Programming and Algorithms Reference MATLAB Based Optical Analysis Toolbox V 3.0

**Table of contents**

[Cover Page](#_topic_CoverPage) 3

[Objective](#_topic_Objective) 6

[General Toolbox Structure](#_topic_GeneralToolboxStructure) 9

[Class Definition Module](#_topic_ClassDefinitionModule) 12

[GUI Related Class](#_topic_GUIRelatedClass) 15

[AODParentWindow: Class](#_topic_AODParentWindowClass) 18

[Surface Editor Panel](#_topic_SurfaceEditorPanel) 22

[System Configuration Panel](#_topic_SystemConfigurationPanel) 30

[Menu Items](#_topic_MenuItems) 35

[AODChildWindow: Class](#_topic_AODChildWindowClass) 39

[Non GUI Related Class](#_topic_NonGUIRelatedClass) 43

[OpticalSystem: Class](#_topic_OpticalSystemClass) 46

[Surface: Class](#_topic_SurfaceClass) 52

[Glass: Class](#_topic_GlassClass) 58

[Coating: Class](#_topic_CoatingClass) 61

[Ray: Class](#_topic_RayClass) 65

[RayTraceResult: Class](#_topic_RayTraceResultClass) 68

[Scalar Ray Tracing Module](#_topic_ScalarRayTracingModule) 71

[Coordinate Transformations](#_topic_CoordinateTransformations) 74

[Calculation of the Path Length](#_topic_CalculationofthePathLength) 77

[Computation of the Intersection Point](#_topic_ComputationoftheIntersectionPoin) 81

[Surface Normal and Incidence Angle Computation](#_topic_SurfaceNormalandIncidenceAngleCo) 84

[Refraction (or Reflection)](#_topic_RefractionorReflection) 87

[Paraxial and Meridional Ray Tracing Module](#_topic_ParaxialandMeridionalRayTracingM) 90

[Paraxial Ray Tracing Module](#_topic_ParaxialRayTracingModule) 93

[Meridional Ray Tracing Module](#_topic_MeridionalRayTracingModule) 97

[Polarization Ray Tracing Module](#_topic_PolarizationRayTracingModule) 101

[Jones Matrix to Polarization Ray Tracing Matrix](#_topic_JonesMatrixtoPolarizationRayTrac) 107

[Polarization Ray Tracing through Interfaces](#_topic_PolarizationRayTracingthroughInt) 111

[Polarization Properties of Optical System](#_topic_PolarizationPropertiesofOpticalS) 116

[Examples of Extending the Toolbox](#_topic_ExamplesofExtendingtheToolbox) 121

[Method 1: As part of the toolbox](#_topic_Method1Aspartofthetoolbox) 124

[Transverse Ray Aberration Diagram](#_topic_TransverseRayAberrationDiagram) 127

[Procedures](#_topic_Procedures) 130

[Results](#_topic_Results) 140

[Method 2: As separate tool using the existing toolbox](#_topic_Method2Asseparatetoolusingtheexi) 143

[Longitudinal Ray Aberration Diagram](#_topic_LongitudinalRayAberrationDiagram) 146

[Procedures](#_topic_Procedures1) 149

[Results](#_topic_Results1) 153

**Cover Page**

MATLAB Based Optical System Analyzer Toolbox

Programming and Algorithms Reference Version 3.0

Optical System Design Research Group

Institute of Applied Physics

Friedrich Schiller University of Jena

05/23/2014

Norman G. Worku

Advisor: Prof. Dr. Herbert Gross

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

**Objective**

# Objective

Optical System Analyzer Toolbox is a GUI based optical ray tracing toolbox in MATLAB, which is being designed by optical system design and simulation research group, Institute of Applied Physics, Friedrich Schiller University of Jena.

The objective of this 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>"**. They shall be recorded and included in the next versions.

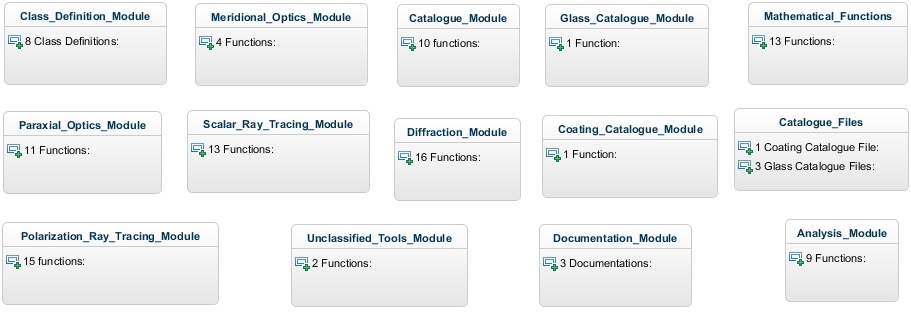
*Created with the Personal Edition of HelpNDoc:* [*Produce electronic books easily*](http://www.helpndoc.com/create-epub-ebooks)

**General Toolbox Structure**

# General Toolbox Structure

The toolbox is written using object oriented programming approach. So the main part of the toolbox will be the class definitions with all member properties and methods. The following is the high level class diagram showing the classes used in the toolbox together with their interconnections.

Here is the block diagram showing all modules and number of functions in each module of the toolbox.



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

**Class Definition Module**

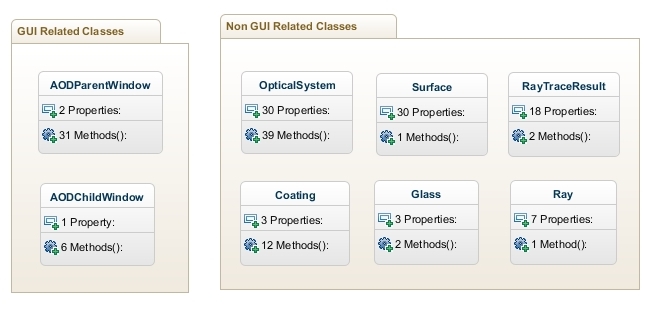
# Class Definition Module

It is a module of the toolbox where all the class definitions are placed. Each class is defined with its own properties and methods. The properties are all listed inside the class definition file and all the methods are placed inside a folder named "@Class\_Name". Defining the methods in a separate file reduces the size of the main class definition file and makes it more easy to read.

The toolbox has two main categories of classes :

1. GUI Related
2. Non GUI Related

Here are the classes with the number of properties and methods they include.



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

**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:* [*Easily create EBooks*](http://www.helpndoc.com/feature-tour)

**AODParentWindow: Class**

# AODParentWindow Class

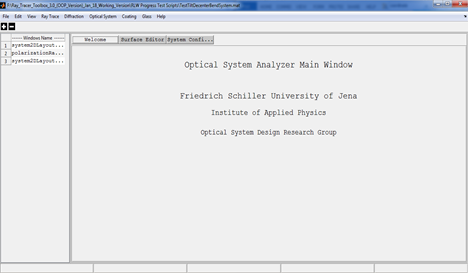
This class is used to define the main graphical user interface window which allows setting and analyzing optical systems using the toolbox.

Number of Properties: 2

Number of Methods: 31

**Windows Description**

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



The main window has the following main parts:

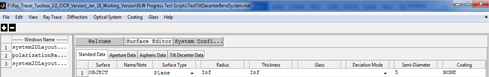
* **Title Bar:**
  + The full path name of the current optical system is displayed.
* **Menu Bar:**
  + It contains Menu items which are used to use different features of the toolbox.
  + For detailed description see the "Menu Bar" section below.
* **Tool Bar:**
  + Contains buttons for frequently used features of the toolbox. It acts as shortcuts to the menu items.
* **Opened Window List Panel:**
  + It is panel to the left side of the window and displays the list of all currently opened child windows. This allows access of the child windows just a click away in case of working with multiple child windows.
* **Main Panel:**
  + It is the largest part of the main window where the following panels are displayed:
    - **Welcome Panel:** Display welcome message.
    - **Surface Editor Panel:** Contains tabbed surface data editors.
    - **System Configuration Panel:** Has tabbed windows for inputting system configurations.
* **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:* [*Full-featured multi-format Help generator*](http://www.helpndoc.com/help-authoring-tool)

Surface Editor Panel

**Surface Editor Panel**

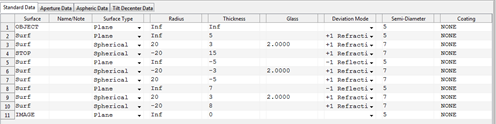
It is part of the main panel of the parent window and contains spreadsheet like data editors for entering and editing the surface data of an optical system.



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

**1. Standard Data**

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



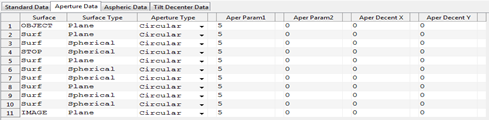
|  |  |  |
| --- | --- | --- |
| **Column Header** | **Description** | **Remarks** |
| Surface | Used to indicate the surface index or whether the surface is object or, image or stop surface or not. | Object and image surfaces are always the first and the last surfaces of the system.  To make a surface a stop, type “s” or “stop” in the surface column.  Any text inputs other than “s” or “stop” will be rejected by the system |
| Name/Comment | To enter name and notes related to the surface. | Any text inputs are accepted and it has no real functional significance. |
| Surface Type | A pop up menu to choose the type of surface. | Currently Plane, Spherical, and Conic Aspherical surfaces are functional. Other surface types are to be included in the future. |
| 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. |
| 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. |
| Deviation Mode | Pop up menu used to indicate the surface as reflective (-1) or refractive (+1). |  |
| Semi diameter | Define the semi diameter of the surface. | This is linked with the surface aperture definition spreadsheet. If the surface aperture type is “None” and some value is entered for semi diameter then, the aperture type will automatically be changed to circular with radius equal to the semi diameter entered. But if the surface has its aperture defined, then the semi diameter will only be used to plot the surface and not for aperture calculations. |
| 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. |

Notes:

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**

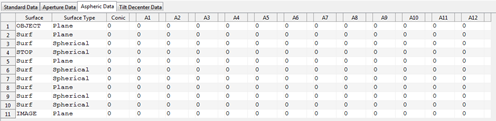
It contains data defining surface apertures.



|  |  |  |
| --- | --- | --- |
| **Column Header** | **Description** | **Remarks** |
| Surface | See “standard data” sheet | Just repeated from standard data entry sheet for surface identification. |
| Surface Type | See “standard data” sheet | Just repeated from standard data entry sheet for surface identification. |
| Aperture Type | Pop up menu to select the type of aperture used for the surface | Currently “Circular” , "Elliptical"and “Rectangular” aperture shapes are functional. Others are left for future versions. |
| Aperture Param 1 | Aperture Parameter | Only positive numerical values are allowed. It has different purpose for different aperture types.  Radius for circular aperture and x side length for rectangular. |
| Aperture Param 2 | Aperture Parameter | Only positive numerical values are allowed. Not applied for circular and y side length for rectangular. |
| Aper Decenter X | Aperture Decenter in X | Not functional |
| Aper Decenter Y | Aperture Decenter in Y | Not functional |

**3. Aspheric Data**

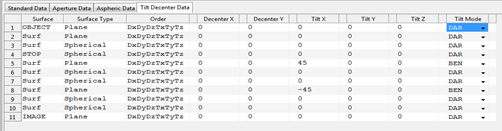
It contains data defining aspherical surfaces such as aspheric constants and polynomial coefficients.



|  |  |  |
| --- | --- | --- |
| **Column Header** | **Description** | **Remarks** |
| Surface | See “standard data” sheet | Just repeated from standard data entry sheet for surface identification. |
| Surface Type | See “standard data” sheet | Just repeated from standard data entry sheet for surface identification. |
| Conic | To enter the conic constant of the surface. | This is used for surface types indicated to be aspherical in the standard data entry. If the surface type is not aspheric then the conic constant will automatically be 0. |
| A1…A12 | Polynomial Coefficients | Not functional |

**4. Tilt and Decenter Data**

It includes tilt and decenter parameters of the surfaces.



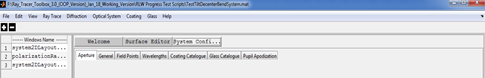
|  |  |  |
| --- | --- | --- |
| **Column Header** | **Description** | **Remarks** |
| Surface | See “standard data” sheet | Just repeated from standard data entry sheet for surface identification. |
| Surface Type | See “standard data” sheet | Just repeated from standard data entry sheet for surface identification. |
| Order | Order in which tilt and decenter are performed. | All the tilt and decenter operations shall be listed otherwise it will be invalid. |
| 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. |
| Tilt Mode | Pop up menu for selecting the mode of the tilt determining the reference coordinate axis after the current surface. |  |

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

System Configuration Panel

**System Configuration Panel**

It is part of the main panel of the parent window and 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. 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.



**2. 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.



**3. Field Points**

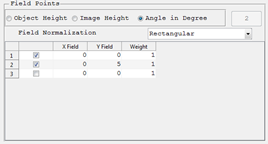
To define set of field points, which can later be used to analyze the system. Field points can be defined either by specifying the object height or by ray angles. If object heights are used to define the field points, the heights are measured in lens units. Field angles are always in degrees. The angles 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

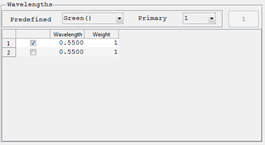
l2+m2+n2 = 1

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



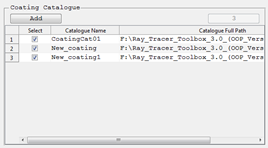
**4. Wavelengths**

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.



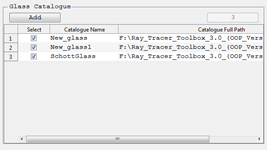
**5. Coating Catalogue**

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.



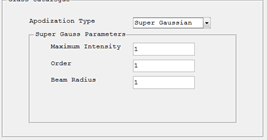
**6. Glass Catalogue**

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:* [*Single source CHM, PDF, DOC and HTML Help creation*](http://www.helpndoc.com/help-authoring-tool)

Menu Items

**Menu Items**

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:** 
  + **Surface Editor:** Opens the “Surface Editor” panel for current optical system.
  + **System Configuration:** Opens the “Optical System Configuration” panel.
* **Ray Tracing:** 
  + **Scalar Ray Trace:** Opens single ray data entry window which enables tracing of a single ray through current optical system.
  + **Polarization Ray trace:** Opens single polarized ray data entry window which enables polarization ray tracing of a single ray through current optical system.
* **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:**
  + **Optical System Analysis:**
    - **Footprint Diagram:** Plots footprint diagram of a bundle of ray on a given surface of an optical system.
    - **Polarization Ellipse Map:** Shows distribution of polarization ellipse over the pupil area for a given polarized ray at a given surface.
    - **Paraxial Analysis:** Performs paraxial analysis and displays the result in the command window.
    - **Polarization Aberration:** Displays different graphs showing the polarization aberration of the optical system.
* **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.
* **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.

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

**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: 1

Number of Methods: 6

**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.

**Windows Description**

Child 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:* [*Free HTML Help documentation generator*](http://www.helpndoc.com)

**Non GUI Related Class**

# Non GUI Related Class Definitions

These are classes that are used for the functions of the toolbox which are not related to the graphical user interfaces directly. There are six non GUI related classes used in the toolbox.

* 1. OpticalSystem:
  2. Surface:
  3. Glass:
  4. Coating:
  5. Ray:
  6. RayTraceResult:

**Design Philosophy**

* The class definitions include the properties and a class constructor method only. All other class methods are defined in separate files and put in common folder named "@ClassName".

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

**OpticalSystem: Class**

# OpticalSystem Class

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

Number of Properties: 30

Number of Methods: 39

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

**Important Methods**

**1. Main Ray Tracing Function**

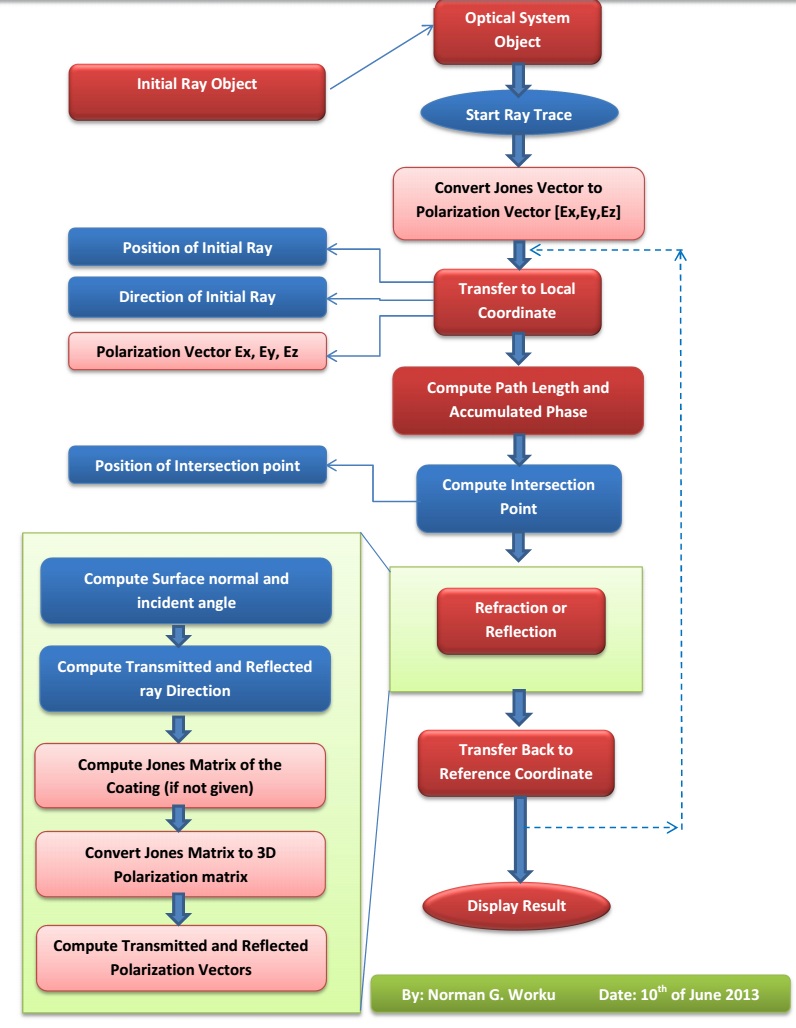
**Syntax:**

**Function Call:**

polarizedRayTracerResult = optSystem.tracePolarizedRay(objectRay,considerSurfAperture)

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:**



To trace N rays through the system:

* First construct the updated OpticalSystem object from the surface and configuration data in the parent window.
* Then construct 1xN sized array of Ray objects indicating the initial rays to be traced. For unpolarized rays the Jones vector should be [NaN;NaN].
* Give the OpticalSystem object and the array of Ray objects to the "tracePolarizedRay" function. It will return a 2D matrix of the RayTraceResult object with size [nSurface x N].

**Note 1:** For single ray object the "tracePolarizedRay" returns a vector (size = nSurface) of RayTraceResult object with each element corresponding to the ray properties at each surface of the optical system starting from Object surface to Image surface. And for N ray objects, the result will be a a 2D matrix of the RayTraceResult object with size [nSurface x N].

**Note 2:** 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:**

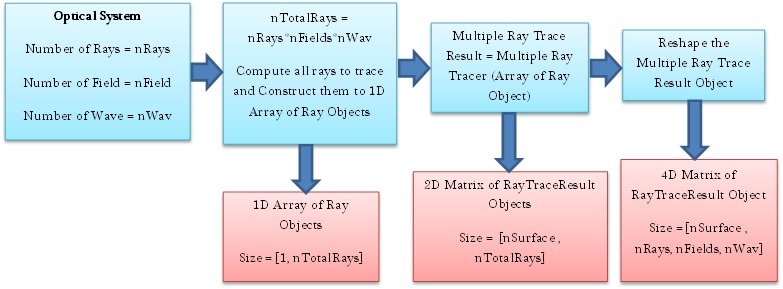
**Function Call:**

[multipleRayTracerResult,pupilCoordinates,pupilGridIndices] = ...

optSystem.multipleRayTracer(wavLen,...

fieldPointXY,nRay,PupSamplingType,JonesVec,considerSurfAperture)

**Flow Chart:**



To trace multiple rays, all the inputs for all rays to be traced should be given as input. If N rays are to be traced, the sizes of the inputs required:

* + - [1xN] wavLen,
    - [2xN] fieldPointXY,
    - [1] nRay,
    - [1] PupSamplingType,
    - [2xN] JonesVec, and
    - [1] considerSurfAperture.

A vector of ray objects are first constructed and traced through the system. The total number of ray will be nRay\*nWav\*nField, that is all rays from each field point with each of wavelengths will be traced. The results are restructured to a 4 dimensional matrix [nSurf x nRay x nField x nWav] so that accessing the results for specific field and wavelength is simplified. The final reshaped result makes access of the results for specific field and wavelength very simple.

For instance, 4DRayTraceResult (surfIndex, : , fieldIndex, wavIndex): directly gives the ray trace results for all the rays from field point fieldIndex and with wavelength index wavIndex at surface surfIndex.

**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:* [*Free Web Help generator*](http://www.helpndoc.com)

**Surface: Class**

# Surface Class

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

Number of Properties: 30

Number of Methods: 1

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

**Important Methods**

**1. Surface Coordinate Transformation Matrix Computation**

**Syntax:**

[surfaceCoordinateTM,nextReferenceCoordinateTM] = surf.TiltAndDecenter (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:

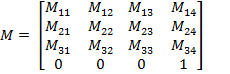
* **DAR (Decenter and Return):** The new reference coordinate after the tilted/decentered surface will return back to that before the surface. The tilt and decenter (X and Y but not Z) operation will be reversed and the coordinate becomes that before the tilt and decentered operation is done but moved 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:

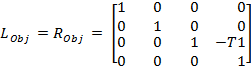


The 3x3 matrix in the upper left side of the transformation matrix corresponds to the total rotation matrix and the last column represents the total decenter vector of the surface local coordinate with respect to the global coordinate. The last row is not necessary but is added just to simplify matrix 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:* [*Free EBook and documentation generator*](http://www.helpndoc.com)

**Glass: Class**

# Glass Class

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

Number of Properties: 3

Number of Methods: 2

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

**Important Methods**

**1. Abbe Number Computation**

**Syntax:**

[abbeNumber] = Glass.getAbbeNumber(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:

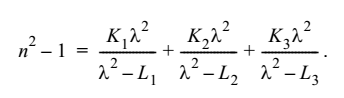
abbeNumber = (nd-1)/(nf-nc)

**2. Refractive Index Computation**

**Syntax:**

[n] = Glass.getRefractiveIndex(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:* [*Free EPub producer*](http://www.helpndoc.com/create-epub-ebooks)

**Coating: Class**

# Coating Class

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

Number of Properties: 3

Number of Methods: 12

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

**Important Methods**

**1. Transmission and Reflection Coefficient Calculations**

**Syntax:**

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

(wavLenInUm,incAngle,ns,nc,primaryWaveLenInUm)

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

(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 ] = coating.getRefractiveIndexThicknessTable(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:**

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

angleStep,primWavLenInUm,indexBefore,indexAfter,...

axesHandle,tableHandle,textHandle)

coating.plotCoatingReflectionVsWavelength(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:* [*Easy to use tool to create HTML Help files and Help web sites*](http://www.helpndoc.com/help-authoring-tool)

**Ray: Class**

# Ray Class

This class is used to define polarized ray objects, which contains all informations related with the ray defined.

Number of Properties: 7

Number of Methods: 1

The class supports constructors to construct an array of Ray objects from array of its properties.

**Important Methods**

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

**RayTraceResult: Class**

# RayTraceResult Class

This class is used to store ray trace results in the form of arrays.

Number of Properties: 18

Number of Methods: 2

Constructors can construct multiple RayTraceResult objects for multiple ray trace.

**Important Methods**

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

**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 class. Here the specific algorithms and mathematical formulations used in the ray tracing are described.

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

**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 class), 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.

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

**Calculation of the Path Length**

# 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, s0 = 0; thus p= s0 .

But when the surface is not a plane and the routine of the surface is expressed as:



**Spherical surface:**



And if we try to solve the equation of the surface to get the coordinate z given as a function of c, x, and y. The solution would be:



And we find that the solution that we need is as follows and only the negative sign is taken into consideration:



Therefore let the coordinates of P be (X, Y, Z) and the length of the segment from (X1, Y1, 0) to (X, Y, Z) is , we have







P lies on the surface, and therefore the equations could be put in the form



And we obtain a quadratic equation for  as:



Or



Thus, the solution of the quadratic is:



In which





Considering the special case that the ray goes through the vertex of the surface, we can see that the negative sign is taken into consideration just like the solution we need of the general surface equation. Also in general  is much less than. The result is



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:**



And similar with the solution of the spherical surface, the conic surface that we need only takes the negative sign into consideration:



The result will be:



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

**Computation of the Intersection Point**

# Computation of the Intersection Point

The routine to calculate the intersection point (after we have got the path length of the ray) is based on the formulas like







Here

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

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

**Surface Normal and Incidence Angle Computation**

# Surface Normal and Incidence Angle 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.

1. When the surface is sphere,

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.

1. When the surface is conic,

The unit normal vector of the aspherical surface:

Here we only use the expression of the conic section:



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

This is concluded according to the normal vector of spherical surface:



The **incident angle** should also be calculated before refraction. Local incident angle between the ray and the normal vector of the surface of given direction cosines is calculated. Since the coordinates are rotational symmetrical, the angle is always positive.

Direction cosines of the incident ray at the surface j 

Local unit normal vector at the surface j 

The incident angle is

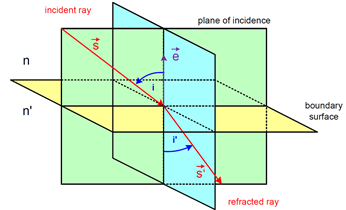


*Created with the Personal Edition of HelpNDoc:* [*Easily create EBooks*](http://www.helpndoc.com/feature-tour)

**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:* [*Free CHM Help documentation generator*](http://www.helpndoc.com)

**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.

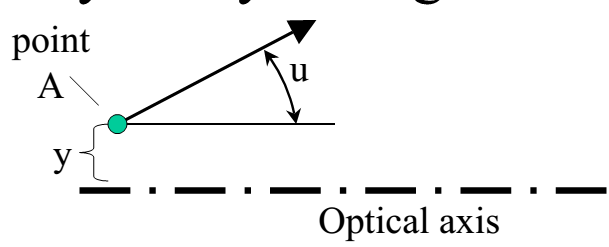
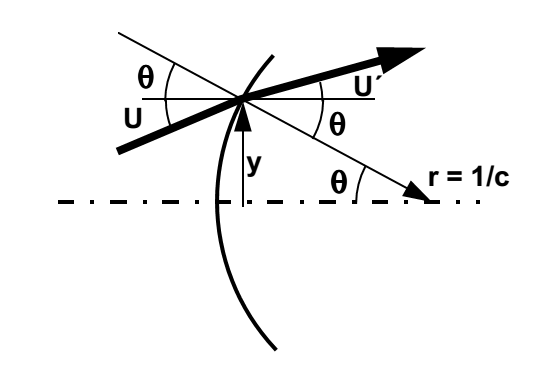
Both paraxial yni and meridional QU ray trace functions are written in such a way that both forward and backward ray tracing is possible.

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

**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:* [*Free help authoring environment*](http://www.helpndoc.com/help-authoring-tool)

**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.

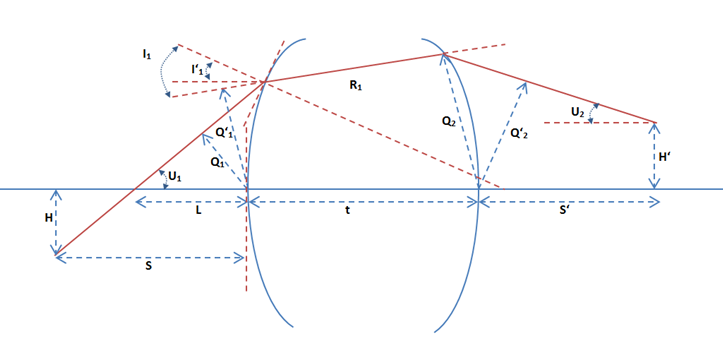


Figure above shows important parameters used in the QU algorithm

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

**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**

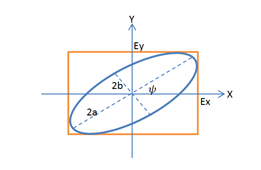
Due to transversal nature of fields (no component along the direction of propagation), any field can be represented by a complex two component vector known as Jones Vector. The polarization state of a polarized field propagating along z–axis is represented by



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

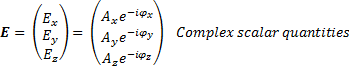


For a sequence of multiple optical elements, the effective Jones matrix can be determined by multiplying individual Jones matrices in reverse order as given in the following equation:



# 2. Three Dimensional Ray Tracing Calculus

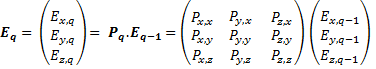
In three dimensional ray tracing calculus, the polarization states of a field are represented as three element electric field vectors. They are simply vectors containing the components of the electric field in the global coordinate system.





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:* [*Free HTML Help documentation generator*](http://www.helpndoc.com)

**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.

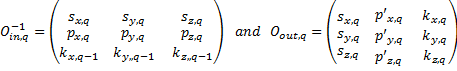
The local coordinates before and after the interface are related as follows. The reflected and transmitted rays kq both lie on the same plane which contains the incident ray kq-1. The p (p for “parallel”) components of the local coordinates also lie in this plane by definition. As the s (s for “senkrecht” 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:* [*Full-featured Kindle eBooks generator*](http://www.helpndoc.com/feature-tour/create-ebooks-for-amazon-kindle)

**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).

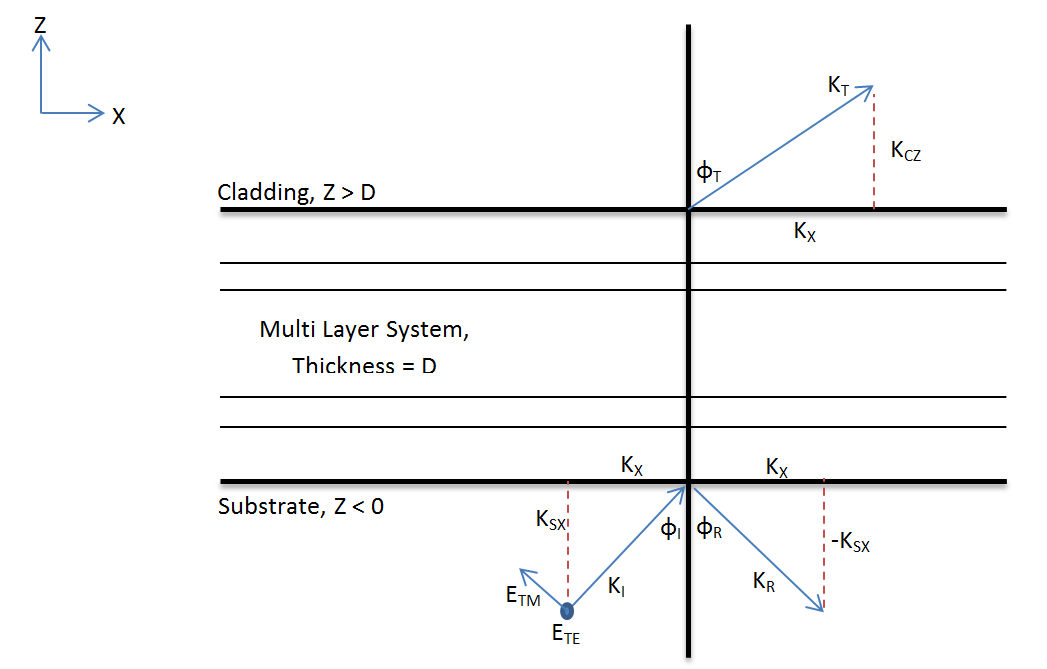


Where the  are the amplitude transmission (reflection) coefficients for s or p - components of the polarized field. For single plane interfaces, the coefficients can be calculated from the Fresnel’s equations and for layered interfaces matrix method for stratified media, which leads to the general form 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



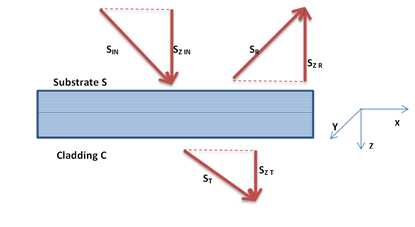






The coefficients computed above describe the relation between the complex amplitudes of incident, reflected and transmitted plane waves. They are called generalized Fresnel’s coefficients.

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:* [*Single source CHM, PDF, DOC and HTML Help creation*](http://www.helpndoc.com/help-authoring-tool)

**Polarization Properties of Optical System**

# Analysis of Polarization Properties in Three Dimensional Ray Tracing Calculus

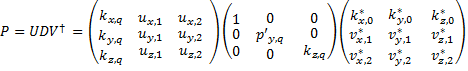
One of the purposes of polarization ray tracing is determination of the diattenuation and retardance associated with ray paths through an optical system. It can also be used to compute the polarization aberration function of an optical system. In this section those polarization properties will be defined 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) .

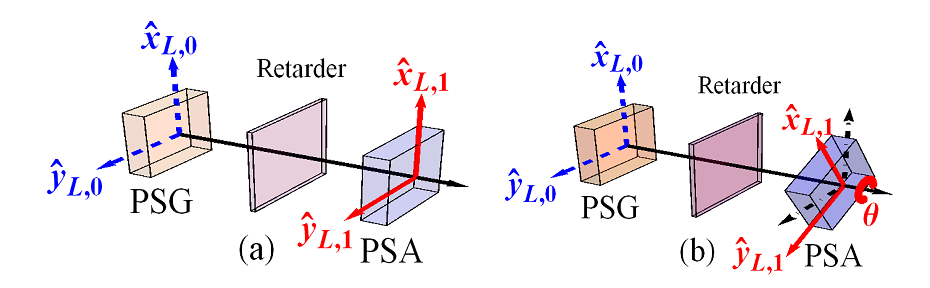


By definition the diagonal elements of D matrix are the singular values of P. The first column of V is incident wave vector K0. The other two columns of V, v1 and v2, are two special polarization vectors in the incident transverse plane that generate the maximum and minimum transmitted flux. Similarly, 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:* [*Free EBook and documentation generator*](http://www.helpndoc.com)

**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:* [*Easily create iPhone documentation*](http://www.helpndoc.com/feature-tour/iphone-website-generation)

**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.

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

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

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)

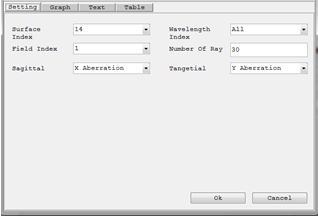
end

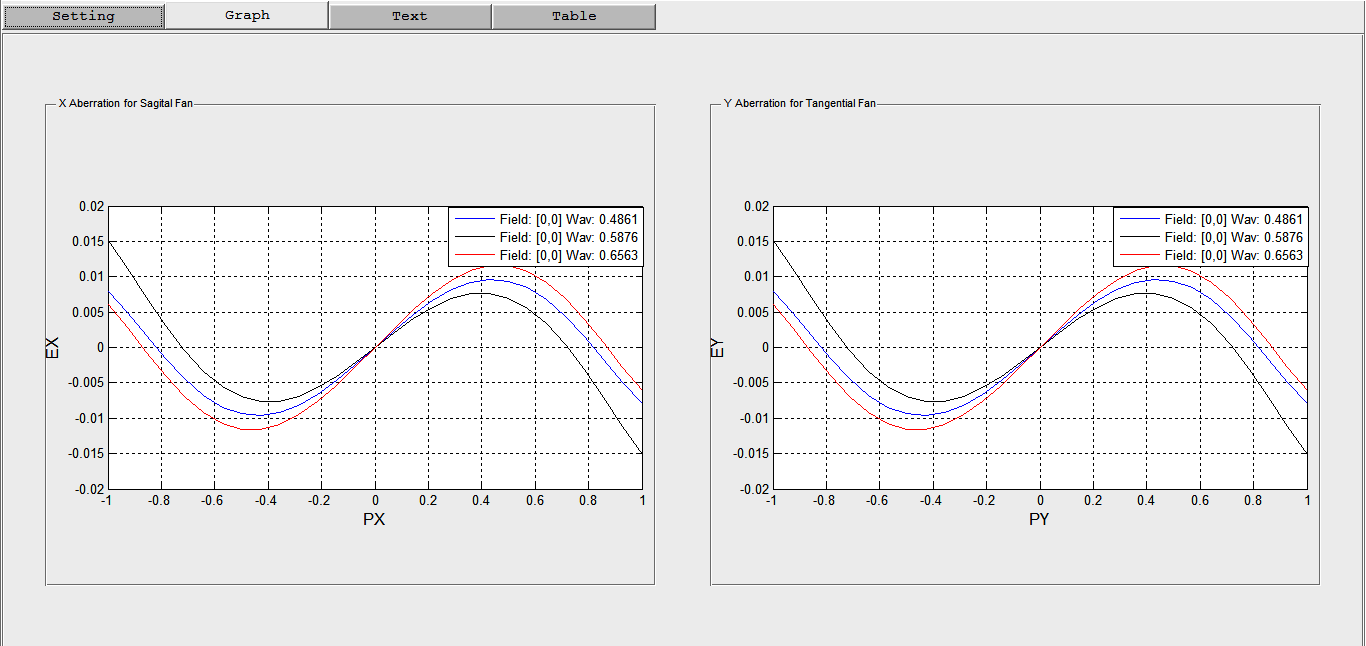
*Created with the Personal Edition of HelpNDoc:* [*Free HTML Help documentation generator*](http://www.helpndoc.com)

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)

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

# Method 1: 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.

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

**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:* [*Free iPhone documentation generator*](http://www.helpndoc.com/feature-tour/iphone-website-generation)

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:* [*Create iPhone web-based documentation*](http://www.helpndoc.com/feature-tour/iphone-website-generation)

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

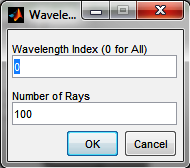
doubleGauss28 = which('DoubleGauss28.mat');

% Construct the optical system object from the saved file

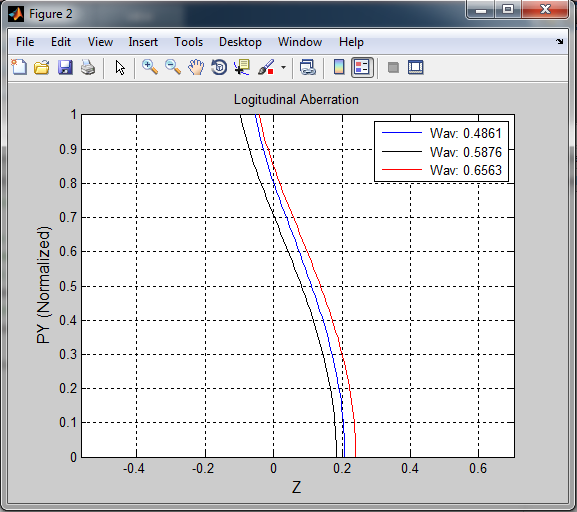
OS = OpticalSystem(doubleGauss28);

plotLongitudinalAberration(OS);

**Setting Window:**



**Graphical Longitudinal Ray Aberration Diagram Window:**



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