to plot for the tangential fan. |

*Created with the Personal Edition of HelpNDoc:* [*Full-featured EBook editor*](http://www.helpndoc.com/create-epub-ebooks)

Longitudinal Aberration

**Longitudinal Aberration Diagram**



**Purpose:**

An analysis window (added as an external extension to the toolbox - As an example) which displays the longitudinal aberration as a function of pupil height at each wavelength.

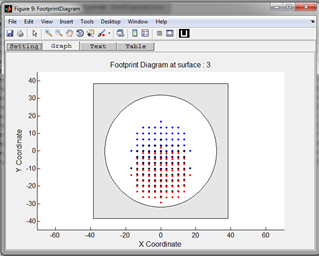
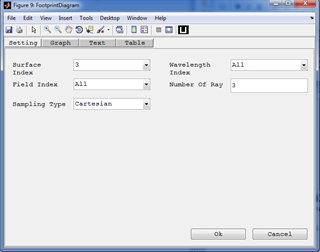
**Setting:**

|  |  |
| --- | --- |
| **Settings** | **Description** |
| Wavelength Index | Index of the wavelength value for the analysis. |
| Number of Rays | The number of rays to be traced per each field point and each wavelength. |

*Created with the Personal Edition of HelpNDoc:* [*Free help authoring environment*](http://www.helpndoc.com/help-authoring-tool)

Footprint Diagram

**Footprint Diagram**



**Purpose:**

Display the footprint diagram of the optical system at a given surface.

**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Surface Index | The surface index used to view the footprint diagram. |  |
| Field Index | Index of the field value for the rays to be displayed in the layout. | All fields can be used at the same time. |
| Wavelength Index | Index of the wavelength value for the rays to be displayed in the layout. | All wavelengths can be used at the same time. |
| Number of Rays | The number of rays to be traced per each field point and each wavelength. | Number less than 3 is not recommended. |
| Sampling Type | Select the sampling types to be used for pupil. | Currently supported: Cartesian, Tangential, Sagital, and Random sampling. All others are treated as Cartesian. |

*Created with the Personal Edition of HelpNDoc:* [*Full-featured Documentation generator*](http://www.helpndoc.com)

Hybrid Diffraction

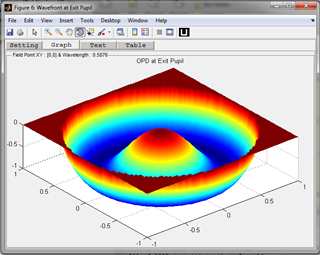
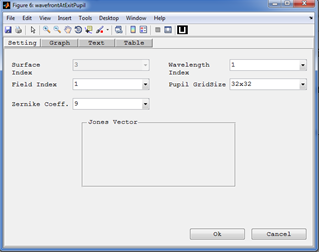
**Hybrid Diffraction**

To account for diffraction in system aperture a hybrid diffraction model is implemented in the toolbox. That is ray is traced until exit pupil and then the complex field is reconstructed at the exit pupil. The complex field is then propagated from the exit pupil to the focal plane using simple diffraction algorithm (Far field diffraction integral).

*Created with the Personal Edition of HelpNDoc:* [*Easily create EBooks*](http://www.helpndoc.com/feature-tour)

Wavefront @ Exit Pupil

**Wavefront @ Exit Pupil**



**Purpose:**

Display the wavefront error surface at the exit pupil. It corresponds to the optical path difference (OPD) all rays with respect to the chief ray at the exit pupil.

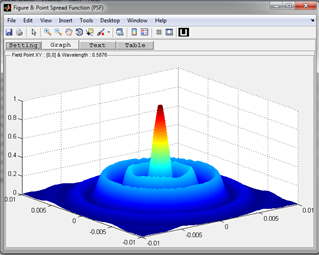
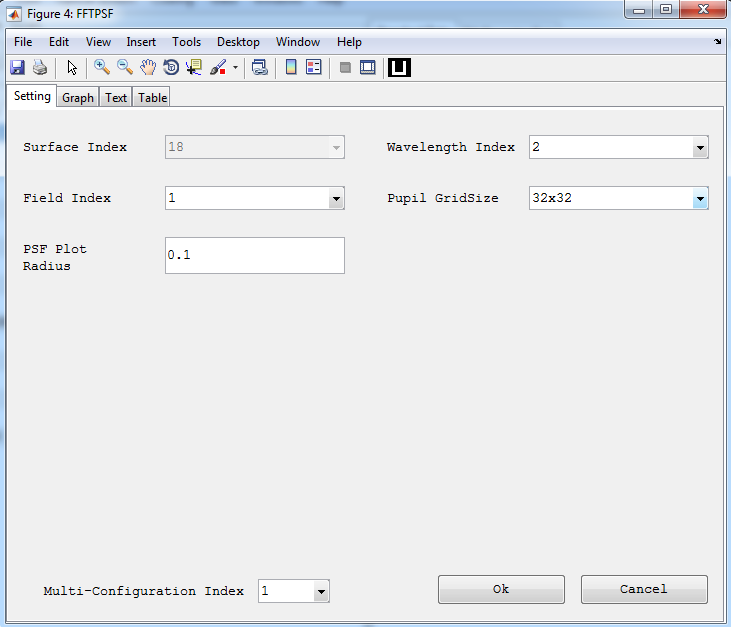
**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Surface Index | The surface index used to compute the OPD. | Currently it is not functional and set fixed to the image surface. |
| Field Index | Index of the field value for the rays to be displayed in the layout. | A single index should be selected. |
| Wavelength Index | Index of the wavelength value for the rays to be displayed in the layout. | A single index should be selected. |
| Zernike coefficients. | The number of Zernike polynomials used for surface fitting. | Maximum of 64 coefficients can be computed. |
| Pupil Grid size | The sampling grid size of the entrance pupil. | Pupil sampling type is Cartesian. |

*Created with the Personal Edition of HelpNDoc:* [*Free Web Help generator*](http://www.helpndoc.com)

FFT PSF

**FFT PSF**



**Purpose:**

Display the point spread function of the system. It is based on the chirp Z-transform FFT algorithm.

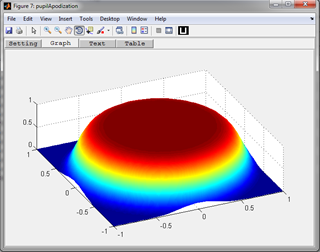
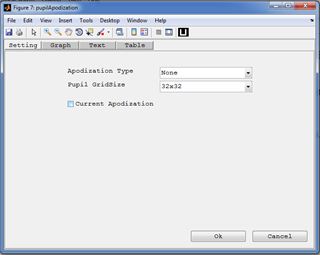
**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Surface Index | The surface index used to compute the OPD. | Currently it is not functional and set fixed to the image surface. |
| Field Index | Index of the field value for the rays to be displayed in the layout. | A single index should be selected. |
| Wavelength Index | Index of the wavelength value for the rays to be displayed in the layout. | A single index should be selected. |
| PSF plot radius. | Specify the radius required for the PSF plot. Changes the sampling distance in the focal plane. | CZT FFT allows the decoupling of the sampling in the pupil plane from that in the focal plane. |
| Pupil Grid Size | The sampling grid size of the entrance pupil. |  |

*Created with the Personal Edition of HelpNDoc:* [*Single source CHM, PDF, DOC and HTML Help creation*](http://www.helpndoc.com/help-authoring-tool)

Pupil Apodization

**Pupil Apodization**



**Purpose:**

Display the pupil apodization profile used for the system. It can also be used to show any profiles possible in the toolbox.

**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Apodization Type | Select the type of apodization profile to view. Different parameter entry windows will be displayed depending on the type selected. | Currently only uniform and super gaussian profile is supported. |
| Pupil Grid Size | The sampling grid size of the entrance pupil. | Pupil sampling type is Cartesian. |
| Current Apodization | Fixes the apodization type and parameters to the one used by the current optical system. |  |

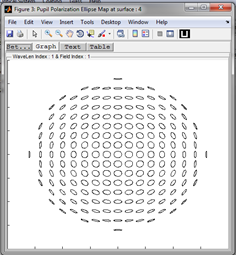
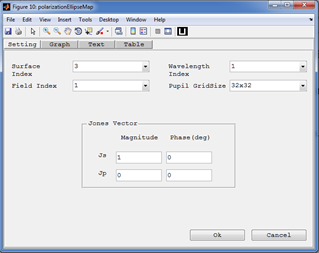
*Created with the Personal Edition of HelpNDoc:* [*Easily create EPub books*](http://www.helpndoc.com/feature-tour)

Polarization

*Created with the Personal Edition of HelpNDoc:* [*Create HTML Help, DOC, PDF and print manuals from 1 single source*](http://www.helpndoc.com/help-authoring-tool)

Polarization Ellipse Map

**Polarization Ellipse Map**



**Purpose:**

Display the pupil polarization map of the system for a given initial polarization over the pupil at a given surface. It just displays the map in the x-y plane considering the Ex and Ey.

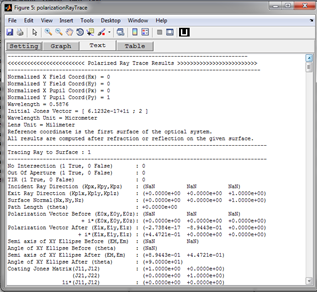
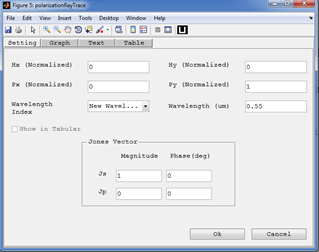
**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Surface Index | The surface index used to view the footprint diagram. |  |
| Field Index | Index of the field value for the rays to be displayed in the layout. | A single index should be selected. |
| Wavelength Index | Index of the wavelength value for the rays to be displayed in the layout. | A single index should be selected. |
| Pupil Grid Size | The sampling grid size of the entrance pupil. |  |
| Jones Vector | Defines the initial polarization vector of the ray. It is given in s-p-k coordinate system which is not the same as the global x-y-z coordinate in many cases. | The phase values of Jones vector are always given in degrees. |

*Created with the Personal Edition of HelpNDoc:* [*Write eBooks for the Kindle*](http://www.helpndoc.com/feature-tour/create-ebooks-for-amazon-kindle)

Polarization Ray Trace

**Polarization Ray Trace**



**Purpose:**

To see the real polarization ray trace results for single ray.

**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Hx(Normalized) ,Hy(Normalized) | Normalized values for field coordinates/angles in the object space. | Valid values [-1,1] |
| Px(Normalized) ,Py(Normalized) | Normalized values for pupil coordinates at the entrance pupil. | Valid values [-1,1] |
| Wavelength Index | Index of the wavelength value for the ray. | A single index should be selected. But new wavelengths can also be used. |
| Wavelength | The wavelength to be used for the ray trace in micrometer. | If "New Wavelength" is selected for a wavelength index, the new wavelength should be specified here in micrometer. |
| Jones Vector | Defines the initial polarization vector of the ray. It is given in s-p-k coordinate system which is not the same as the global x-y-z coordinate in many cases. | The phase values of Jones vector are always given in degrees. |

*Created with the Personal Edition of HelpNDoc:* [*Easily create EBooks*](http://www.helpndoc.com/feature-tour)

Polarization Aberration:

**Polarization Aberration**

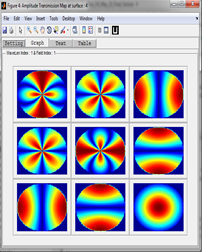
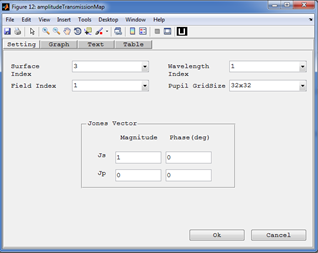
Polarization aberration function represent variation of polarization ray tracing matrix of the system (which represents the polarization property of the system) over the pupil area. It can be used to analyze the polarization property of an optical system by plotting polarization dependent properties of the system over the pupil area. The following plots are used to analyze that. They are all derived from the polarization aberration function:

* **Amplitude Transmission Map:**
* **Phase Map:**
* **Wavefront Diattenuation Map:**
* **Wavefront Retardation Map:**

*Created with the Personal Edition of HelpNDoc:* [*Easily create HTML Help documents*](http://www.helpndoc.com/feature-tour)

*Amplitude Transmission Map*

**Amplitude Transmission Map**



**Purpose:**

Plot of the amplitude part of the total polarization ray tracing matrix at different pupil locations. It is a 3X3 matrix of color plots.

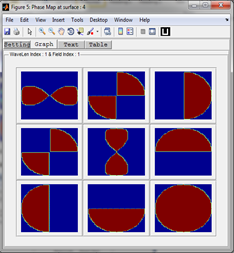
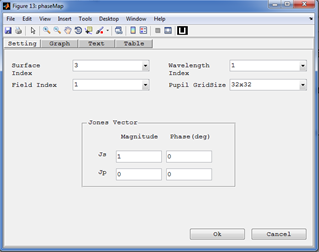
**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Surface Index | The surface index used to view the footprint diagram. |  |
| Field Index | Index of the field value for the rays to be displayed in the layout. | A single index should be selected. |
| Wavelength Index | Index of the wavelength value for the rays to be displayed in the layout. | A single index should be selected. |
| Pupil Grid Size | The sampling grid size of the entrance pupil. |  |
| Jones Vector | Defines the initial polarization vector of the ray. It is given in s-p-k coordinate system which is not the same as the global x-y-z coordinate in many cases. | The phase values of Jones vector are always given in degrees. |

*Created with the Personal Edition of HelpNDoc:* [*Generate EPub eBooks with ease*](http://www.helpndoc.com/create-epub-ebooks)

*Phase Map*

**Phase Map**



**Purpose:**

Plot of the phase part of the total polarization ray tracing matrix at different pupil locations. It can show the phase variation in range of one wave limit. It is a 3X3 matrix of color plots and it has setting parameters similar to amplitude transmission map.

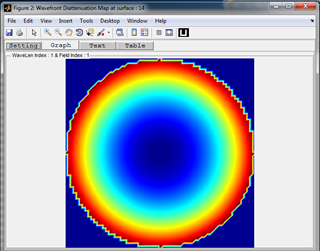
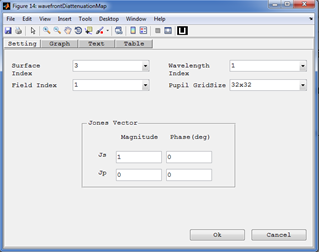
**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Surface Index | The surface index used to view the footprint diagram. |  |
| Field Index | Index of the field value for the rays to be displayed in the layout. | A single index should be selected. |
| Wavelength Index | Index of the wavelength value for the rays to be displayed in the layout. | A single index should be selected. |
| Pupil Grid Size | The sampling grid size of the entrance pupil. |  |
| Jones Vector | Defines the initial polarization vector of the ray. It is given in s-p-k coordinate system which is not the same as the global x-y-z coordinate in many cases. | The phase values of Jones vector are always given in degrees. |

*Created with the Personal Edition of HelpNDoc:* [*Easily create Help documents*](http://www.helpndoc.com/feature-tour)

*Wavefront Diattenuation Map*

**Wavefront Diattenuation Map**



**Purpose:**

Plot of the diattenuation related to the total polarization ray tracing matrix at different pupil locations. It is a polarization dependent apodization map. It is a single color map showing diattenuation related to the total polarization ray tracing matrix over the pupil coordinates.

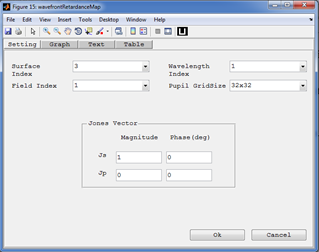
**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Surface Index | The surface index used to view the footprint diagram. |  |
| Field Index | Index of the field value for the rays to be displayed in the layout. | A single index should be selected. |
| Wavelength Index | Index of the wavelength value for the rays to be displayed in the layout. | A single index should be selected. |
| Pupil Grid Size | The sampling grid size of the entrance pupil. |  |
| Jones Vector | Defines the initial polarization vector of the ray. It is given in s-p-k coordinate system which is not the same as the global x-y-z coordinate in many cases. | The phase values of Jones vector are always given in degrees. |

*Created with the Personal Edition of HelpNDoc:* [*Free iPhone documentation generator*](http://www.helpndoc.com/feature-tour/iphone-website-generation)

*Wavefront Retardation Map*

**Wavefront Retardation Map**



**Purpose:**

Plot of the retardance related to the total polarization ray tracing matrix at different pupil locations. . It is a polarization dependent optical path difference (OPD) map. It is a single color map showing retardance related to the total polarization ray tracing matrix over the pupil coordinates.

**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Surface Index | The surface index used to view the footprint diagram. |  |
| Field Index | Index of the field value for the rays to be displayed in the layout. | A single index should be selected. |
| Wavelength Index | Index of the wavelength value for the rays to be displayed in the layout. | A single index should be selected. |
| Pupil Grid Size | The sampling grid size of the entrance pupil. |  |
| Jones Vector | Defines the initial polarization vector of the ray. It is given in s-p-k coordinate system which is not the same as the global x-y-z coordinate in many cases. | The phase values of Jones vector are always given in degrees. |

*Created with the Personal Edition of HelpNDoc:* [*Easy CHM and documentation editor*](http://www.helpndoc.com)

Pulse Analysis

**Pulse Analysis**

(Yet to be written , Sorry :( )

*Created with the Personal Edition of HelpNDoc:* [*Full-featured EPub generator*](http://www.helpndoc.com/create-epub-ebooks)

**Coating**

**Coating**

In the toolbox the coating parameter can be:

* **None:** The coating is just ignored and the corresponding jones matrix would be a 2x2 unity.
* **Bare Glass:** The Jones matrix is computed from the Fresnels coefficients for bare glass or a single interface would be used.
* **A multilayer coating:**  The Jones matrix is computed from the Fresnels coefficients for multilayer interface would be used.

All coating will be stored in coating catalogues. The coating and coating catalogue can be accessed, edited, and deleted. New coating and coating catalogues can also be defined by the user.

*Created with the Personal Edition of HelpNDoc:* [*Easy CHM and documentation editor*](http://www.helpndoc.com)

New Coating Catalogue

**New Coating Catalogue**



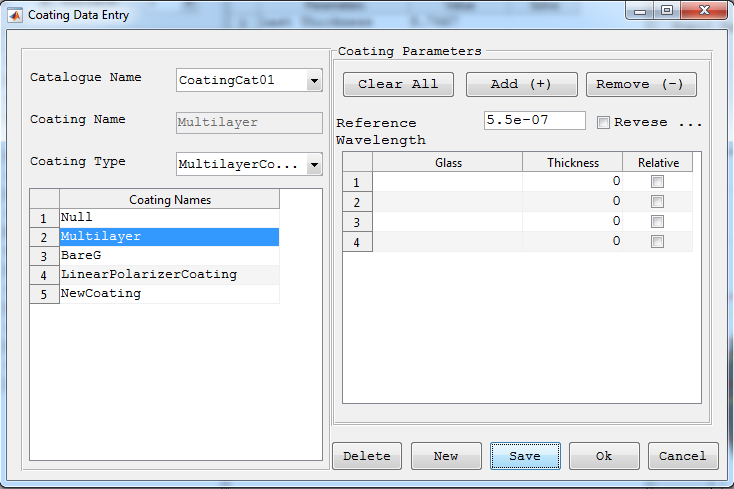
**Purpose:**

To create and add a new coating catalogue to the default '...\CatalogueFiles' folder of the toolbox.

*Created with the Personal Edition of HelpNDoc:* [*Create iPhone web-based documentation*](http://www.helpndoc.com/feature-tour/iphone-website-generation)

Coating Data Editor

**Coating Data Editor**



**Purpose:**

A window used to add new, edit and delete existing coating in the coating catalogues used by the optical system.

**Windows Components**

* **Catalogue Name Popup:** Select the coating catalogue to look for coatings.
* **Coating Name Text Box:** Display name of the coating selected. Or it can also be used to enter a coating name of the new coating to be added.
* **Coating Type Popup:** Display type of the coating selected. Or it can also be used to enter a coating type of the new coating to be added.
* **Caoting List Table:** Display all coating in the current catalogue.
* **Coating Parameters:** Used to view, and edit parameters of the selected coating. This section varies depending on the type of coating selected.
* **Delete:** Deletes the currently selected coating from the catalogue.
* **New:** To add a new coating.
* **Save:** Save the coating data entered to current catalogue. This buttons saves the current coating in to file, disables all coating parameters fields and then toggles to 'Edit' after the coating is saved.

*Created with the Personal Edition of HelpNDoc:* [*Free help authoring tool*](http://www.helpndoc.com/help-authoring-tool)

Coating Analysis

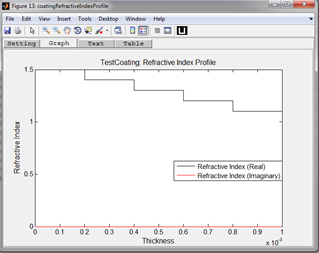
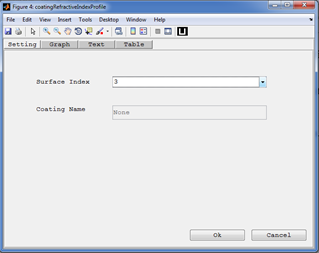
**Coating Analysis**

The toolbox supports analysis of multilayer coating. Here coating properties such as transmission and reflection coefficients vs incidence angle and wavelength are included. Coating used in the optical system can be taken by selecting the surface index or a new coating can be analyzed by entering its name directly.

*Created with the Personal Edition of HelpNDoc:* [*Free EPub producer*](http://www.helpndoc.com/create-epub-ebooks)

Refractive Index Profile

**Refractive Index Profile**



**Purpose:**

Displays the refractive index of a given multilayer coating. The real and complex refractive indices are drawn in different colors in the same graph.

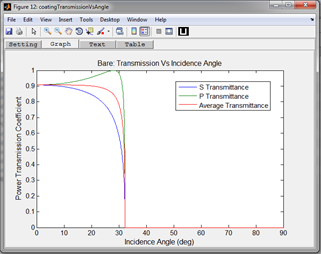
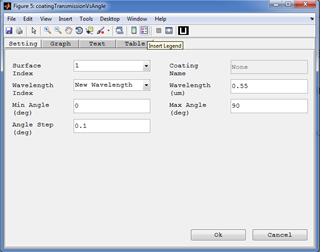
**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Surface Index | The surface index to take and analyze the coating. |  |
| Coating Name | Name of the selected coating. | The coating catalogue containing the coating should be included in the catalogue list of the system. |

*Created with the Personal Edition of HelpNDoc:* [*Write EPub books for the iPad*](http://www.helpndoc.com/create-epub-ebooks)

Coating Property Versus Incidence Angle

**Coating Properties vs Incidence Angle**



**Purpose:**

* + - **Reflection vs Angle:**

Computes and plots the S, P, and average polarization intensity coefficients for reflection for the specified surface with its coating as a function of incident angle.

* + - **Transmission vs Angle:**

Computes and plots the S, P, and average polarization intensity coefficients for transmission for the specified surface with its coating as a function of incident angle.

* + - **Diattenuation vs Angle:**

Computes and plots the R (reflected) and T (transmitted) diattenuation for the specified surface as a function of incident angle.

* + - **Retardance vs Angle:**

Computes and plots the retardance for the specified surface as a function of incident angle.

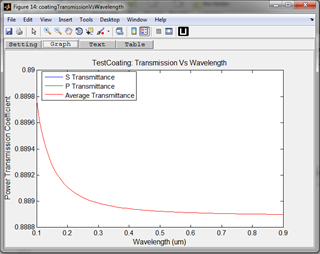
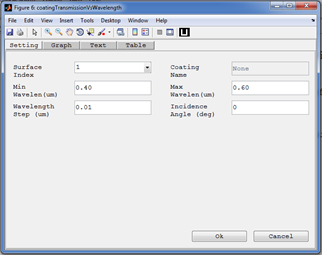
**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Surface Index | The surface index to take and analyze the coating. |  |
| Coating Name: | Text box to enter a coating name to be analyzed. | If the New Coating is selected for Surface Index then the user is allowed to enter the coating name. |
| Minimum Angle: | The minimum angle of incidence to plot in degree. This defines the left edge of the plot. |  |
| Maximum Angle: | The maximum angle of incidence to plot in degree. This defines the right edge of the plot. |  |
| Angle Step: | The angle step in degrees for plotting the graphs. |  |
| Wavelength Index: | Index of wavelength to be used from those inserted in system configuration window. |  |
| Wavelength: | Text box to enter a wavelength not defined in the system configuration window. |  |

*Created with the Personal Edition of HelpNDoc:* [*Free EPub and documentation generator*](http://www.helpndoc.com)

Coating Properties vs Wavelength

**Coating Properties vs Wavelength**



**Purpose:**

* + - **Reflection vs Wavelength:**

Computes and plots the S, P, and average polarization intensity coefficients for reflection for the specified surface with its coating as a function of Wavelength for a given incidence angle.

* + - **Transmission vs Wavelength:**

Computes and plots the S, P, and average polarization intensity coefficients for transmission for the specified surface with its coating as a function of wavelength for a given incidence angle.

* + - **Diattenuation vs Wavelength:**

Computes and plots the R (reflected) and T (transmitted) diattenuation for the specified surface as a function of Wavelength for a given incidence angle.

* + - **Retardance vs Wavelength:**

Computes and plots the retardance for the specified surface as a function of Wavelength for a given incidence angle.

**Setting:**

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Surface Index | The surface index to take and analyze the coating. |  |
| Coating Name: | Text box to enter a coating name to be analyzed. | If the New Coating is selected for Surface Index then the user is allowed to enter the coating name. |
| Minimum Wavelength: | The minimum wavelength (in nm) . This defines the left edge of the plot. |  |
| Maximum Wavelength: | The maximum wavelength (in nm) . This defines the right edge of the plot. |  |
| Wavelength Step: | The wavelength (in nm) step for plotting the graphs. |  |
| Incident Angle: | Text box to enter an incident angle (in degrees) |  |

*Created with the Personal Edition of HelpNDoc:* [*Free Web Help generator*](http://www.helpndoc.com)

**Glass**

**Glass**

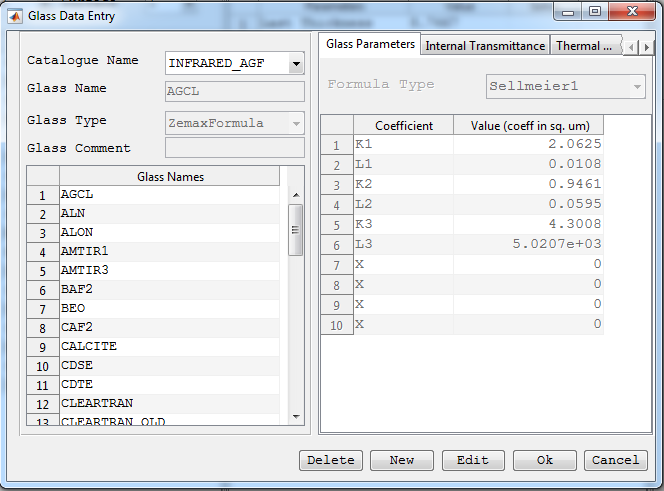
In the toolbox the glass parameter could be:

* **FixedIndex:** Defined fixed refractive index, abbe number and partial dispersion.
* **Sellmeir1:** Defined by the six sellmeir coefficients.

*Created with the Personal Edition of HelpNDoc:* [*Easy to use tool to create HTML Help files and Help web sites*](http://www.helpndoc.com/help-authoring-tool)

Glass Data Editor

**Glass Data Editor**



**Purpose:**

A window used to add new, edit and delete existing glass in the glass catalogues used by the optical system.

**Windows Components**

* **Catalogue Name Popup:** Select the glass catalogue to look for glass.
* **Glass Name Text Box:** Display name of the glass selected. Or it can also be used to enter a glass name of the new glass to be added.
* **Glass Type Popup:** Display type of the glass selected. Or it can also be used to enter a glass type of the new glass to be added.
* **Glass List Table:** Display all glass in the current catalogue.
* **Glass Parameters Panel:** Used to view and edit the parameters of the glass selected. its content varies depending on the type of glass selected.
* **Delete:** Deletes the currently selected glass from the catalogue.
* **New:** To add a new glass.
* **Save:** Save the glass data entered to current catalogue. This buttons saves the current glass in to file, disables all glass parameters fields and then toggles to 'Edit' after the glass is saved.

*Created with the Personal Edition of HelpNDoc:* [*Easily create CHM Help documents*](http://www.helpndoc.com/feature-tour)

New Glass Catalogue

**New Glass Catalogue**



**Purpose:**

To create and add a new glass catalogue to the default \CatalogueFiles folder of the toolbox.

*Created with the Personal Edition of HelpNDoc:* [*Produce Kindle eBooks easily*](http://www.helpndoc.com/feature-tour/create-ebooks-for-amazon-kindle)

**Help**

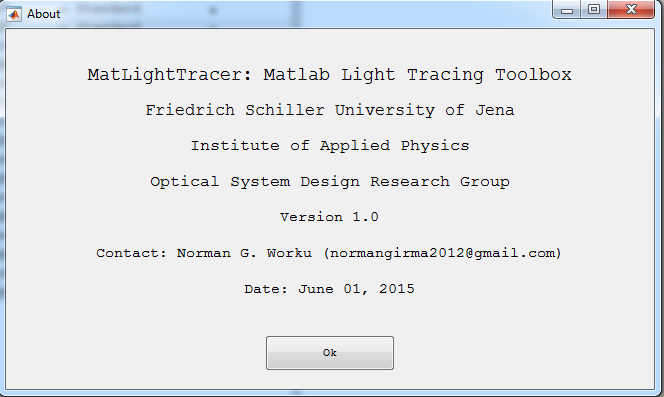
**Help**

The toolbox is well documented and has user manual and programming reference. User manual gives walkthrough of the GUI based usage of the toolbox. The programming reference organized all the detailed commented codes of the toolbox in indexed and interlinked html file format.

*Created with the Personal Edition of HelpNDoc:* [*Free EBook and documentation generator*](http://www.helpndoc.com)

About

**About**



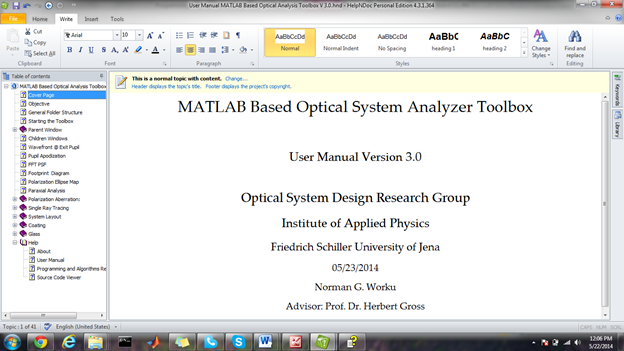
**Purpose:**

Short description about the toolbox.

*Created with the Personal Edition of HelpNDoc:* [*Create HTML Help, DOC, PDF and print manuals from 1 single source*](http://www.helpndoc.com/help-authoring-tool)

User Manual

**User Manual**

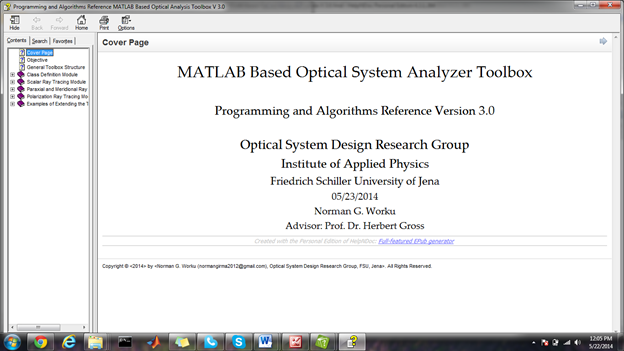


**Purpose:**

Open the electronic form of the user manual.

*Created with the Personal Edition of HelpNDoc:* [*Write EPub books for the iPad*](http://www.helpndoc.com/create-epub-ebooks)

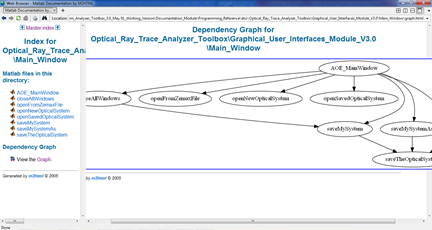
Programming and Algorithms Reference



*Created with the Personal Edition of HelpNDoc:* [*Easily create iPhone documentation*](http://www.helpndoc.com/feature-tour/iphone-website-generation)

Source Code Viewer

**Source Code Viewer**



**Purpose:**

Opens an html window which was generated by using a free matlab toolbox called M2HTML [<http://www.artefact.tk/software/matlab/m2html/>]. The html document generated enables users to view and navigate through all functions and class definitions of the toolbox. It also clearly shows function interdependencies in both graphical and hyperlinked text forms. This will have great importance for updating the toolbox as it enables the user to see which other functions are affected by the modification of a given function.

*Created with the Personal Edition of HelpNDoc:* [*Easy EPub and documentation editor*](http://www.helpndoc.com)

**Using MatLightTracer From Matlab Command**

# Using MatLightTracer From Matlab Command

Even though MatLightTracer toolbox has its own GUI which could be used to setup and analyze optical systems, it sometimes becomes neccessary to use the functionality of the toolbox from the matlab command window or an other external toolbox. This section of the user manual describes some important points which could be helpful when trying to use the MatLightTracer toolbox with out its GUI. This enables integration of the toolbox to other projects.

*Created with the Personal Edition of HelpNDoc:* [*Create iPhone web-based documentation*](http://www.helpndoc.com/feature-tour/iphone-website-generation)

**Loading an Optical System**

# Loading an Optical System

Performing any analysis of an optical system in the command window starts with loading a previously saved optical system in to current work space. To load a previously saved optical systems, the following function can be used

*newOpticalSystem = OpticalSystem(<The Optical System Full Path>)*

The function accepts the full file path of the optical system to be loaded. The toolbox can load optical system from two kinds of files

* + MatLightTracer File (\*.mat file) created by the toolbox
  + Zemax File (\*.zmx file) created by zemax (version 13 or latest)

Therefore if the user specifies the full path of a file in one of the above format, then the optical system will be loaded to the variable *newOpticalSystem.*

The *newOpticalSystem* is a data structure (object) which can be used to access the optical system parameters and passed to other functions (such as optical system analysis function) as input argument.

**Example**:

*doubleGaussFullFileName = 'D:\MatLightTracer\_GitHub\Sample\_Optical\_Systems\Double Gauss 28 degree field.mat';*

*optSystem = OpticalSystem(doubleGaussFullFileName);*

*Created with the Personal Edition of HelpNDoc:* [*Easy EPub and documentation editor*](http://www.helpndoc.com)

**Creating Initial Rays For Optical Systems**

# Creating Initial Ray Bundles for Optical System

To trace rays through an optical system, the ray bundle should be generated first. The MatLightTracer toolbox provides three different ways to define or create initial ray bundle for tracing through an optical system.

1. By defining the all parameters of your initial rays and putting them in ScalarRayBundle struct.

Example: To create two rays with starting from different positions and propagating along direction and having the same wavelength, the following lines of codes could be used

*rayPositions = [0, 0;*

*0.001, -0.001;*

*0, 0];*

*rayDirections = [0, 0;*

*0, 0;*

*1, 1];*

*rayWavelengths = [0.55\*10^-6];*

*initialRayBundle = ScalarRayBundle( rayPositions, rayDirections, rayWavelengths);*

2. By using the field index, wavelength index and the number of pupil points. For this you need to load optical system first. Then the following function from the toolbox can be used create initial ray bundle. It is recommended to use this method before tracing rays through an optical system.

*fieldIndices = 1;*

*wavelengthIndices = 1;*

*nRay1=31;*

*nRay2=31;*

*pupilSamplingType = 'Cartesian'; % Can also be 'Tangential','Sagital','Cross','Polar','Random'*

*initialRayBundle = getInitialRayBundle(optSystem, wavelengthIndices, fieldIndices, nRay1, nRay2, pupilSamplingType);*

All data in the *ScalarRayBundle* struct (such as Positions and Directions) are listed as an array of the corresponding parameters for each ray in the bundle.

But the parameters of the rays are arranged in such away that the following manner so that each ray from each field point and with a given wavelength can be easily addressed.

**Example:** The ScalarRayBundle.Position struct is arranged in the following manner



The RayBundle structure has a field called *FixedParameters* which is used to hold information about the total number of field points, wavelengths and pupil points used to create the rays in the RayBundle structure. The struct has the following fields

*FixedParameters.TotalNumberOfPupilPoints :* The number of pupil points

*FixedParameters.TotalNumberOfFieldPoints :* The number of field points

*FixedParameters.TotalNumberOfWavelengths :* The number of wavelengths

And the *FixedParameters* can be used in extracting the ray parameters specified by its field index, wavelength index and pupil index.

3. By using other functions to generate some specific rays for an optical system. Special rays of an optical system such as chief and marginal rays can be generated by special functions provided by the toolbox .

*% Special single ray generating functions*

*fieldIndex = 1;*

*wavelengthIndex = 1;*

*chiefRay = getChiefRay(optSystem, fieldIndex, wavelengthIndex);*

*mariginalRay = getMariginalRay(optSystem, fieldIndex, wavelengthIndex);*

*Created with the Personal Edition of HelpNDoc:* [*Free iPhone documentation generator*](http://www.helpndoc.com/feature-tour/iphone-website-generation)

**Tracing Rays Through Optical Systems**

**Tracing Multiple Rays**

To trace rays through optical systems, the following steps can be followed:

**Step 1:** Define your optical system using the GUI of the toolbox and save the system to .mat file. See the first part of the user manual on how to use the GUI to build an optical system. If you have already your optical system saved as \*.mat or \*.zmx file then you can skip this step.

**Step 2:** Load the optical system in to the Matlab command window by using the full file name (See section "Loading an Optical System").

*newOpticalSystem = OpticalSystem(<The Optical System Full Path>)*

**Step 3:** Define initial ray bundles (See section "Creating Initial Rays for Optical Systems"). In short the initial ray bundles can be created by three ways:

1. By defining the all parameters (positions, directions and wavelengths) of your initial rays and putting them in ScalarRayBundle struct.

*initialRayBundle = ScalarRayBundle( rayPositions,rayDirections,rayWavelengths);*

2. By using the field index, wavelength index and the number of pupil points for a given optical system.

*initialRayBundle = getInitialRayBundle(optSystem,wavelengthIndices, fieldIndices, nRay1,nRay2,pupilSamplingType);*

3. By using other functions to generate some specific rays for an optical system.

*chiefRay = getChiefRay(optSystem,fieldIndex,wavelengthIndex);*

*mariginalRay = getMariginalRay(optSystem,fieldIndex,wavelengthIndex);*

**Step 4:** Specify the parameters to be returned by the ray tracing routine by using *RayTraceOption* Struct. As the user is not interested in everything which comes out from the ray tracer, the ray tracer function accepts an option struct which has flags about what to compute and what not to during the ray tracer. The ray trace option struct has the following fields:

* + Consider Polarization : *False* by default.
  + Consider Surface Aperture : *True* by default.
  + Record Intermediate Results : *False* by default.
  + Compute Geometrical Path Length : *True* by default.
  + Compute Optical Path Length : *False* by default.
  + Compute Group Path Length : *False* by default.
  + Compute Refractive Index : *True* by default.
  + Compute Refractive Index First Derivative : *False* by default.
  + Compute Refractive Index Second Derivative : *False* by default.
  + Compute Group Index : *False* by default.

*[ options ] = RayTraceOptionStruct(ConsiderPolarization,ConsiderSurfaceAperture,RecordIntermediateResults,ComputeGeometricalPathLength, ComputeOpticalPathLength,ComputeGroupPathLength,ComputeRefractiveIndex,ComputeRefractiveIndexFirstDerivative,ComputeRefractiveIndexSecondDerivative, ComputeGroupIndex);*

**Step 5:** Use a ray tracing function to trace your rays through an optical system. The toolbox has two functions to trace rays depending on the input parameters required

* 1. **The main ray tracing routine**

This function requires the optical system, initial ray bundle and an option struct for ray tracing. It returns a ray tracing result as struct which can be directly used to access different ray trace results or can be used as an input to ray trace result accessing functions which extract certain part of the result and return in a certain format. (See accessing the ray tracing result section).

*rayTracerResultReshaped = rayTracer(optSystem, objectRayBundle,rayTraceOptionStruct,endSurface,nRayPupil,nField,nWav)*

*% rayTracer: main function of polarized ray tracer from object surface to the*

*% endSurface (inclusive)*

*% The function is vectorized so it can work on multiple sets of*

*% inputs once at the same time. That is array of ray objects.*

*% Inputs*

*% optSystem: data type "OpticalSystem"*

*% ObjectRay: data type "Ray" or array of Ray object*

*% startSurf,endSurf: Indices of start and end surface. ObjectRay is*

*% assumed to be given just after the start surface and it will be traced*

*% till end surface (inclusive)*

*% rayTraceOptionStruct: struct with options indicating what to*

*% compute and consider during ray trace. (See RayTraceOptionStruct()*

*% function for more details)*

*% Output:*

*% rayTracerResult: (array if all surface results are recorded) of "RayTraceResult" or can be*

*% matrix of RayTraceResult objects if the input is array of Ray*

*% object. Size : nSurface X nTotalRay*

*% Note: The intersection points and lengths are all in lens unit.*

* 1. **The multiple ray tracing routine**

This is another ray tracing optioin provided in the MatLightTracer toolbox. The function directly accepts the field indices, and wavelength indices and the number of rays instead of initial ray bundle. It returns the pupil meshgrid as Nx X Ny X 2 matrix of pupil sampling points used for ray tracing and the indices of pupil sampling points which lay outside the system aperture.

*[multipleRayTracerResult,pupilMeshGrid,outsidePupilIndices ] = multipleRayTracer(optSystem,wavelengthIndices,fieldIndices,...*

*nPupilPoints1,nPupilPoints2,pupSamplingType,rayTraceOptionStruct,endSurface)*

*% Trace bundle of rays through an optical system based on the pupil*

*% sampling specified. Multiple rays can be defined with wavelengthIndices (1XnWav),*

*% fieldIndices (1XnField) and the total number of ray will be nRay\*nWav\*nField*

*% That is all rays from each field point with each of wavelegths will be*

*% traced. And the result will be 4 dimensional matrix (nNonDummySurface X nRay X nField X nWav).*

*Created with the Personal Edition of HelpNDoc:* [*Write eBooks for the Kindle*](http://www.helpndoc.com/feature-tour/create-ebooks-for-amazon-kindle)

**Accessing Ray Tracing Results**

**Accessing Ray Tracing Results**

Once the ray tracing is performed, the results can be accessed from the ray trace result struct returned by the ray tracing routines. As all the initial rays are traced simultaneously to make use of the matrix processing capability of Matlab, the ray trace result structure have results corresponding to all rays traced in a matrix. So the structure is basically similar to that of the scalar ray bundle

**Example:**

The RayTraceResult.RayExitPosition struct is arranged in the following manner



Like in the ScalarRayBundle, the RayTraceResult structure has a field called *FixedParameters* which is used to hold information about the total number of field points, wavelengths and pupil points used to create the rays in the RayBundle structure traced. The struct has the following fields

*FixedParameters.TotalNumberOfPupilPoints :* The number of pupil points

*FixedParameters.TotalNumberOfFieldPoints :* The number of field points

*FixedParameters.TotalNumberOfWavelengths :* The number of wavelengths

*FixedParameters*.*LensUnitFactor* : The unit factor used for current system

*FixedParameters*.*WavelengthUnitFactor* : The wavelength unit factor for the system

And the *FixedParameters* can be used in extracting the ray trace result parameters of a ray specified by its field index, wavelength index and pupil index.

In order to further simplify accessing of ray trace results, the toolbox provides special set of functions which are dedicated in extracting certain ray trace results from the ray trace result struct. Those functions can be found in the folder (*"...\MatLightTracer\_GitHub\4. Optical\_System\_Package\RayTraceResult"*). All functions follow certain similar form shown by the following example functions.

**Example 1:** Ray Exit Directions

*[ exitRayDirections ] = getAllSurfaceExitRayDirection( allSurfaceRayTraceResult, rayPupilIndices, rayFieldIndices, rayWavelengthIndices)*

*: Returns a 3XnSurface matrix of the exit ray directions for all surfaces in the raytraceresult struct.*

**Example 2:** Ray failure flags

*[ noIntersectionPoints ] = getAllSurfaceNoIntersectionPoint( allSurfaceRayTraceResult, rayPupilIndices, rayFieldIndices, rayWavelengthIndices)*

*: Returns a 1XnSurface vector indicating flag for no intersection point for the specific ray with all surfaces.*

*Created with the Personal Edition of HelpNDoc:* [*iPhone web sites made easy*](http://www.helpndoc.com/feature-tour/iphone-website-generation)

**Programming and Algorithms Reference**

MatLightTracer : MATLAB Light Tracing Toolbox

Programming & Algorithms Reference

Version 1.0

*Created with the Personal Edition of HelpNDoc:* [*Single source CHM, PDF, DOC and HTML Help creation*](http://www.helpndoc.com/help-authoring-tool)

**Programming Approach**

**Programming Approach**

The toolbox was originally written using object oriented programming approach. It is known that object oriented programs in Matlab are slower than struct based counterparts and since we are not utilizing most features of object oriented approach (such as inheritance at least for the time being), we could systematically replace Classes with Structs with out affecting the overall organization. Therefore, to boost the performance (speed) of the ray creation, ray tracing and result collection routines, most classes were replaced with struct with out changing the overall organization of the toolbox. Like in the object oriented organization, all functions working on a given struct are placed in a single folder named by the struct it represents. So the overall structure is not altered and it is easy to go back to object oriented approach if necessary.

The toolbox is composed of a number of packages each with their own set of modules dedicated for certain specific purpose. The overall toolbox folder structure can be seen from the "General Toolbox Folder Structure" section of the user manual.

*Created with the Personal Edition of HelpNDoc:* [*Easy EPub and documentation editor*](http://www.helpndoc.com)

**Toolbox Structures and Classes**

**Toolbox Structures and Classes**

The backbones of the MatLightTracer toolbox are the structures and classes defined for different entities in the toolbox. All classes and struct defintions are place inside a folder whose name is the same as that of the class/struct. All functions for manupilating the struct/or objects of the class are placed in the same folder. Following is the list of structs and classes with brief description. Some important structures and classes are discussed spearately in the following sections.

(The table has to be completed :( Sorry)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Folder (Package) Name** | **Sub Folder (Module) Name** | **Structs / Classes** | **Fields/Properties** | **Functions/Methods** | **Description** |
| System\_Building\_Blocks\_Package | Optical\_Elements/Surface\_Module |  |  |  |  |
| Optical\_Elements/Component\_Module |  |  |  |  |
| Aperture\_Module |  |  |  |  |
| Glass\_Module |  |  |  |  |
| Coating\_Module |  |  |  |  |
| Material\_Catalogue\_Module |  |  |  |  |
| Grating\_Module |  |  |  |  |
| Light\_Representation\_Package | Ray\_Module |  |  |  |  |
| Gaussian\_Module |  |  |  |  |
| Complex\_Field\_Module |  |  |  |  |
| Light\_Source\_Package | HarmonicFieldSource |  |  |  |  |
| Complex\_Field\_Spacial\_Profile\_Defintions |  |  |  |  |
| Complex\_Field\_Spectral\_Profile\_Defintions |  |  |  |  |
| Complex\_Field\_Polarization\_Profile\_Defintions |  |  |  |  |
| GaussianBeamDecomposition |  |  |  |  |
| Optical\_System\_Package | OpticalSystem |  |  |  |  |
| Optical\_System\_Analysis\_Module |  |  |  |  |
| Optical\_System\_Import\_Export |  |  |  |  |
| RayTraceResult |  |  |  |  |
| Enumeration\_Module |  |  |  |  |
| Math\_Package | Fourier\_Transforms |  |  |  |  |
| Other\_Mathematical\_Functions |  |  |  |  |
| GUI\_Package | @ParentWindow |  |  |  |  |
| @ChildWindow |  |  |  |  |
| GeneralObjectInputGUI |  |  |  |  |
| Documentation\_Package | User\_Manual\_And\_Technical\_Reference |  |  |  |  |
| Source\_Code\_Explorer |  |  |  |  |
| Matlab\_Reference\_Books |  |  |  |  |
| Catalogue\_Files |  |  |  |  |  |
| Database\_Files |  |  |  |  |  |
| Enhanced\_Matlab\_Features |  |  |  |  |  |
| PROGRESS\_Files |  |  |  |  |  |
| Toolbox\_Extensions |  |  |  |  |  |
| Unclassified\_Tools\_Module |  |  |  |  |  |
| XXFiles |  |  |  |  |  |
|  |  |  |  |  |  |

*Created with the Personal Edition of HelpNDoc:* [*Generate EPub eBooks with ease*](http://www.helpndoc.com/create-epub-ebooks)

**GUI Related Class**

**GUI Related Class Definitions**

These are classes that are used for building and operating the graphical user interface of the toolbox. Basically there are two classes used for the GUI.

1. AODParentWindow: Class for the Main GUI of the toolbox.
2. AODChildWindow: Class for all child windows which pop up on the top of the main window.

**Design Philosophy**

* All GUIs are built directly by writing the MATLAB codes to define, position and set properties of the uicontrols with out using GUIDE. This allowed design of more dynamic and easily programmable user interfaces. There will be no .FIG file for the user interfaces and the number of files required for a GUI will be reduced from 2 to 1.

*Created with the Personal Edition of HelpNDoc:* [*Write EPub books for the iPad*](http://www.helpndoc.com/create-epub-ebooks)

AODParentWindow: Class

**AODParentWindow Class**

This class is used to define the main graphical user interface window which allows setting and analyzing optical systems using the toolbox.

Number of Properties: XX

Number of Methods: XX

**Important Methods (Functions)**

*Created with the Personal Edition of HelpNDoc:* [*Easily create Help documents*](http://www.helpndoc.com/feature-tour)

AODChildWindow: Class

**AODChildWindow Class**

This class is used to define the child windows which are used as dialog boxes to input parameters and display output results (graphical, text or tabular) to the user during analyzing optical systems using the toolbox.

Number of Properties: XX

Number of Methods: XX

**Design Philosophy**

All child windows shall be an instances of a single class, the AODChildWindow class, so that they will have uniform layout throughout the toolbox. Defining all child windows as a single class file allows sharing of UI controls among the child windows when necessary. For instance, the setting panel of a child window to trace a scalar ray and polarized ray will have text boxes to accept the normalized field points coordinates and the normalized pupil coordinates in common. So it will be enough to define those shared UI controls only once and use in both windows.

Similarly all child windows will have "Ok" and "Cancel" buttons but they perform different actions for different windows. The buttons are defined only once and their call backs will be determined using switch case statements in the callback functions defined for the buttons.

**Important Methods (Functions)**

*Created with the Personal Edition of HelpNDoc:* [*Easily create iPhone documentation*](http://www.helpndoc.com/feature-tour/iphone-website-generation)

**OpticalSystem: Struct**

**OpticalSystem Struct**

This class is used to define the optical system object which contains all informations related with the optical system defined.

Number of Properties: XX

Number of Methods: XX

Most of the properties defined in the optical system class are used for optical system analysis in the toolbox.

**Important Methods (Functions)**

**1. Main Ray Tracing Function**

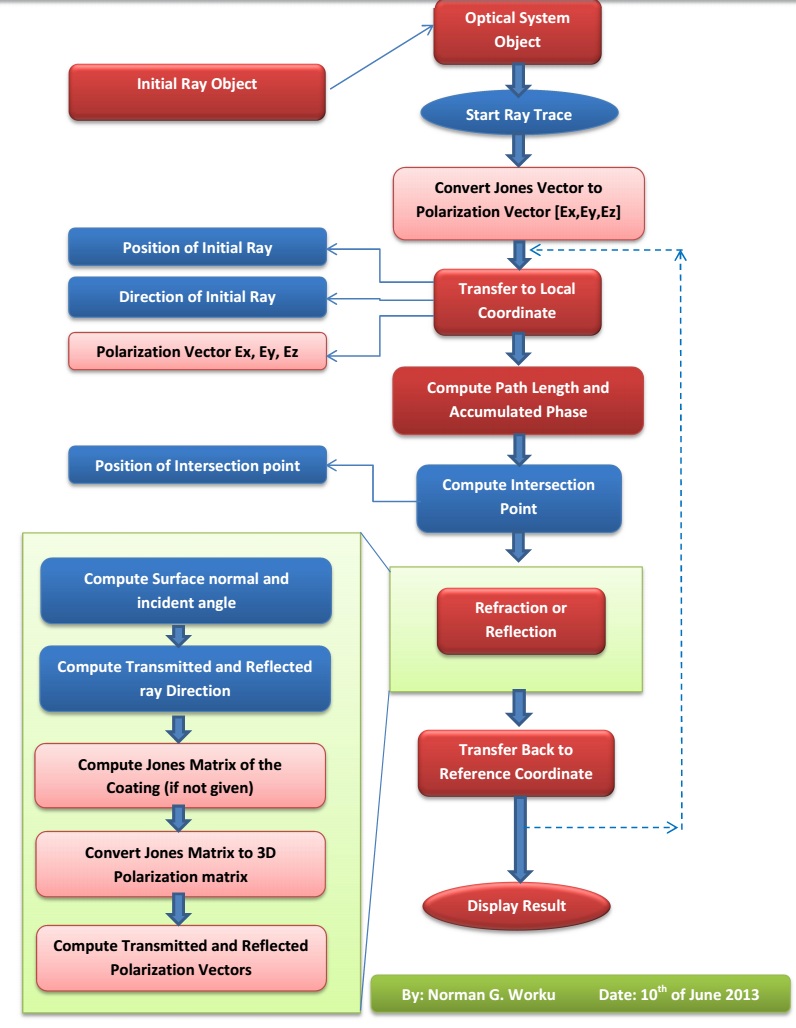
**Syntax:**

**Function Call:**

rayTracerResult = rayTracer(optSystem,objectRay,rayTraceOption)

It is the main ray tracing function which is basis for most analysis features of the toolbox. The function is used to trace Ray objects from object surface to image surface and return all the necessary results of the ray trace.

**Basic Flow Chart:**



**Note:** To simplify tracing multiple rays, an other function called "multipleRayTracer" is defined which does not require the user to construct the initial ray objects explicitly. See the description for "multipleRayTracer".

**2. Multiple Ray Tracing Function**

**Syntax:**

**Function Call:**

[multipleRayTracerResult,pupilCoordinates,pupilGridIndices] = multipleRayTracer(optSystem,wavLen,...

fieldPointXY,nRay,PupSamplingType,JonesVec,considerSurfAperture)

**3. Other Function**

The name of all functions corresponds to their purpose. And all functions are well documented inside with comments whenever necessary. So it is possible to understand the code.

*Created with the Personal Edition of HelpNDoc:* [*Full-featured Kindle eBooks generator*](http://www.helpndoc.com/feature-tour/create-ebooks-for-amazon-kindle)

**Surface: Struct**

**Surface Struct**

This class is used to define the optical surface object, which contains all informations related with the optical surfaces defined.

Number of Properties: XX

Number of Methods: XX

Most of the properties defined in the surface class are used for optical system analysis in the toolbox.

**Important Methods (Functions)**

**1. Surface Coordinate Transformation Matrix Computation**

**Syntax:**

[surfaceCoordinateTM,nextReferenceCoordinateTM] = TiltAndDecenter (surf,refCoordinateTM,prevSurfCoordinateTM,prevThickness)

This function updates the coordinate transformation matrix of the surface from previous reference axis and surface tilt and decenter parameters.

**Tilt and Decenter**

Tilting and decentering of optical surfaces results in non-rotationally symmetric systems which are used in many modern optical systems. Some of those systems include fold mirrors, periscopes, scanning systems and prisms. Modeling of tilt and decenter requires careful treatment of the local coordinate system of the surface and the global coordinate of the system. In the ray trace toolbox three coordinate systems were used to define the surface tilt and decenter of optical systems.

* **Global Coordinate:** It is the main coordinate systems based on which all other coordinate systems are defined. It is assumed to be fixed to the first surface of the optical system. Using the first surface as global coordinate system instead of the object plane avoids problem arising when object at infinity is used.

In addition to the global coordinate, two coordinate systems are established for each surface of the optical system. They are defined using the fixed global coordinate system as reference.

* **Local Coordinate of Surface:** It is coordinate system of each surface. Its origin is the vertex point of the surface and the unit vectors defining the three axes are determined by the orientation of the surface with respect to the global coordinate system.
* **Reference Coordinate after Surface:** It is a coordinate system which acts as reference for the tilt and decenter operation of the surface following current surface in the optical system. This allows the user to set tilt and decenter parameters with respect to previous surface instead of the global coordinate, which is more complex.

Tilt and decenter can be applied to any surface with respect to the reference coordinate and so no special surface is required for coordinate break. And three tilt modes are used to determine how the reference coordinate system should be computed after the tilted/decentered surface. These tilt modes are:

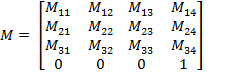
* **DAR (Decenter and Return):** The new reference coordinate after the tilted/decentered surface will return back to that before the surface. The tilt and decenter (X and Y but not Z) operation will be reversed and the coordinate becomes that before the tilt and decentered operation is done but moved forward in z direction by decenter-Z.
* **NAX (New Axis**): The axis of tilted/decentered surface will be the new axis for all surfaces that follow the decentered surface. That is the reference axis will be the current surface local coordinate axis.
* **BEN (Bended Surface):** The new axis after the decentered surface will be determined by the law of reflection. This can only be applied for mirrors and the new coordinate starts at the surface vertex but the axes are determined by the law of reflection. The new axes will still be right handed coordinate system with changed sign of the z-axis.

Surface tilts are defined by specifying Euler angles of rotation Tx, Ty and Tz, about the three axes of the reference coordinate system for each surface. All the angles are directly given by the user as surface parameters. The sign of the tilt angles follows a mathematical convention that it is positive for counter-clockwise rotation and negative for clockwise rotation. In an Euler angle system, each of the three tilt operations takes place in the tilted coordinate system of the preceding tilt. Thus, tilting is non-commutative and undoing tilts requires operations in the reverse order.

Surface decenter values indicate the location of surface vertex with respect to the surface reference coordinate. The x and y decenters values with respect to the origin of the reference coordinate system is entered by the user as surface parameter. And the surface thickness is taken as the decenter in the z-direction.

For mathematical purpose, the surface local and reference coordinate systems are represented by 4x4 matrices called the coordinate transformation matrices. A coordinate transformation matrix defines the rotation matrix and decenter vector of a given coordinate system with respect to the global coordinate system.

This simplifies the conversion of local/reference to global coordinates system and vice versa for each surface in the optical system. The coordinate transformation matrix M is given as:

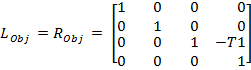


The 3x3 matrix in the upper left side of the transformation matrix corresponds to the total rotation matrix and the last column represents the total decenter vector of the surface local coordinate with respect to the global coordinate. The last row is not necessary but is added just to simplify matrix operations.

As mentioned above the tilt and decenter operations are not commutative, the order shall be specified by the user. Once the tilt and decenter data of each surface in the optical system is specified, the coordinate transformation matrices for each surface local and reference coordinate systems can be computed.

In the following discussion the L and R will be used for the coordinate transformation matrix of the surface local and reference coordinate with respect to the global coordinate respectively. And LR will be used for the transformation matrix of surface local coordinate with respect to the reference coordinate.

For object surface the surface local and reference coordinate are made the same and the corresponding coordinate transformation matrix to global coordinate will be:



where T1 is the first thickness after object surface.

As the global coordinate is fixed to the first surface next to the object, the z decenter value will be negative of the thickness.

As the tilt and decenter data of the next surface are given with respect to the previous reference coordinate system, the local to reference coordinate transformation matrix of the surface can be computed from the tilt and decenter data sequentially using the operations given in the following table. The algorithm starts with the local to reference transformation matrix for object surface, which is an identity matrix , as both coordinate systems are identical for the object surface.

|  |  |
| --- | --- |
| **Tilt/Decenter** | **Operation on the Transformation Matrix** |
| Dx |  |
| Dy |  |
| Dz |  |
| Tx |  |
| Ty |  |
| Tz |  |

Once the surface local to reference coordinate system transformation matrix is determined, the surface local to global coordinate transformation matrix of the surface can then be computed by:



To continue with the next surface, the new reference coordinate system  has to be determined based on the tilt mode of the current surface.

The following table summarizes the operations required to get , the transformation matrix for the next reference coordinate, depending on the surface tilt mode.

|  |  |
| --- | --- |
| **Tilt Mode** |  |
| Decenter and Return (DAR) | Make the next reference coordinate the same as the previous reference coordinate but translated in z by the thickness. |
| New Axis (NAX) | Make the next reference coordinate the same as the current surface local coordinate. |
| Bend (BEN) | Compute the next reference coordinate  from the current surface local coordinate by following the following two Steps:   1. Apply Tx and Ty for the second time to get the new axis (bended axis) 2. Compute and apply new Tz so that a meridional ray will remain a meridional ray in the surfaces following the BEN surface [3].) |

*Created with the Personal Edition of HelpNDoc:* [*Full-featured Kindle eBooks generator*](http://www.helpndoc.com/feature-tour/create-ebooks-for-amazon-kindle)

**Glass: Struct**

**Glass Struct**

This struct is used to define the glass object, which contains all informations related with the glass defined.

Number of Properties: XX

Number of Methods: XX

All of the properties defined in the glass class are used for optical system analysis in the toolbox.

**Important Methods (Functions)**

**1. Abbe Number Computation**

**Syntax:**

[abbeNumber] = getAbbeNumber(Glass,wavLenF,wavLenD,wavLenC)

This function returns Abbe number of glass which is computed from the refractive indices of the glass at three different wavelengths using the formula:

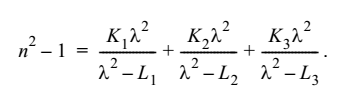
abbeNumber = (nd-1)/(nf-nc)

**2. Refractive Index Computation**

**Syntax:**

[n] = getRefractiveIndex(Glass,wavLen)

This function returns the refractive index of the glass at any given wavelength using Sellmeir equation.



**Note 1:** Both functions are vectorized i.e. giving a vector of wavelengths results in a vector of refractive indices or Abbe numbers.

*Created with the Personal Edition of HelpNDoc:* [*Produce electronic books easily*](http://www.helpndoc.com/create-epub-ebooks)

**Coating: Struct**

**Coating Struct**

This Struct is used to define the coating object, which contains all informations related with the coating defined.

Number of Properties: XX

Number of Methods: XX

Most of the properties defined in the coating class are used for optical system analysis in the toolbox.

**Important Methods (Functions)**

**1. Transmission and Reflection Coefficient Calculations**

**Syntax:**

[ampRs,ampRp,powRs,powRp,JonesMatrix] = getReflectionCoefficients...

(Coating,wavLenInUm,incAngle,ns,nc,primaryWaveLenInUm)

[ampTs,ampTp,powTs,powTp,JonesMatrix] = getTransmissionCoefficients...

(Coating,wavLenInUm,incAngle,ns,nc,primaryWaveLenInUm)

These functions compute the amplitude and power coefficients of transmission and reflection using general Fresnel's equations. Both functions are vectorized so they can work on multiple sets of inputs once at the same time. i.e incAngle or wavLen becomes array. For N number of rays the dimensions of amp(pow)Ts(p) would be 1xN and that of Jones Matrix would be 2x2xN.

The amplitude and power coefficients for both polarizations are separately returned and the Jones matrix which is the collection of amplitude coefficients is also computed. The Jones matrix is the same as [Ts , 0 ; 0 , Tp] or [Rs , 0 ; 0 , Rp] for all coating types except ideal jones matrix which can also have non-zero values in the non diagonal terms.

The functions use the Fresnel's coefficient computation routine for general multi-layered system. See the description for Fresnel's coefficient computation in Polarization Ray tracing module of the toolbox.

.

**2. Computation of the Refractive Index & Thickness of the Coating**

**Syntax:**

[ refIndexAll,thicknessAll ] = getRefractiveIndexThicknessTable(coating,wavLenInUm, primaryWaveLenInUm )

The function is used to compute the refractive index and thickness of a given multilayer coating by considering the following points:

* + Change thickness in relative value to the absolute thickness using the primary wavelength. The actual thickness of the coating is determined by: d = (wavLen0/n0)\*T

where wavLen0 is the primary wavelength in micrometers , n0 is the real part of the index of refraction of the coating at the primary wavelength, and T is the "optical thickness" (relative thickness) of the coating layer.

* + The multilayer are repeated NumberOfRepetetion times.
  + The array will be flipped upside down if the coating is to be used in reverse.

**3. Computation of Coating properties versus incidence angle and wavelength**

**Syntax:**

plotCoatingReflectionVsAngle(coating,wavLenInUm,minAngle,maxAngle,...

angleStep,primWavLenInUm,indexBefore,indexAfter,...

axesHandle,tableHandle,textHandle)

plotCoatingReflectionVsWavelength(coating,incAngle,...

minWavelengthInUm, maxWavelengthInUm, wavelengthStepInUm,primWavLenInUm,...

indexBefore,indexAfter,axesHandle,tableHandle,textHandle)

These functions plot properties of a coating in a given axes.

*Created with the Personal Edition of HelpNDoc:* [*Free HTML Help documentation generator*](http://www.helpndoc.com)

**Scalar Ray Tracing Module**

**Scalar Ray Tracing Module**

It is a module which contains functions which are used for scalar ray tracing. The general flow of ray tracing is described in the documention section of OpticalSystem struct. Here the specific algorithms and mathematical formulations used in the ray tracing are described.

*Created with the Personal Edition of HelpNDoc:* [*Single source CHM, PDF, DOC and HTML Help creation*](http://www.helpndoc.com/help-authoring-tool)

**Coordinate Transformations**

**Coordinate Transformations**

As described in the documentation for Surface class, each surface defines its own local coordinate system. The coordinate transformation matrix for transforming the global coordinate to the local one is given as a 4x4 matrix for each surface. And as it can be seen from the ray tracing flow chart, (see the documentation section for OpticalSystem struct), ray tracing through a surface requires coordinate transformations between the local and global coordinate systems. Those transformations can be simply done by multiplying with the rotation matrix and adding/subtracting the decenter vector. Both rotation matrix and decenter vectors are included in the coordinate transformation matrix of the surface.

**Coordinate Convention**

In this toolbox the global coordinate is fixed to the base of the first surface of the optical system (that is the surface next to object plane). This enables the infinity thickness after the object plane which is used in the case of defining a collimated bundle of rays. So the coordinate transformation matrices of each surface are computed using this global coordinate and their respective tilt and decenter parameters.

*Created with the Personal Edition of HelpNDoc:* [*Easily create EPub books*](http://www.helpndoc.com/feature-tour)

**Intersection Point and Surface Normal Computation**

**Intersection Point and Surface Normal Computation**

Once all initial ray data are converted to local coordinates of the current surface, then the next step is to compute the intersection point of the rays with the surface and the corresponding surface normal at the intersection point. This can be done by analytical formulas for simple standard conic surfaces but require numerical calculation for general case. In this section we will discuss both methods as described in [*G.H.Spencer and M.V.R.K Murty 1962, General Ray-Tracing Procedure*]

The method for computing intersection points and surface normals involves three basic steps. To keep the analytical formulas handy, only spherical and conic surfaces are used in the analytical case. The numerical method is suitable for any surface given its implicit definition F(X,Y,Z) with the corresponding derivatives.

**Step 1: Calculation of the Path Length**

After the coordinate system is transferred into the local coordinate, it is convenient to determine first the intersection of the ray with the z = 0 plane.

At first, s0 is calculated by using:



Then, the intersection point with the Z = 0 plane could be calculated as:







Thus, the parameter p of the distance along the ray measured from (X0, Y0, Z0) is now replaced by s0 and sf .



sf is measured from the point (X1, Y1, 0) to the intersection point on the surface.

When the surface is a plane, sf = 0; thus p = s0 .

But when the surface is not a plane and the additional path should be computed using the following methods :

**Spherical surface: Analytic Formula**

The implicit surface equation for spherical surface is given by



with

.

Let the coordinates of P, the intersection point, be (X, Y, Z) and the length of the segment from (X1, Y1, 0) to (X, Y, Z) is sf =  (the additional path), we have







The result for the additional path is given by (See the original publication for detailed derivation)



where





No loss of significant figures when the form is used numerically.

In this form, the value of  can be substituted to give , the intersection point of the incidence; this completes the transfer process.

**Conic section: Analytic Formula**

For simple conic sections the implicit surface equation for spherical surface is given by



with



The result for the additional path is given by (See the original publication for detailed derivation)



**General Surface: Numerical Method**

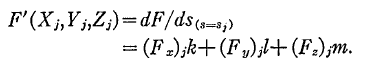
Instead of direct analytical computation, the additional path sf can be computed numerically using Newton Rampson iteration method. Using the subscript j to denote the iteration number, we write



where



and



The process may be started from the first approximation

s1 = 0

and is terminated with the value sf for which



for a given tolerance e.

The above iterations usually converges fast, but if the derivative for F(X,Y,Z) becomes zero then it will not converge. Such cases are rare in practice usually occur for grazing incidence. In such cases the iteration will just oscillate and at some point will diverge to infinity. Therefore we have set the maximum iteration number set to catch such cases. In addition to that, if the ray intersects the surface in more than one point then the iteration could converge to wrong value. But such cases are rare in practice.

**Step 2: Computation of the Intersection Point**

After the path length is computed, the intersection points can be easily calculated using the following formulas







Here

* + The path length of the ray 
  + The start point 
  + The direction unit vector 

**Step 3: Surface Normal Computation**

Before calculating the refraction, the **normal vector at the surface** should be calculated according to the type of the surface, the curvature of the surface, the shape of the surface, and the position of the intersection point.

**Spherical Surface: Analytic Formula**

The normal vector of spherical surface:



* +  is the radius of this spherical surface, which means we could use
  +  the radius curvature instead of  in this function.

**Conic Surface: Analytic Formula**

For a conic section given by:



* + c is the curvature
  +  is the parameter which defines the shape of the surface.

The normal vector becomes:



**General Surface: Numerical Method**

The surface normal of a general surface defined implicitly by F(X,Y,Z) is given as the partial derivative in each direction



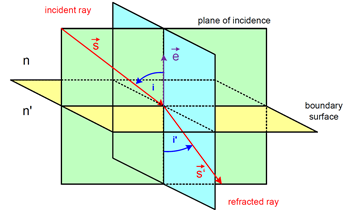
Actually the surface normal is computed as part of the numerical iteration during the intersection point computation in the 1st step.

*Created with the Personal Edition of HelpNDoc:* [*Full-featured Documentation generator*](http://www.helpndoc.com)

**Refraction (or Reflection)**

**Calculation of Refraction or Reflection**

In the toolbox functions which use the general equation for Snell’s law in three dimensions to compute the new refracted and reflected direction.



The three dimensional version of Snell’s law states that the direction of the refracted ray is given by the following relation.



And special case of reflection occurs when  and  and the relation reduces to



* + 
  + 
  + 

The condition of total internal reflection is determined simply by checking the new ray direction for being complex number.

*Created with the Personal Edition of HelpNDoc:* [*Easily create EPub books*](http://www.helpndoc.com/feature-tour)

**Incidence Angle**

**Calculation of Incidence Angle**

Incidence angles corresponding to local angle between the incident ray and the normal vector of the surface at the intersection point can be computed as follows:

Direction cosines of the incident ray at the surface j 

Local unit normal vector at the surface j 

The incident angle is



Since the coordinates are rotational symmetrical, the angle is always positive.

*Created with the Personal Edition of HelpNDoc:* [*Free Kindle producer*](http://www.helpndoc.com/feature-tour/create-ebooks-for-amazon-kindle)

**Paraxial and Meridional Ray Tracing Module**

**Paraxial and Meridional Ray Tracing**

Spherical surfaces (including plane surface with radius ->inf) are the most commonly used surfaces in optics. Rays incident to spherical surfaces are generally categorized in to three main categories:

* **Meridional Rays:** are those rays which lie in a meridional plane which also contains the optical axis of the system, the surface normal and the refracted rays.
* **Skew Rays:** are rays that are not in a common plane with the optical axis.
* **Paraxial Rays:** are meridional or skew rays which have small angle from the optical axis.

Paraxial and meridional ray tracing form the basis for a great deal of optical aberration theory developed so far for conventional reflective and refractive surfaces with rotational symmetric arrangements. These include all first order properties of the system such as focal length, F/#, and pupil sizes and locations, seidel aberrations and gaussian beam data.

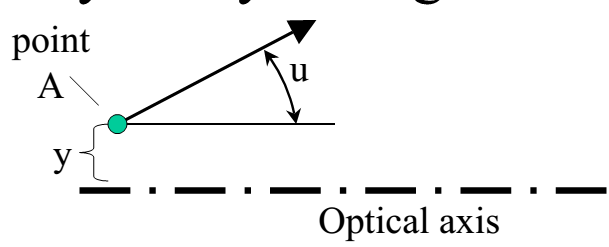
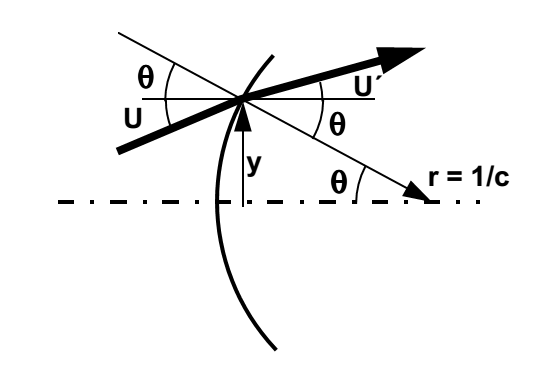
Both paraxial yni and meridional QU ray trace functions are written in such a way that both forward and backward ray tracing is possible.

*Created with the Personal Edition of HelpNDoc:* [*Easy to use tool to create HTML Help files and Help web sites*](http://www.helpndoc.com/help-authoring-tool)

**Paraxial Ray Tracing Module**

**ParaxialRay Tracing**

There are three commonly used methods for paraxial ray tracing, namely: yni method, ynu method and matrix method. In all paraxial ray tracing methods, a ray at a point is given by its height from the optical axis, y, and its slope (tangent of angle), u (for small angles u can be angle in radians).



Although ynu method is commonly used for efficient ray tracing by hand or in simple programs such as excel sheets, some commercial ray tracing programs use yni method as it leads to computation of the paraxial incidence angle as an intermediate result of the ray trace. So, the yni method is used to develop paraxial ray tracing module for our ray tracing toolbox. In paraxial approximation the equations for conic sections becomes that of spherical surfaces.

The important formulas employed for yni ray tracing method are:













In addition to the basic yni ray trace function, the paraxial ray tracing module includes functions to determine:

* The entrance and exit pupil (position and size): using reverse yni ray tracing
* The chief and marginal ray: using the position and size of the pupil
* The back focal length and effective focal length of the system
* The image and object space numerical apertures
* The paraxial magnification

The system aperture is given by either the object space NA or the entrance pupil diameter and the stop surface index is set by the user using the system configuration dialog.

To determine location of entrance pupil

1. Trace an arbitrary paraxial ray from axial point at stop surface back towards to the object side using the paraxial ray tracing.
2. The resulting object side ray is the chief ray for the corresponding ray height.
3. The axial crossing point of the object side ray determines the location of the entrance pupil as, by definition, the chief ray crosses the optical axis at entrance pupil if it is not refracted by the optical system.

Once the location of entrance pupil is determined, the object side numerical aperture and the entrance pupil diameter can computed from each other if one is already known.



The chief ray angle for a given field height can be computed by projecting a ray from the field point to the center of entrance pupil diameter.



Image space numerical aperture can be computed from the marginal ray angles in the image space. And the exit pupil can be located following similar method as for entrance pupil but now tracing the paraxial ray from stop in forward direction to the image side.

The back focal length and effective focal lengths can be computed by tracing a paraxial ray which is parallel to the optical axis and determining its intersection point with the optical axis in the image space.

*Created with the Personal Edition of HelpNDoc:* [*Easily create EBooks*](http://www.helpndoc.com/feature-tour)

**Meridional Ray Tracing Module**

**Meridional Ray Tracing**

For meridional rays tracing the QU method was implemented as it can be applied for both spherical and plane surfaces. In QU method, a meridional ray is defined by the angle U, the angles which the rays make with the optical axis, and distance Q, the vertical distance from the vertex of the surface to the ray.

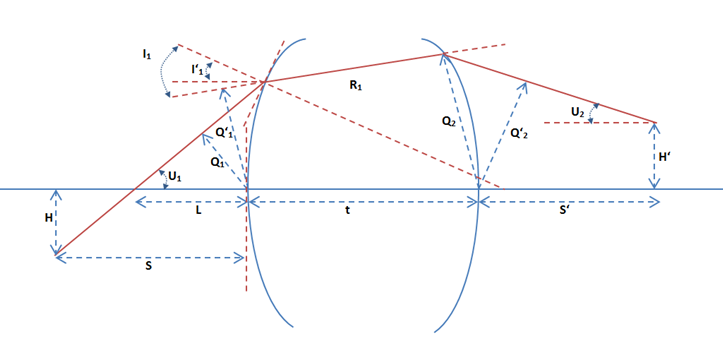


Figure above shows important parameters used in the QU algorithm

In the figure:

* R: Radius of curvature of the surface . It is +ve if the center of curvature is to the right of the vertices.
* c: Curvature of the surface at the vertex. It is +ve if the radius is +ve as c = 1/R.
* U and U’: Angles which the rays make with the optical axis (before and after the surface). The angle is +ve if the slop of the ray is +ve.
* I and I’: Angle of incidence and refraction which the ray makes with the surface normal before and after the surface. They are measured from the surface normal to the ray and are +ve if in counter clock wise direction.
* Q and Q’: The vertical distance from the vertex of the surface to the ray before and after the surface. It is +ve if the ray is above the surface.
* L: distance from the surface vertex to the intersection point of the ray with the optical axis. It is +ve if the intersection point is to the right of the surface vertex.
* S and S’: distance from the surface vertex to the ray location along the optical axis before and after the surface. It is +ve if the ray is located to the right of the surface vertex.
* H and H’: the vertical distance from optical axis to the ray location before at object and image plane. It is +ve if the point is above the optical axis.

The QU algorithm

1. Express the initial meridional ray in QU format using the following relations from the position [0,Py,Pz] and direction [0,Dy,Dz] of the ray



1. Transfer the ray to the next surface using equations







1. Compute the ray position and direction from the final QU parameters using



For plane surfaces:



For Others:



 *the ray angle from the surface normal*

*Created with the Personal Edition of HelpNDoc:* [*Free EPub producer*](http://www.helpndoc.com/create-epub-ebooks)

**Polarization Ray Tracing Module**

**Polarization Ray Tracing Module**

It is a module which contains functions which are used for polarization ray tracing. The general flow of polarization ray tracing is described in the documention section of OpticalSystem class. Here the specific algorithms and mathematical formulations used in the polarized ray tracing are described.

**Light and Polarization**

Classically, light can be defined as an electromagnetic field which consists of oscillations of photons which occur in the direction perpendicular to that of the energy flow. Hence light is called transverse electromagnetic wave. The oscillation of a single photon can be described by equation shown below.



Where



Depending on the magnitudes of its components, the wave vector has certain orientation in space. Such orientation of wave vector in space is known as the polarization of the wave vector[2] .

**Polarization States**

The polarization state of a plane electromagnetic wave is given by the curve which the tip of the electric field vector  follows in a plane which is transverse to the direction of propagation. Polarization state can be determined from the phase difference and magnitude of different components of the electromagnetic wave.

There are several mathematical formalisms to describe the polarization state of a given electromagnetic wave at a given frequency. Some of these include :

* Jones Vector
* Ellipse of Polarization

**Jones Vector**

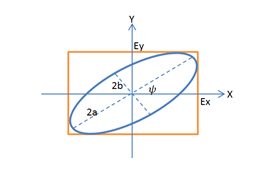
Due to transversal nature of fields (no component along the direction of propagation), any field can be represented by a complex two component vector known as Jones Vector. The polarization state of a polarized field propagating along z–axis is represented by



If the field is not propagating along the z-axis, then the x-y coordinates are referred to as “local coordinates” associated with a particular transverse plane . The Jones vector contains the phase, magnitude and polarization of the electric field. But it can be used to describe polarization states of the fully polarized electric fields.

**Ellipse of Polarization**

In this method polarization state of a polarized electric field is represented by the trajectory followed by the tip of field vector in the transversal plane, which is elliptical, for general case.



The ellipse of polarization is characterized by its orientation angle (), ellipticity (e) determined by semi-major (a) and semi-minor (b) axes of the ellipse, and the direction of rotation.

Those ellipse parameters can be derived from the Jones vector by the following relations.



NB. For  the result will be negative and so 90 should be added to get the correct positive angle.



With



**Methods of Polarization Ray Tracing**

**1. Two Dimensional Jones Calculus**

In Jones calculus the polarization states are represented by Jones Vectors and the effect such optical elements on the polarization state is modeled by multiplication of the Jones vectors with a complex 2x2 matrices called the Jones matrices.



The Jones vector after the optical component is given by

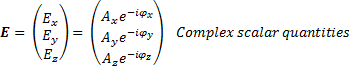


For a sequence of multiple optical elements, the effective Jones matrix can be determined by multiplying individual Jones matrices in reverse order as given in the following equation:



**2. Three Dimensional Ray Tracing Calculus**

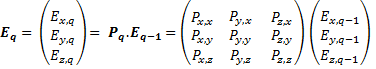
In three dimensional ray tracing calculus, the polarization states of a field are represented as three element electric field vectors. They are simply vectors containing the components of the electric field in the global coordinate system.





Polarization ray tracing matrix P is a 3x3 matrix which transforms the three element electric fields vector as it propagates through an optical element or a sequence of optical elements or even the entire optical system .

If Eq-1 and Eq represent the electric field vector entering and leaving a qth optical element, and Pq represents the polarization ray tracing matrix of the element, then



Note that the polarization ray tracing matrix Pq is associated with specific incident and exciting propagation vectors, kq-1 and kq.

As in the case of Jones calculus, sequence of optical elements, assuming isotropic media between elements, is represented by multiplying the P matrices in the reverse order to get the net polarization ray tracing matrix PTotal as follows [4]:



Propagation through an isotropic media can be represented by identity matrices, as it involves multiplication of constant phase factor. But for anisotropic and birefringent media, the polarization ray tracing matrix will take the form of retarder matrices and/or diattenuation matrices.

The polarization ray tracing matrix is equivalent to Jones matrix except that the former is formulated in global coordinates whereas the later requires local coordinates. And so it is possible to compute one from the other.

*Created with the Personal Edition of HelpNDoc:* [*Easy to use tool to create HTML Help files and Help web sites*](http://www.helpndoc.com/help-authoring-tool)

**Jones Matrix to Polarization Ray Tracing Matrix**

**Transformation of Jones Matrix to Polarization Ray Tracing Matrix**

Because of their equivalence, it is possible to obtain Polarization Ray Tracing Matrix from the Jones Matrix of an optical element for a given incident and exiting directions, **k**q-1 and **k**q. In Jones calculus, calculation of reflection and refraction at qth interface (optical element) requires transformation of the field from global {x,y,z} to local s-p coordinates before the interface {sq,pq,kq-1}and then from local s-p after the interface {s’q,p’q,kq} back to the global coordinates.

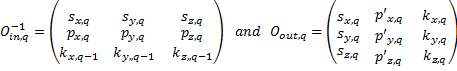
The local coordinates before and after the interface are related as follows. The reflected and transmitted rays kq both lie on the same plane which contains the incident ray kq-1. The p (p for “parallel”) components of the local coordinates also lie in this plane by definition. As the s (s for “senkrecht” or perpendicular in Deutsch) components of the local coordinate systems are perpendicular to the plane containing p and k- components, it will be the same before and after the interface.



And the p - components are perpendicular to both k- and s- components so



Given the local coordinates before and after the interface in terms of global xyz coordinate  orthogonal matrices for the transformation of the electric field vector from global to local and then back to local coordinate are given by [7] as follows:



 acts on incident field **E**q-1 in global coordinate system and transforms to {sq,pq,kq-1} local coordinate and  transforms the fields in the local coordinate {s’q,p’q,kq} back to the global coordinates.

The conversion of a 2x2 Jones Matrix, which is defined in the local {s,p} coordinates, to the polarization ray tracing matrices involves the following steps:

1. The incident field vector Eq-1 is transformed to the local {sq,pq,kq-1} coordinate



1. The resulting vector is multiplied with the Jones matrix. But before that the Jones matrix should be converted to 3x3 matrix by padding a row and a column (with 1 in their last term and 0 in the rest).



This vector is automatically in the local coordinate system {s’q,p’q,kq} after the element.

1. Finally the vector is transformed back to the global

. This will be the electric field vector Eq after the optical element.

So



Therefore, the polarization ray tracing matrix  is given by



Note that a given Jones matrix can give different polarization ray tracing matrices depending on the incident and exiting ray directions.

*Created with the Personal Edition of HelpNDoc:* [*Easily create EBooks*](http://www.helpndoc.com/feature-tour)

**Polarization Ray Tracing through Interfaces**

**Polarization Ray Tracing through Interfaces**

In Jones calculus the reflection and refraction at dielectric, metal and multilayer coated interfaces are described in terms of {s,p} components. The polarization ray tracing matrix for a refraction (transmission) and reflection can be obtained from the corresponding Jones matrix (Jt for transmission and Jr for reflection).

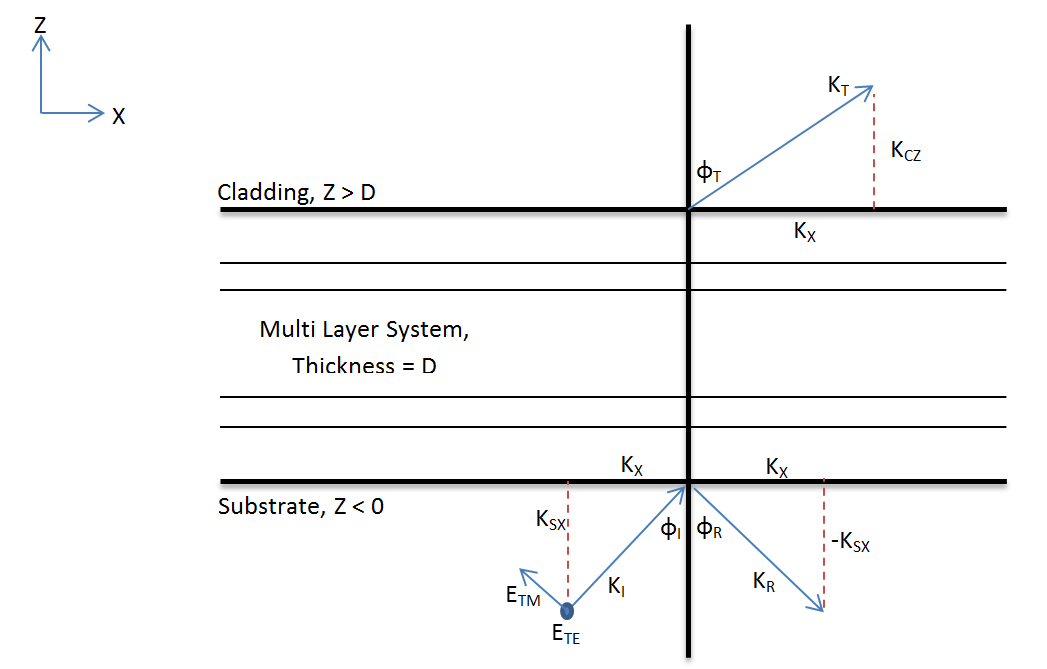


Where the  are the amplitude transmission (reflection) coefficients for s or p - components of the polarized field. For single plane interfaces, the coefficients can be calculated from the Fresnel’s equations and for layered interfaces matrix method for stratified media, which leads to the general form of Fresnel’s equations can be applied [9][10]. In the following section the general method of field propagation through stratified media is discussed first followed by brief summary of special case of single plane interface.

Reflection-Transmission Problem for General Layer System

For problems with translational invariance in at least one direction, (like in homogenous infinite media, layers or interfaces), an electric field can be decomposed to two independent components. The field component which is perpendicular to the plane of propagation is called the transverse electric (TE) component and the other one which is parallel to the plane of propagation is called the transverse magnetic (TM) components.

The transmission and reflection coefficients of a multi-layer system shown in the following figure for each component (TE and TM) can be compute independently and are given in the following section.



As field is propagating in x-z pane, the wave vector has only two components kX and kZ, tangential and normal to the interface respectively. The tangential component of the wave vector is conserved throughout the entire structure. The total length of the wave vector on each layer is given by the dispersion relation for dispersive, isotropic, homogeneous media. As a consequence, the normal component (kZ) varies and depends on the electric permittivity of each layer.



In substrate and cladding the wave vectors can be related to the refractive indices, incident angle and transmitted angle by the following relations:





To compute the transmission and reflection coefficients of a layered system, a matrix method for fields in layer system can be applied. For the method the system is represented by a transfer matrix. The transfer matrix for a single layer can be computed from its thickness and permittivity, using continuity of transverse component of electric field and wave vectors at the boundary. For multilayer system, the transfer matrix is the product of the transfer matrices of each layers of the system in reverse order.











Once the transfer matrix for a multilayer system is determined, the coefficients of transmission and reflection can be obtained using the following relations. Only the final result is presented here.

1. TE Polarization (s) Component

Reflection Coefficient



Transmission Coefficient



1. TM Polarization (p) Component

Reflection Coefficient



Transmission Coefficient



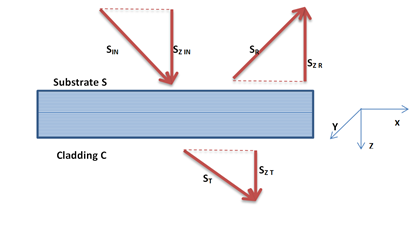






The coefficients computed above describe the relation between the complex amplitudes of incident, reflected and transmitted plane waves. They are called generalized Fresnel’s coefficients.

But for physical reasons, it is common to consider the portion of reflected and transmitted energy fluxes instead of the complex amplitudes. Energy flux is defined by the normal component of the Poynting vectors sz.



The coefficients relating the incident energy flux with the reflected and transmitted energy fluxes are called the reflectance (r) and transmittance (t) of the system. They can be related to the Fresnel’s coefficients as flows:





In addition to reflection and transmission, part of incident energy will also be absorbed in the multilayer structure. Since the energy should be conserved, the equation

is valid for both TM and TE polarizations where ‘a’ is absorption coefficient .

The matrix method can be used for general case of multi-layer system such as multi-layer coatings. And for bare uncoated glass the multilayer system becomes only a single interface and the general matrix methods simplifies to set of Fresnel’s equations for single interface .

*Created with the Personal Edition of HelpNDoc:* [*Full-featured Help generator*](http://www.helpndoc.com/feature-tour)

**Polarization Properties of Optical System**

**Analysis of Polarization Properties in Three Dimensional Ray Tracing Calculus**

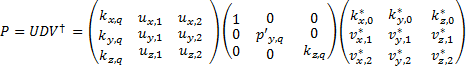
One of the purposes of polarization ray tracing is determination of the diattenuation and retardance associated with ray paths through an optical system. It can also be used to compute the polarization aberration function of an optical system. In this section those polarization properties will be defined and algorithms for their computation using the three dimensional polarization ray tracing calculus will be presented.

**Diattenuation**

Diattenuation: is measure of polarization dependent transmittance of a given optical element. Literally diattenuation means "two attenuations", and is used to compare the loss of intensity of the s polarized light compared to the p polarized light. It depends on the maximum and minimum intensity transmittances considered over all incident polarization states as:



In Jones calculus the eigenvectors of Jones matrix represent the two polarization states for maximum and minimum transmittance and thus the diattenuation can be computed from the corresponding eigenvalues. But for three dimensional polarization ray tracing calculus, eigenvectors of the P matrix do not generally represent polarization states because in general light rays enter and exit in different directions. The diattenuation of the P matrix can be calculated by using the singular value decomposition (SVD) .

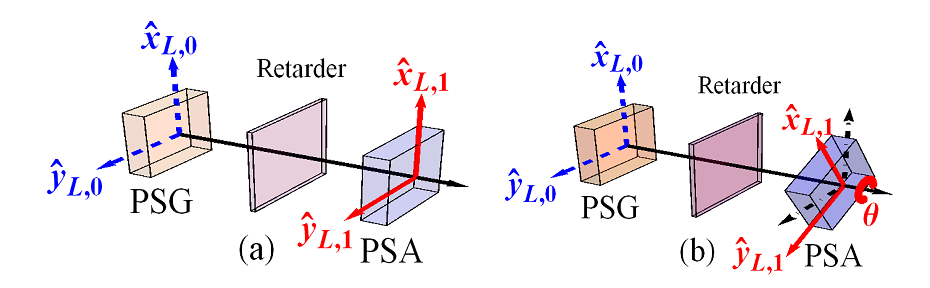


By definition the diagonal elements of D matrix are the singular values of P. The first column of V is incident wave vector K0. The other two columns of V, v1 and v2, are two special polarization vectors in the incident transverse plane that generate the maximum and minimum transmitted flux. Similarly, the columns of U are the exiting propagation vector KQ and two orthogonal polarization vectors u1 and u2 in the exiting transverse plane. Expressing an arbitrary incident polarization as linear combination of these orthogonal polarization states v1 and v2 , the following relation can be derived for diattenuation .



**Retardance**

Retardance is defined as polarization dependent phase change or optical path difference associated with a ray path through an optical system. In Jones calculus the proper retardance is defined as the actual accumulation of polarization dependent optical path difference due to the optical elements. In addition to that a polarization state analyzer reads additional retardance if the exiting local coordinates are not parallel to the local incident coordinates. This is the retardance due to local coordinate transformation. The following figure illustrates this,



If the retarder is just an empty compartment and has identity jones matrix, the polarization state analyzer in (a) measures the unity matrix (zero retardance). But if the analyzer is roated by some angle, the incident and exiting local coordinates are no longer parallel and so the analyzer will measure the rotation matrix instead of identity matrix.

Therefore, while computing retardance it is necessary to separate the proper retardance from the one which is caused by coordinate transformation. The concept of parallel transport ray tracing matrix Q was introduced to perform such separation in three dimensional calculus.

A parallel transport ray tracing matrix Qq for qth surface (polarizing element) is defined as a 3X3 ray tracing matrix calculated by assuming that the surface/element is non-polarizing. That is, Qq provides well defined relation between local coordinate before and after the surface/element. Just as for the P matrix, the cumulative parallel ray transport matrix can be defined as the product of all consecutive Q’s.



Therefore, Q-1Total can reverse all the geometric transformation. And the operation



gives a polarization ray tracing matrix MTotal with all the effects of coordinate transformation removed (i.e. with the exiting local coordinate made parallel to the incident ray local coordinate). Now the retardance computed from the M matrix will be the proper retardance of the system.

Any element (corresponding M matrix) can be expressed as a product of a pure retarder with a pure diattenuator, and the retardance is well-defined to be that of the pure retarder. This can be achieved by polar decomposition of M matrix.



Then proper retardance can be calculated as the phase difference of non-unity eigenvalues of MToatl,R.

**Polarization Aberration Function**

Polarization aberration P(h,r,λ) of an optical system is variation of polarization properties of an optical system with object coordinates , pupil coordinates and wavelength. It can be determined by performing polarization ray tracing of a ray bundle through an optical system. As ray tracing program can only trace a limited number of rays, the polarization aberration function P(h,r,λ) determined will be a matrix of sampled values at different locations in the exit pupil, indicated by r. Each element of the matrix indicates the total polarization ray tracing matrices for entire path of the ray passing through that specific location.

Tracing an L X M grid of rays through an optical system results in L X M grid of propagation vectors, L X M grid of positions and L X M grid of optical path lengths.







And the sampled polarization aberration function will be L X M grid of the polarization tracing matrices at each location. If the exit pupil is not rectangular, the entries corresponding to locations outside the pupil will be filled with zeros.



As arguments of the complex matrices of the polarization aberration function are always less than , optical path lengths are calculated separately.

*Created with the Personal Edition of HelpNDoc:* [*Easily create EPub books*](http://www.helpndoc.com/feature-tour)

**Examples of Extending the Toolbox**

**Examples of Extending the Toolbox**

To extend the functionalities of the toolbox there are two possibilities.

1. Adding feature which becomes the integral part of the toolbox
2. Adding features which is not integrated to the toolbox but uses the toolbox to setup and analyze optical systems.

Each approach has advantages and disadvantages. The following sections discuss each method by giving an example of extending the toolbox.

*Created with the Personal Edition of HelpNDoc:* [*Easy to use tool to create HTML Help files and Help web sites*](http://www.helpndoc.com/help-authoring-tool)

**Method 1: As part of the toolbox**

**Method 1: As part of the toolbox**

In this method some the existing codes of the toolbox are modified and the new feature becomes integrated as part of the toolbox.

**Advantages:**

* It allows making use of already defined uicontrols for the graphical user interface.
* The user interface of the new feature becomes similar to all windows of the toolbox.
* The feature can easily be reused as it becomes the part of the toolbox.

**Disadvantages:**

* Requires editing the existing code which could be error prone to the whole toolbox.
* It requires some understanding of the existing code of the toolbox.

**Example:** See the transverse ray aberration diagram.

*Created with the Personal Edition of HelpNDoc:* [*Easy to use tool to create HTML Help files and Help web sites*](http://www.helpndoc.com/help-authoring-tool)

**Method 2: As separate tool using the existing toolbox**

**Method 2: As separate tool using the existing toolbox**

In this method a new separate tool will be developed and none of the existing codes of the toolbox are modified. The new feature becomes separated from the toolbox.

**Advantages:**

* Does not requires editing the existing code, which could be error prone to the whole toolbox.
* It does not requires deep understanding of the existing code of the toolbox. Just high level understanding would be enough.

**Disadvantages:**

* It requires completely new graphical user interface and hence could not be similar to other windows of the toolbox.
* The feature will not be easy to reuse as it is separate application.

**Example:** See the Longitudinal ray aberration.

*Created with the Personal Edition of HelpNDoc:* [*Full-featured EPub generator*](http://www.helpndoc.com/create-epub-ebooks)

**Add New Surface Types**

**Add New Surface Types**

The surface module of the MatLightTracing toolbox ( located at *...\MatLightTracer\_GitHub\1. System\_Building\_Blocks\_Package\Optical\_Elements\_Module\Surface\_Module*) has two main folders:

1. **Surface**

Here a struct *'Surface'* is defined with all properties which are common to all surface types such as *Type, Radius, Conic, Thickness, Glass*, ...

The " *Type* " field determines the specific class of surface type to which the current surface belongs. For instance, the *Type* field could be *Standard, EvenAsphere,...*

In addition to the fields common to all surface types, the *Surface* struct has an additional field called "*UniqueParameres*" which is used to hold parameters which are unique to certain type of surface. The *UniqueParameres* field is struct by itself which can have different sub-fields depending on the surface type.

For instance, for *EvenAsphere* surface type, the polynomial coefficients are fields of the *UniqueParameres* struct*.* But the *UniqueParameres* of the *Standard* surface is just an empty struct as there are no additional parameters unique to *Standard* surface types.

2. **Surface\_Defintions**

This folder has the definition files of each surface type in the MatLightTracer toolbox. The name of the surface type definition files should be the same as the surface type name. For instance *Standard* types are defined in the file named *Standard.m* and *EvenAsphere* surface type is defined in Matlab function file named *EvenAsphere.m.*

With the MatLightTracing it is simple to extend supported surface types by adding a new surface types. It requires writing a separate Matlab function which follows certain common format described in this section. Once the function is written and included in the Surface definition folder, the new surface can be used in any optical systems as any other existing surface types.

Let us assume we want to add an *Toroidal* surface type, then we have to follow the following steps.

**Step 1**: Surface Parameter Identification

Identify the parameters used in the new surface type to be defined. The radius of curvature and conic constant are common to all surface types so you need to find any additional parameter required. In the case of Toroidal surface we need

* Radius of Rotation and
* Polynomial coefficients (C2, C4, C6, C8, C10, C12, C14).

**Step 2:** Add a function named *'Toroidal.m'* to the following folder

*'...\MatLightTracer\_GitHub\1. System\_Building\_Blocks\_Package\Optical\_Elements\_Module\Surface\_Module\Surface\_Definitions'.*

These functions all have the following common format

*[ returnDataStruct] = Toroidal (returnFlag, uniqueParameters ,inputDataStruct)*

* + *Toroidal*: Is the name of the surface type
  + *returnFlag*: an integer number indicating what to be returned to the calling function (for instance a ray tracing routine).
  + *uniqueParameters*: is an struct of unique parameters unique to the current user defined surface/feature.
  + *inputDataStruct*: a structure of additional parameters required for computing the required parameter.
  + *returnDataStruct*: a struct with the results to be returned to the calling function.

NB. The inputDataStruct and returnDataStruct may vary depending on the return type required.

Now based on the return flag, the function computes different things as follows

**1: About the surface**

**inputDataStruct:**

empty

**Output Struct:**

*returnDataStruct.Name*

*returnDataStruct.IsGratingEnabled*

*returnDataStruct.ImageFullFileName*

*returnDataStruct.Description*

**Source Code:**

*surfName = {'Toroidal','TORD'}; % display name*

*% look for image description in the current folder and return*

*% full address*

*[pathstr,name,ext] = fileparts(mfilename('fullpath'));*

*imageFullFileName = {[pathstr,'\Surface.jpg']}; % Image file name*

*description = {['Toroidal: Used to define standard conical surfaces.']}; % Text description*

*returnDataStruct = struct();*

*returnDataStruct.Name = surfName;*

*returnDataStruct.IsGratingEnabled = 1;*

*returnDataStruct.IsExtraDataEnabled = 1;*

*returnDataStruct.ImageFullFileName = imageFullFileName;*

*returnDataStruct.Description = description;*

**Description***:*

Return the general information related with the current surface type.

**2: Surface specific 'UniqueSurfaceParameters' table field names and initial values in Surface Editor GUI**

**inputDataStruct:**

empty

**Output Struct:**

*returnDataStruct.UniqueParametersStructFieldNames*

*returnDataStruct.UniqueParametersStructFieldDisplayNames*

*returnDataStruct.UniqueParametersStructFieldTypes*

*returnDataStruct.DefaultUniqueParametersStruct*

**Source Code:**

*uniqueParametersStructFieldNames = {'RadiusOfRotation','C2','C4','C6','C8','C10','C12','C14'};*

*uniqueParametersStructFieldDisplayNames = {'Radius Of Rotation','C 2','C 4','C 6','C 8','C 10','C 12','C 14'};*

*uniqueParametersStructFieldTypes = {'numeric','numeric','numeric','numeric','numeric', 'numeric','numeric','numeric'};*

*defaultUniqueParametersStruct = struct();*

*defaultUniqueParametersStruct.RadiusOfRotation = Inf;*

*defaultUniqueParametersStruct.C2 = 0;*

*defaultUniqueParametersStruct.C4 = 0;*

*defaultUniqueParametersStruct.C6 = 0;*

*defaultUniqueParametersStruct.C8 = 0;*

*defaultUniqueParametersStruct.C10 = 0;*

*defaultUniqueParametersStruct.C12 = 0;*

*defaultUniqueParametersStruct.C14 = 0;*

*returnDataStruct = struct();*

*returnDataStruct.UniqueParametersStructFieldNames = uniqueParametersStructFieldNames;*

*returnDataStruct.UniqueParametersStructFieldDisplayNames = uniqueParametersStructFieldDisplayNames;*

*returnDataStruct.UniqueParametersStructFieldTypes = uniqueParametersStructFieldTypes;*

*returnDataStruct.DefaultUniqueParametersStruct = defaultUniqueParametersStruct;*

**Description***:*

Returns the names, display names, data types, and default values of all fields of the *UniqueParameters* struct of the current surface type.

* + **FieldNames** – Cell array of actual names of the fields in the struct UniqueParameters. Therefore they should fulfill normal structure field name criterion. Later they can be referenced as *uniqueParamStruct.(fieldNameString)*
  + **FieldDisplayNames** – Cell array of field names to be displayed in user interfaces. They can be any string but short and meaningful names are recommended.
  + **FieldFormats** – Cell array of field formats. They can be one of the following
* ‘numeric’ : for numbers
* ‘logical’ : for Boolean
* ‘char’ : for strings
* {‘Choise1’, ‘Choise1’} : for multiple choice variables
* ‘Glass’ : for Glass names
* ‘Coating’ : for coating names
* ‘SQS’: for sequence of surfaces

**3: Surface specific 'Extra Data' table names and initial values in Surface Editor GUI**

**inputDataStruct:**

empty

**Output Struct:**

*returnDataStruct.UniqueExtraDataName*

*returnDataStruct.DefaultUniqueExtraData*

**Source Code:**

*uniqueExtraDataName = {'ZernikeStandardSagCoefficients'};*

*defaultUniqueExtraData = [0];*

*returnDataStruct = struct();*

*returnDataStruct.UniqueExtraDataName = uniqueExtraDataName;*

*returnDataStruct.DefaultUniqueExtraData = defaultUniqueExtraData;*

**Description***:*

This is used to hold an array of numbers associated with the given surface. This is used if the surface has for example so many polynomial coefficients.

**4: Return the surface sag at given xyGridPoints computed from rayPosition ( Used for plotting the surface)**

**inputDataStruct:**

*inputDataStruct.X*

*inputDataStruct.Y*

**Output Struct:**

*returnDataStruct.MainSag*

*returnDataStruct.AlternativeSag*

**Source Code:**

*surfaceRadius = inputDataStruct.Radius;*

*surfaceConic = inputDataStruct.Conic;*

*X = inputDataStruct.X;*

*Y = inputDataStruct.Y;*

*radiusOfRotation = surfaceParameters.RadiusOfRotation;*

*C2 = surfaceParameters.C2;*

*C4 = surfaceParameters.C4;*

*C6 = surfaceParameters.C6;*

*C8 = surfaceParameters.C8;*

*C10 = surfaceParameters.C10;*

*C12 = surfaceParameters.C12;*

*C14 = surfaceParameters.C14;*

*mainSag = computeToroidalSurfaceSag(surfaceRadius,surfaceConic,...*

*radiusOfRotation,C2,C4,C6,C8,C10,C12,C14,X,Y);*

*returnDataStruct = struct();*

*returnDataStruct.MainSag = mainSag;*

*returnDataStruct.AlternativeSag = mainSag;*

**Description***:*

NB: X and Y are vectors of values corresponding to points on XY plane for which the surface sag has to be computed. The function to compute the surface sag can be either placed in the same function as the surface definition or can be just a separate function. But it is recommended is to keep everything here in the single surface definition file so that addition, edition and removal of any surface types becomes very easy.

**5: Paraxial ray trace results (Ray height and angle)**

**inputDataStruct:**

*inputDataStruct.InputParaxialRayParameters*

*inputDataStruct.IndexBefore*

*inputDataStruct.IndexAfter*

*inputDataStruct.Wavelength*

*inputDataStruct.ReflectionFlag*

*inputDataStruct.ReverseTracingFlag*

**Output Struct:**

*returnDataStruct.OutputParaxialRayParameters*

**Source Code:**

*y = inputDataStruct.InputParaxialRayParameters(1,:);*

*u = inputDataStruct.InputParaxialRayParameters(2,:);*

*reverseTracing = inputDataStruct.ReverseTracingFlag;*

*reflection = inputDataStruct.ReflectionFlag;*

*indexBefore = inputDataStruct.IndexBefore;*

*indexAfter = inputDataStruct.IndexAfter;*

*surfaceRadius = inputDataStruct.Radius;*

*% the height doesnot change*

*yf = y;*

*% for angle compute based on the direction of propagation*

*if ~reverseTracing*

*%forward trace*

*c = 1/surfaceRadius;*

*n = indexBefore;*

*nPrime = indexAfter;*

*else*

*%reverse trace*

*c = -1/surfaceRadius;*

*n = indexAfter;*

*nPrime = indexBefore;*

*end*

*if reflection*

*n = -n;*

*end*

*paI = u+yf\*c; %The yui method generates the paraxial angles of incidence*

*% during the trace and is probably the most common method used in computer programs.*

*uf = u+((n/nPrime)-1)\*paI;*

*outputParaxialRayParameters = [yf,uf]';*

*returnDataStruct = struct();*

*returnDataStruct.OutputParaxialRayParameters = outputParaxialRayParameters;*

**Description***:*

Since the paraxial ray tracing calculation is simple it is just done right here in the code. But it can also be done in a separate function.

**6: New ray direction for real ray tracing**

**inputDataStruct**:

*inputDataStruct.RayDirection*

*inputDataStruct.LocalSurfaceNormal*

*inputDataStruct.IndexBefore*

*inputDataStruct.IndexAfter*

*inputDataStruct.WavelengthInUm*

*inputDataStruct.DiffractionOrder*

*inputDataStruct.GratingVectorDirection*

*inputDataStruct.GratingLinesPerMicrometer*

**Output Struct:**

*returnDataStruct.NewLocalRayDirection*

*returnDataStruct.TIR*

**Source Code:**

*rayDirection = inputDataStruct.RayDirection;*

*localSurfaceNormal = inputDataStruct.LocalSurfaceNormal;*

*indexBefore = inputDataStruct.IndexBefore;*

*indexAfter = inputDataStruct.IndexAfter;*

*wavelengthInUm = inputDataStruct.WavelengthInUm;*

*diffractionOrder = inputDataStruct.DiffractionOrder;*

*gratingVectorDirection = inputDataStruct.GratingVectorDirection;*

*gratingLinesPerMicrometer = inputDataStruct.GratingLinesPerMicrometer;*

*% Use the general snells law*

*[newLocalRayDirection,TIR] = computeGeneralRefractionReflection ...*

*(rayDirection,localSurfaceNormal,indexBefore,indexAfter,...*

*wavelengthInUm,diffractionOrder,gratingVectorDirection,gratingLinesPerMicrometer);*

*returnDataStruct = struct();*

*returnDataStruct.NewLocalRayDirection = newLocalRayDirection;*

*returnDataStruct.TIR = TIR;*

**Description***:*

The new ray direction for real ray tracing can usually be computed using the general\_reflection\_refraction routine which uses the generalized vectorial Snells law together with grating equations for computing the new ray direction after an interface. But this option is provided here and explicitly open for programming in order to allow user defined algorithms used for refracted and reflected ray directions.

**7: Return the function values of F(X,Y,Z) at the given ray intersection points**

**inputDataStruct:**

*inputDataStruct.RayIntersectionPoint*

**Output Struct:**

*returnDataStruct.Fxyz*

**Source Code:**

*R = surfaceParameters.RadiusOfRotation;*

*if abs(R) == Inf*

*R = sign(R)\*10^12; % Since Inf results in NaN*

*end*

*c = (1/inputDataStruct.Radius);*

*k = inputDataStruct.Conic;*

*C2 = surfaceParameters.C2;*

*C4 = surfaceParameters.C4;*

*C6 = surfaceParameters.C6;*

*C8 = surfaceParameters.C8;*

*C10 = surfaceParameters.C10;*

*C12 = surfaceParameters.C12;*

*C14 = surfaceParameters.C14;*

*X = inputDataStruct.RayIntersectionPoint(1,:);*

*Y = inputDataStruct.RayIntersectionPoint(2,:);*

*Z = inputDataStruct.RayIntersectionPoint(3,:);*

*fy = (c.\*(Y.^2))./(1+sqrt(1-(k+1)\*c^2\*(Y.^2)))+...*

*(C2\*Y.^2 + C4\*Y.^4 + C6\*Y.^6 + C8\*Y.^8 + C10\*Y.^10 + C12\*Y.^12 + C14\*Y.^14);*

*Fxyz = Z - fy - (1/(2\*R))\*(X.^2 + Z.^2 - fy.^2);*

*returnDataStruct.Fxyz = Fxyz;*

**Description***:*

The numerical method of intersection point computation requires the implicit surface function F(X,Y,Z) . If our surface has analytical formula then that can be just rearranged in the form F(X,Y,Z) = 0 and used directly here.

**8: Return F'(X,Y,Z),the derivatives function values of F, at the given ray intersection points and the surface normals**

**inputDataStruct:**

*inputDataStruct.RayIntersectionPoint*

*inputDataStruct.RayDirection*

**Output Struct:**

*returnDataStruct.FxyzDerivative*

*returnDataStruct.SurfaceNormal*

**Source Code:**

*c = (1/inputDataStruct.Radius);*

*conic = inputDataStruct.Conic;*

*radiusOfRotation = surfaceParameters.RadiusOfRotation;*

*C2 = surfaceParameters.C2;*

*C4 = surfaceParameters.C4;*

*C6 = surfaceParameters.C6;*

*C8 = surfaceParameters.C8;*

*C10 = surfaceParameters.C10;*

*C12 = surfaceParameters.C12;*

*C14 = surfaceParameters.C14;*

*X = inputDataStruct.RayIntersectionPoint(1,:);*

*Y = inputDataStruct.RayIntersectionPoint(2,:);*

*Z = inputDataStruct.RayIntersectionPoint(3,:);*

*k = inputDataStruct.RayDirection(1,:);*

*l = inputDataStruct.RayDirection(2,:);*

*m = inputDataStruct.RayDirection(3,:);*

*% Compute its the derivative F'(X,Y,Z)*

*[Fx,Fy,Fz] = computeToroidalPartialDerivates(c,conic,radiusOfRotation,...*

*C2,C4,C6,C8,C10,C12,C14,X,Y);*

*Fderivative = Fx.\*k + Fy.\*l + Fz.\*m;*

*surfNormal = [Fx;Fy;Fz];*

*normalizedSurfaceNormal = normalize2DMatrix( surfNormal,1);*

*returnDataStruct.SurfaceNormal = normalizedSurfaceNormal;*

*returnDataStruct.FxyzDerivative = Fderivative;*

**Description***:*

The numerical method of intersection point computation requires the derivatives of the implicit surface function F'(X,Y,Z) . If our surface has analytical formula then that can be just rearranged in the form F(X,Y,Z) = 0 and then the analytical derivatives could be obtained and used directly here.

**9: Return the ray Exit position (This allows the ray input and exit positions to be decoupled)**

**inputDataStruct:**

*inputDataStruct.RayIntersectionPoint*

**Output Struct:**

*returnDataStruct.LocalExitRayPosition*

**Source Code:**

*localRayExitPoint = inputDataStruct.RayIntersectionPoint;*

*returnDataStruct.LocalExitRayPosition = localRayExitPoint;*

**Description***:*

The exit position of the ray from a given surface is usually equal to the intersection point. But in some ideal surfaces it is not the case, for instance in Kostenbauder surface the ray exit position differs from the intersection point depending on the wavelength. To allow modeling of such ideal surfaces the exit ray position computation is made open for user programming.

**10: Return any additional path related to the surface that is not given by the surface sag.**

**inputDataStruct:**

*inputDataStruct.RayIntersectionPoint*

**Output Struct:**

*returnDataStruct.AdditionalPathLength*

**Source Code:**

*intersectionPoint = inputDataStruct.RayIntersectionPoint;*

*% For now just return 0. but shall be corrected*

*additionalPathLength = 0\*intersectionPoint(1,:);*

*returnDataStruct.AdditionalPathLength = additionalPathLength;*

**Description***:*

This is to be used if there is any additional path associated with the surface but not computed from the normal geometric path length computation.

**Step 3:** Add *'Toroidal'* in to list of supported surface of the MatLightTracer toolbox

Once the function Toroidal.m has been successfully implemented and tested, then the final stage of integrating the surface type in to the MatLightTracer toolbox is to add the type in to the supported list. This can be done by modifying the function *GetSupportedSurfaceTypes(index)*  from (*'...\MatLightTracer\_GitHub\1. System\_Building\_Blocks\_Package\Optical\_Elements\_Module\Surface\_Module'*) as follows:

*function [ fullNames,displayNames ] = GetSupportedSurfaceTypes(index)*

*%GETSUPPORTEDSURFACES Returns the currently supported surfaces as cell array*

*if nargin < 1*

*index = 0;*

*end*

*displayNames = {'Standard','Ideal Lens','Example Surface','Kostenbauder',...*

*'Dummy','Even Asphere','ExtendedEvenAsphere',****'Toroidal'****};*

*fullNames = {'Standard','IdealLens','ExampleSurface','Kostenbauder','Dummy',...*

*'EvenAsphere','ExtendedEvenAsphere','Toroidal'};*

*if index*

*displayNames = displayNames{index};*

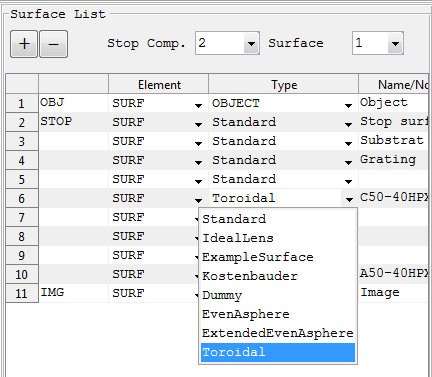
*fullNames = fullNames{index};*

*end*

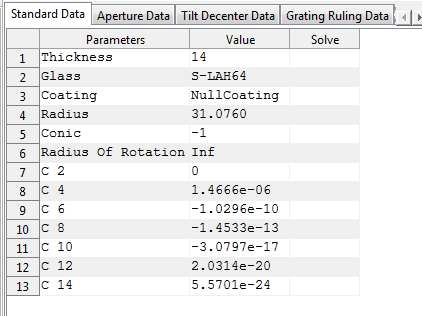
*end*

**Results After New Surface Type Addition**

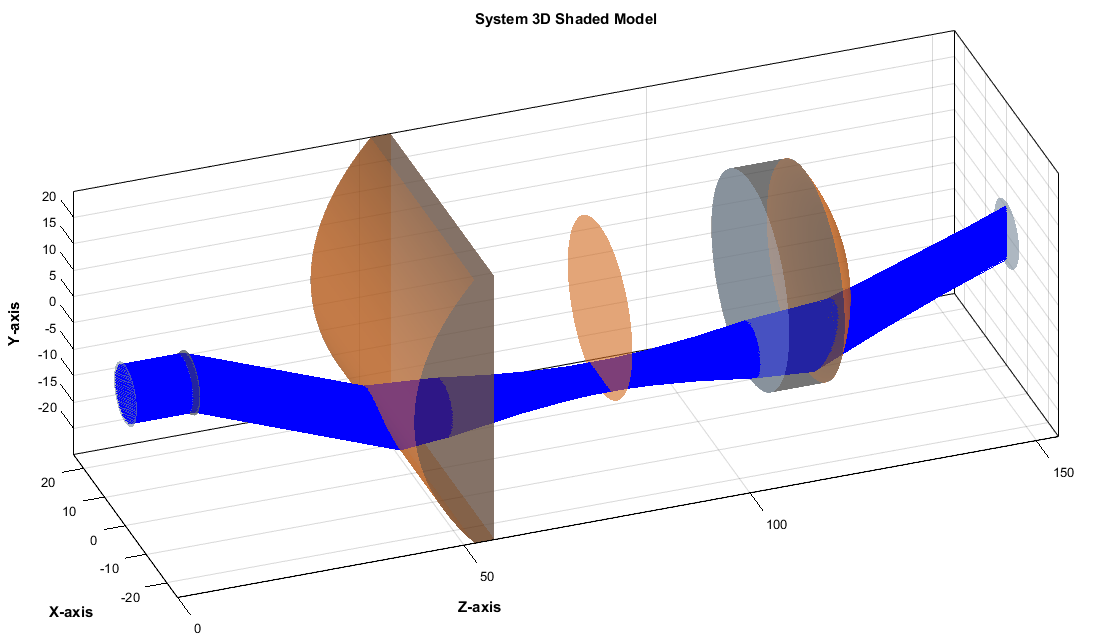
Then when the MatLightTracer is restarted the list of supported elements shown as popup menu in the optical element list panel should be updated and the new surface type would appear in the list.



And when the Toroidal surface is selected then the corresponding optical element parameter editor panel should show all properties of the new surface type including those stated as its unique parameters as shown below.



Finally here is an example of optical system containing the new surface type (cylindrical lens). All other features of the MatLightTracer toolbox works exactly in the same way for the new surface type as well.



*Created with the Personal Edition of HelpNDoc:* [*Full-featured EBook editor*](http://www.helpndoc.com/create-epub-ebooks)

**Transverse Ray Aberration Diagram**

**Toolbox Extension Example: Transverse Ray Aberration Diagram**

**Purpose:**

To add an analysis window to the toolbox which shows ray aberrations as a function of pupil coordinate.

*Created with the Personal Edition of HelpNDoc:* [*Free Kindle producer*](http://www.helpndoc.com/feature-tour/create-ebooks-for-amazon-kindle)

Procedures

**Procedures: Transverse Ray Aberration Diagram**

**Step 1: Define the Input and Outputs**

**Output:**

Graphs showing the transverse aberration (in X or Y) of sagittal or tangential rays with respect to the cheif ray at a given surface.

**Input:**

The analysis window requires the following setting parameters:

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Surface Index | The surface index used to compute the transverse aberration. |  |
| Field Index | Index of the field value for the analysis. | All fields can be used at the same time if a single wavelength is selected. |
| Wavelength Index | Index of the wavelength value for the analysis. | All wavelengths can be used at the same time if a single field point is selected. |
| Number of Rays | The number of rays to be traced per each field point and each wavelength. | Number less than 3 is not recommended. |
| Sagittal | Selects which aberration component to plot for the sagittal fan. | Since sagittal fans are  functions of the x pupil coordinate, the default is to plot the x component of the  aberration. |
| Tangential | Selects which aberration component to plot for the tangential fan. | Since tangential fans  are functions of the y pupil coordinate, the default is to plot the y component of the  aberration. |

**Step 2: Determine the Toolbox Modules to be Extended**

This feature requires extension of the following modules of the toolbox:

* **AODParentWindow Class:** 
  + A new menu item and/or toolbar button should be added. This can be done by editing the "InitializeMenuAndToolbarItems.m" file in the class folder. Here a sub menu item called "Ray Aberration" is added to the Optical System menu !! Here is the code added for the menu item addition with corresponding callbacks:

% Menu Item Definition

aodHandles.menuTransverseRayAberration = uimenu( ...

'Parent', aodHandles.menuOpticalSystemAnalysis, ...

'Tag', 'menuTransverseRayAberration', ...

'Label', 'Ray Aberration', ...

'Checked', 'off', ...

'Callback', {@menuTransverseRayAberration\_Callback,parentWindow});

% Menu Item Callback Definition

function menuTransverseRayAberration\_Callback(~,~,myParent)

AODChildWindow('transverseRayAberration',myParent);

end

* **AODChildWindow Class:** 
  + A new analysis window called "transverseTransverseRayAberration" should be added.
  + **Setting Window:** 
    - Most uicontrols (such as Surface Index, Field Index , ...) required for the new window are already defined in the toolbox as they are used for existing analysis windows. So they can be used directly with out redefining just by setting their position property. But the uicontrols to select tangetial and sagital aberration component are new to the toolbox and so should be defined first in the child window class.
    - Here the toolbox function "BuildChildWindowSettingPanel.m" is edited as follows:
      * The child windows are categorized in to different groups to simplify sharing of codes to design their graphical user interfaces. But the new window "transverseTransverseRayAberration" doesn't seems to fit in to any of the existing categories, so it is placed in t the "others" group.

% Uncategorized

others = ...

{ 'coatingRefractiveIndexProfile',...

'footprintDiagram',...

'system2DLayoutDiagram',...

'system3DLayoutDiagram',...

'paraxialAnalysis',...

'pupilApodization',...

'transverseRayAberration'};

% ------------------------------------------------------------

% This code was added just in the line just above the line saying "Add any new

% uicontrol definitions here"

% ------------------------------------------------------------

lblTangetialAberration = uicontrol( ...

'Tag', 'lblTangetialAberration', ...

'Style', 'text', ...

'HorizontalAlignment','Left',...

'FontSize',fontSize,'FontName', 'FixedWidth',...

'Visible','off',...

'String', 'Tangetial');

popTangetialAberration = uicontrol( ...

'Tag', 'popTangetialAberration', ...

'FontSize',fontSize,'FontName', 'FixedWidth',...

'Visible','off',...

'Style', 'popupmenu', ...

'BackgroundColor', [1 1 1], ...

'String', ['X Aberration','Y Aberration'],...

'Value',2);

% ------------------------------------------------------------

lblSagittalAberration = uicontrol( ...

'Tag', 'lblSagittalAberration', ...

'Style', 'text', ...

'HorizontalAlignment','Left',...

'FontSize',fontSize,'FontName', 'FixedWidth',...

'Visible','off',...

'String', 'Sagittal');

popSagittalAberration = uicontrol( ...

'Tag', 'popSagittalAberration', ...

'FontSize',fontSize,'FontName', 'FixedWidth',...

'Visible','off',...

'Style', 'popupmenu', ...

'BackgroundColor', [1 1 1], ...

'String', ['X Aberration','Y Aberration'],...

'Value',1);

%% Add any new uicontrol definitions here

% The following code was added inside the main switch case = 'others' to position the setting panel uicontrols for the % new analysis window

case lower('transverseRayAberration')

set(lblSurfaceIndex,...

'Parent', childHandle.panelMainTab(1), ...

'Units', 'normalized', ...

'Visible','on',...

'Position', [0.02 0.85 0.15 0.09]);

childHandle.lblSurfaceIndex = lblSurfaceIndex;

set(popSurfaceIndex,...

'Parent', childHandle.panelMainTab(1), ...

'Units', 'normalized', ...

'Visible','on',...

'Position', [0.22 0.88 0.25 0.07]);

childHandle.popSurfaceIndex = popSurfaceIndex;

childHandle.txtCoatingName = txtCoatingName;

set(lblWavelengthIndex,...

'Parent', childHandle.panelMainTab(1), ...

'Units', 'normalized', ...

'Visible','on',...

'Position', [0.53 0.85 0.20 0.09]);

childHandle.lblWavelengthIndex = lblWavelengthIndex;

set(popWavelengthIndex,...

'Parent', childHandle.panelMainTab(1), ...

'Units', 'normalized', ...

'Visible','on',...

'String',['All';get(popWavelengthIndex,'String')],...

'Position', [0.73 0.88 0.25 0.07]);

childHandle.popWavelengthIndex = popWavelengthIndex;

childHandle.txtWavelength = txtWavelength;

%------------------------------------------------------

set(lblFieldIndex,...

'Parent', childHandle.panelMainTab(1), ...

'Units', 'normalized', ...

'Visible','on',...

'Position', [0.02 0.75 0.20 0.09]);

childHandle.lblFieldIndex = lblFieldIndex;

set(popFieldIndex,...

'Parent', childHandle.panelMainTab(1), ...

'Units', 'normalized', ...

'Visible','on',...

'String',['All';get(popFieldIndex,'String')],...

'Position', [0.22 0.78 0.25 0.07]);

childHandle.popFieldIndex = popFieldIndex;

set(lblNumberOfRay,...

'Parent', childHandle.panelMainTab(1), ...

'Units', 'normalized', ...

'Visible','on',...

'Position', [0.53 0.75 0.20 0.09]);

childHandle.lblNumberOfRay = lblNumberOfRay;

set(txtNumberOfRay,...

'Parent', childHandle.panelMainTab(1), ...

'Units', 'normalized', ...

'Visible','on',...

'Position', [0.73 0.78 0.25 0.07]);

childHandle.txtNumberOfRay = txtNumberOfRay;

% ----------------------------------------------------

set(lblSagittalAberration,...

'Parent', childHandle.panelMainTab(1), ...

'Units', 'normalized', ...

'Visible','on',...

'Position', [0.02 0.65 0.20 0.09]);

childHandle.lblSagittalAberration = lblSagittalAberration;

set(popSagittalAberration,...

'Parent', childHandle.panelMainTab(1), ...

'Units', 'normalized', ...

'Visible','on',...

'Position', [0.22 0.68 0.25 0.07]);

childHandle.popSagittalAberration = popSagittalAberration;

% ----------------------------------------------------

set(lblTangetialAberration,...

'Parent', childHandle.panelMainTab(1), ...

'Units', 'normalized', ...

'Visible','on',...

'Position', [0.53 0.65 0.20 0.09]);

childHandle.lblTangetialAberration = lblTangetialAberration;

set(popTangetialAberration,...

'Parent', childHandle.panelMainTab(1), ...

'Units', 'normalized', ...

'Visible','on',...

'Position', [0.73 0.68 0.25 0.07]);

childHandle.popTangetialAberration = popTangetialAberration;

%% Add case statements for new windows in the

% "others" category here

* + **"Ok" Button:**
    - The "Ok" button should also be programmed in the child window class to collect and validate the user inputs from setting window and call a function to perform the analysis.
    - Here the toolbox function "btnOkCallback.m" is edited as follows:
      * The child windows are categorized in to different groups to simplify sharing of codes to collect data from the graphical user interfaces. But the new window "transverseTransverseRayAberration" doesn't seems to fit in to any of the existing categories, so it is placed in t the "others" group.

% Uncategorized

others = ...

{ 'coatingRefractiveIndexProfile',...

'footprintDiagram',...

'system2DLayoutDiagram',...

'system3DLayoutDiagram',...

'paraxialAnalysis',...

'pupilApodization',...

'transverseRayAberration'};

% The following code was added inside the main switch case = others

case lower('transverseRayAberration')

surfIndexList = (get(handles.popSurfaceIndex,'String'));

surfIndexString = surfIndexList(get(handles.popSurfaceIndex,'Value'));

surfIndex = str2double(surfIndexString);

if isnan(surfIndex)

disp('The surface index should be valid index number');

return;

end

numberOfRays = str2double(get(handles.txtNumberOfRay,'String'));

wavLengthIndexList = (get(handles.popWavelengthIndex,'String'));

wavLengthIndexString = (wavLengthIndexList(get(handles.popWavelengthIndex,'Value'),:));

if strcmpi(wavLengthIndexString,'New Wavelength')

elseif strcmpi(wavLengthIndexString,'All')

wavIndex = 1:1:currentOpticalSystem1.NumberOfWavelengths;

else

wavIndex = str2double(wavLengthIndexString);

end

fldIndexList = (get(handles.popFieldIndex,'String'));

fldIndexString = (fldIndexList(get(handles.popFieldIndex,'Value'),:));

if strcmpi(fldIndexString,'All')

fldIndex = 1:1:currentOpticalSystem1.NumberOfFieldPoints;

else

fldIndex = str2double(fldIndexString);

end

% Check condition that both field index and wavelength are

% not multiple at the same time. To avoid having multiple

% graphs, All fiedls can be analysed with specific

% wavelength and vice versa.

if strcmpi(fldIndexString,'All') && strcmpi(wavLengthIndexString,'All')

disp('Error:Both field index and wavelength index can not be "All" at the same time.');

return;

end

% Extract the wavelength and field point

wavLen = [(currentOpticalSystem1.WavelengthMatrix(wavIndex,1))'];

fieldPointXY = [(currentOpticalSystem1.FieldPointMatrix(fldIndex,1:2))'];

% Sagital and tangetial aberration components

sagAberrCompList = (get(handles.popSagittalAberration,'String'));

sagittalAberrComp = sagAberrCompList(get(handles.popSagittalAberration,'Value'));

tanAberrCompList = (get(handles.popTangentialAberration,'String'));

tangentialAberrComp = tanAberrCompList(get(handles.popTangentialAberration,'Value'));

% Since two graphs will be drawn (sagital and tangential),

% handles.panelMainTab(2) is passed instead of the axes

% handle.

currentOpticalSystem1.plotTransverseRayAberration(surfIndex,wavLen,...

fieldPointXY,numberOfRays,sagittalAberrComp,tangentialAberrComp,...

handles.panelMainTab(2));

* **OpticalSystem Class:**
  + A new method, called "plotTransverseRayAberration" should be added to the class to perform the analysis and plot the result.
  + The function source code is as shown below

function plotTransverseRayAberration(optSystem,surfIndex,wavLen,...

fieldPointXY,numberOfRays,sagittalAberrComp,tangentialAberrComp,...

plotPanelHandle)

% Displays the transverse ray aberration of sagital and tangetial ray fans

% on any surface with respect to the cheif ray.

% <<<<<<<<<<<<<<<<<<<<<<<<< Author Section >>>>>>>>>>>>>>>>>>>>>>>>>>>>

% Written By: Worku, Norman Girma

% Advisor: Prof. Herbert Gross

% Part of the RAYTRACE\_TOOLBOX V3.0 (OOP Version)

% Optical System Design and Simulation Research Group

% Institute of Applied Physics

% Friedrich-Schiller-University of Jena

% <<<<<<<<<<<<<<<<<<< Change History Section >>>>>>>>>>>>>>>>>>>>>>>>>>

% Date----------Modified By ---------Modification Detail--------Remark

% May 21,2014 Worku, Norman G. Original Version As example of extension

% <<<<<<<<<<<<<<<<<<<<< Main Code Section >>>>>>>>>>>>>>>>>>>>>>>>>>>>>

% Default Inputs

if nargin < 7

disp('Error: The function requires atleast 6 arguments, optSystem,',...

' surfIndex, wavLen, fieldPointXY, numberOfRays, sagittalAberrComp,',...

' and tangentialAberrComp.');

return;

elseif nargin == 7

axesHandle = axes('Parent',figure,'Units','normalized',...

'Position',[0.1,0.1,0.8,0.8]);

else

end

% Assign different symbals and colors for lines of d/t wavelengths

% and feild points respectively

availablelineColor = repmat(['b','k','r','g','c','m'],1,20); % 7\*20 = 140

% spotSymbal = availableSpotSymbal(1:size(fieldPointXY,2));

lineColorList = availablelineColor(1:size(wavLen,2)\*size(fieldPointXY,2));

%cla(axesHandle,'reset')

delete(allchild(plotPanelHandle));

JonesVec = [NaN;NaN];

% polarizedRayTracerResult = nSurf X nRay X nField X nWav

% pupil sampling = 5: Tangential Plane 6: Sagital Plane

PupSamplingTypeSagittal = 6;

PupSamplingTypeTangential = 5;

[sagittalRayTracerResult] = optSystem.multipleRayTracer(wavLen,...

fieldPointXY,numberOfRays,PupSamplingTypeSagittal,JonesVec);

[tangentialRayTracerResult] = optSystem.multipleRayTracer(wavLen,...

fieldPointXY,numberOfRays,PupSamplingTypeTangential,JonesVec);

nSurface = size(sagittalRayTracerResult,1);

nRay = size(sagittalRayTracerResult,2);

nField = size(sagittalRayTracerResult,3);

nWav = size(sagittalRayTracerResult,4);

% trace the cheif ray with either primary wavelength (for multiple

% wavelength analysis)or the specified wavelength (for single wavelegnth).

if nWav > 1 && nField > 1

disp('Error: Both field index and wavelength index can not be multiple at the same time.');

return;

elseif nWav > 1 && nField == 1

% Use the primary wavelength for the cheif ray

cheifRayWavLenInMet = optSystem.getPrimaryWavelength;

% Change wavlegth back to the system wavelength unit

wavUnitFactor = optSystem.getWavelengthUnitFactor;

cheifRayWavLen = cheifRayWavLenInMet/wavUnitFactor;

elseif nWav == 1

% Use the specified wavelength for the cheif ray

cheifRayWavLen = wavLen;

else

end

considerSurfAperture = 0;

% cheifRayTraceResult: nSurf X nField as each field point has different

% cheif rays

cheifRayTraceResult = optSystem.traceCheifRay(fieldPointXY,cheifRayWavLen,considerSurfAperture );

% Use different color for diffrent wavelengths and different field points.

for wavIndex = 1:nWav

for fieldIndex = 1:nField

lineIndex = fieldIndex + (wavIndex-1)\*nField;

cheifRayIntersection = [cheifRayTraceResult(surfIndex,fieldIndex).RayIntersectionPoint];

cheifRayIntersections = repmat(cheifRayIntersection,[1,nRay]);

sagittalFanIntersectionPoints = ...

[sagittalRayTracerResult(surfIndex,:,fieldIndex,wavIndex).RayIntersectionPoint];

tangentialFanIntersectionPoints = ...

[tangentialRayTracerResult(surfIndex,:,fieldIndex,wavIndex).RayIntersectionPoint];

if strcmpi(sagittalAberrComp,'X Aberration')

sagY(lineIndex,:) = sagittalFanIntersectionPoints(1,:) - cheifRayIntersections(1,:);

yLabelSag = 'EX';

elseif strcmpi(sagittalAberrComp,'Y Aberration')

sagY(lineIndex,:) = sagittalFanIntersectionPoints(2,:) - cheifRayIntersections(2,:);

yLabelSag = 'EY';

else

end

if strcmpi(tangentialAberrComp,'X Aberration')

tanY(lineIndex,:) = tangentialFanIntersectionPoints(1,:) - cheifRayIntersections(1,:);

yLabelTan = 'EX';

elseif strcmpi(tangentialAberrComp,'Y Aberration')

tanY(lineIndex,:) = tangentialFanIntersectionPoints(2,:) - cheifRayIntersections(2,:);

yLabelTan = 'EY';

else

end

xLabelTan = 'PY';

xLabelSag = 'PX';

sagX(lineIndex,:) = linspace(-1,1,size(sagY,2));

tanX(lineIndex,:) = linspace(-1,1,size(tanY,2));

legendText{lineIndex} = ['Field: [',num2str(fieldPointXY(1,fieldIndex)),',',...

num2str(fieldPointXY(2,fieldIndex)),']',...

' Wav: ',num2str(wavLen(wavIndex))];

end

end

% Generate two new panel for sagital and tangential fans.

tangentialPlotPanel = uipanel('Parent',plotPanelHandle,...

'Units','Normalized',...

'Position',[0.52,0.1,0.45,0.8],...

'Title',[char(tangentialAberrComp),' for Tangential Fan']);

tangentialPlotAxes = axes('Parent',tangentialPlotPanel,...

'Units','Normalized',...

'Position',[0.1,0.2,0.88,0.6]);

for tanKK = 1:lineIndex

currentLineColor = lineColorList(tanKK);

plot(tangentialPlotAxes,tanX(tanKK,:),tanY(tanKK,:),currentLineColor);

hold on;

end

grid on;

xlabel(tangentialPlotAxes,xLabelTan,'FontSize',12);

ylabel(tangentialPlotAxes,yLabelTan,'FontSize',12);

legend(tangentialPlotAxes,legendText)

sagittalPlotPanel = uipanel('Parent',plotPanelHandle,...

'Units','Normalized',...

'Position',[0.03,0.1,0.45,0.8],...

'Title',[char(sagittalAberrComp),' for Sagital Fan']);

sagittalPlotAxes = axes('Parent',sagittalPlotPanel,...

'Units','Normalized',...

'Position',[0.1,0.2,0.88,0.6]);

for sagKK = 1:lineIndex

currentLineColor = lineColorList(sagKK);

plot(sagittalPlotAxes,sagX(sagKK,:),sagY(sagKK,:),currentLineColor);

hold on;

end

grid on;

xlabel(sagittalPlotAxes,xLabelSag,'FontSize',12);

ylabel(sagittalPlotAxes,yLabelSag,'FontSize',12);

legend(sagittalPlotAxes,legendText)

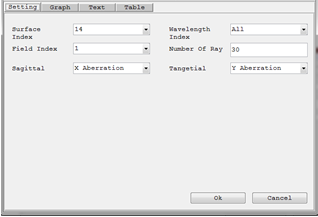
end

*Created with the Personal Edition of HelpNDoc:* [*Free EBook and documentation generator*](http://www.helpndoc.com)

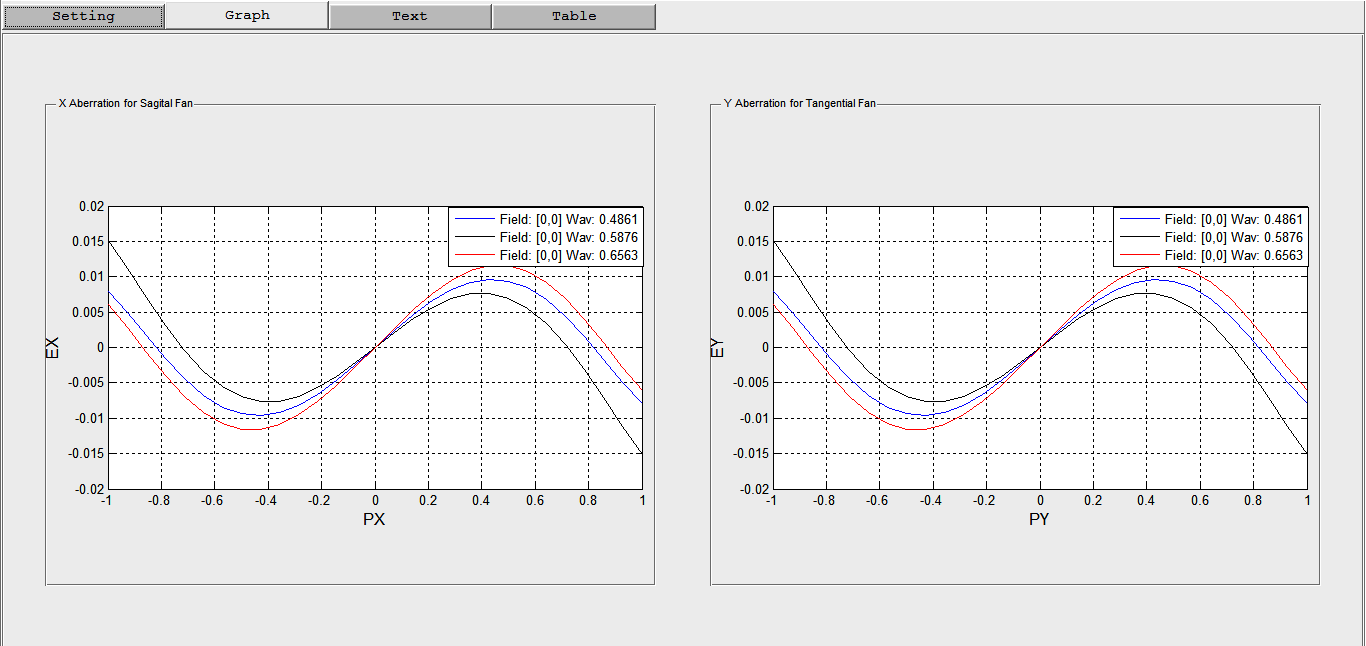
Results

**Results: Transverse Ray Aberration Diagram**

**Setting Window:**



**Graphical Ray Aberration Diagram Window:**



*Created with the Personal Edition of HelpNDoc:* [*Full-featured multi-format Help generator*](http://www.helpndoc.com/help-authoring-tool)

**Longitudinal Ray Aberration Diagram**

**Toolbox Extension Example: Longitudinal Ray Aberration Diagram**

**Purpose:**

To add an analysis window to the toolbox which displays the longitudinal aberration as a function of pupil height at each wavelength. This feature computes the distance from the image surface to where a zonal marginal ray "focuses", or crosses the optical axis. The computation is performed only for the on axis field point, and only for zonal marginal tangential rays as a function of pupil zone. The base of the plot is on axis, and the top of the plot represents the maximum entrance pupil radius. There are no units on the vertical scale because the plot is always normalized to the maximum entrance pupil radius. The horizontal scale is in lens units, and represents the distance from the image surface to the point where the ray crosses the optical axis

*Created with the Personal Edition of HelpNDoc:* [*Full-featured EPub generator*](http://www.helpndoc.com/create-epub-ebooks)

Procedures

**Procedures: Longitudinal Ray Aberration Diagram**

**Step 1: Define the Input and Outputs**

**Output:**

Graphs showing the longitudinal aberration.

**Input:**

The analysis window requires the following setting parameters:

|  |  |  |
| --- | --- | --- |
| **Settings** | **Description** | **Remarks** |
| Wavelength Index | Index of the wavelength value for the analysis. | All wavelengths can be used at the same time. |
| Number of Rays | The number of rays to be traced per each field point and each wavelength. | Number less than 3 is not recommended. |

**Step 2: Design a graphical user interface to take inputs from the user and show the result**

For this case the input can be taken with matlab dialog window. And the resulting plot can be shown in a matlab figure window.

**Step 3: Write the main function**

A new function called "plotLongitudinalAberration.m" was written as follows and placed in the folder "Toolbox\_Extensions" together with its test script:

function [ success ] = plotLongitudinalAberration(opticalSystem)

%PLOTLONGITUDINALABERRATION Plots the longitudinal aberration of the

%optical system.

% User inputs the wavelength indices and number of ray

wavIndices\_nRay = inputdlg({'Wavelength Index (0 for All)',...

'Number of Rays',},...

'Wavelength Index and Number of Rays',1,{'0','100'});

if ~(isempty(wavIndices\_nRay)||...

isempty(wavIndices\_nRay{1})||...

isempty(wavIndices\_nRay{2}))

wavLenIndex = str2double(wavIndices\_nRay{1});

nRay = str2double(wavIndices\_nRay{2});

nWavelength = opticalSystem.NumberOfWavelengths;

if isnan(nRay)||isnan(wavLenIndex) || ...

nRay < 1 || wavLenIndex < 0 || wavLenIndex > nWavelength

disp('Error: All Input should be numeric and valid');

success = 0;

return;

end

else

disp('Error: Invalid Input.');

success = 0;

return;

end

fieldPointXY = [0;0];

wavelengthMatrix = opticalSystem.WavelengthMatrix;

if wavLenIndex == 0

wavLen = (wavelengthMatrix(:,1))';

else

wavLen = wavelengthMatrix(wavLenIndex,1);

end

JonesVec = [NaN;NaN];

nWav = size(wavLen,2);

nField = size(fieldPointXY,2);

% trace the tangntial rays

% polarizedRayTracerResult = nSurf X nRay X nField X nWav

% pupil sampling = 5: Tangential Plane

PupSamplingTypeTangential = 5;

[tangentialRayTracerResult] = opticalSystem.multipleRayTracer(wavLen,...

fieldPointXY,nRay,PupSamplingTypeTangential,JonesVec);

% Assign different symbals and colors for lines of d/t wavelengths

availablelineColor = repmat(['b','k','r','g','c','m','y'],1,20); % 7\*20 = 140

lineColorList = availablelineColor(1:nWav\*nField);

surfIndex = opticalSystem.NumberOfSurfaces;

fieldIndex = 1;

figure;

for wavIndex = 1:nWav

tangentialFanIntersectionPoints = ...

[tangentialRayTracerResult(surfIndex,:,fieldIndex,wavIndex).RayIntersectionPoint];

tangentialFanFinalDirections = ...

[tangentialRayTracerResult(surfIndex,:,fieldIndex,wavIndex).IncidentRayDirection];

opticalAxisIntersectionZ = (tangentialFanIntersectionPoints(2,:))./...

(-tan(acos(tangentialFanFinalDirections(3,:))));

% NaN will result for rays alogn the axis so take the value of the

% immediate upper field point in this case

opticalAxisIntersectionZ(find(isnan(opticalAxisIntersectionZ))) = ...

opticalAxisIntersectionZ(find(isnan(opticalAxisIntersectionZ))-1);

xAxisPoints = opticalAxisIntersectionZ;

yAxisPoints = linspace(1,-1,size(opticalAxisIntersectionZ,2));

% Only take the upper part of the enterance pupil

xAxisPointsUpper = xAxisPoints(yAxisPoints >= 0);

yAxisPointsUpper = yAxisPoints(yAxisPoints >= 0);

legendText{wavIndex} = [' Wav: ',num2str(wavLen(wavIndex))];

currentLineColor = lineColorList(wavIndex);

plot(xAxisPointsUpper,yAxisPointsUpper,currentLineColor);

grid on;

hold on;

end

axis equal;

xlabel('Z','FontSize',12);

ylabel('PY (Normalized)','FontSize',12);

legend(legendText)

success = 1;

end

*Created with the Personal Edition of HelpNDoc:* [*Easily create PDF Help documents*](http://www.helpndoc.com/feature-tour)

Results

**Results: Longitudinal Ray Aberration Diagram**

As the new feature is not integrated in to the main toolbox, it can not be used from the GUI of the toolbox, rather it can be accessed only from command window writing a matlab script. To test the new longitudinal aberration diagram, the following test script was written;

% Test script for plotLongitudinalAberration.m function

% Read the double gauss system

% Open saved optical system

% Get path of the single lens system used for testing

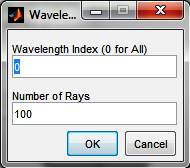
doubleGauss28 = which('DoubleGauss28.mat');

% Construct the optical system object from the saved file

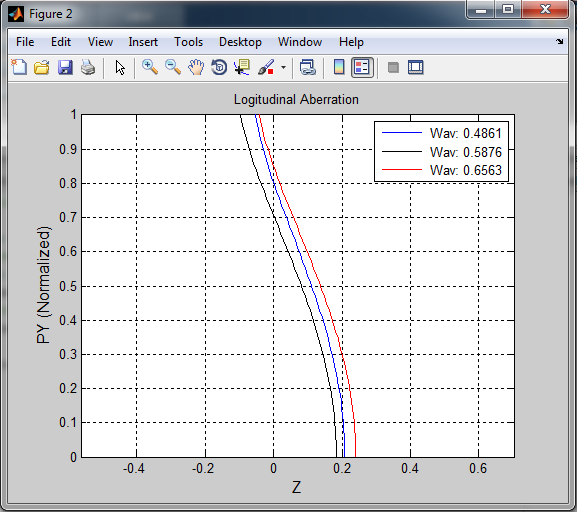
OS = OpticalSystem(doubleGauss28);

plotLongitudinalAberration(OS);

**Setting Window:**



**Graphical Longitudinal Ray Aberration Diagram Window:**



*Created with the Personal Edition of HelpNDoc:* [*Create HTML Help, DOC, PDF and print manuals from 1 single source*](http://www.helpndoc.com/help-authoring-tool)