Chapter 25D
CODE V Supplied Macros

Contents

Menu
Contents
Indexes
Search

Figures

CODE V Supplied Macros

Introduction

A variety of macros are supplied with CODE V. They augment CODE V's built-in capabilities and also offer additional examples of how to write macros of various types. Macros are supplied in a number of categories, ranging from simple utilities to mini-applications for fifth-order analysis, generalized plotting, and others. In most cases, no programming is required to make use of these macros - simply follow the input syntax shown here. In the case of generalized plotting, the supplied macros are used in user-written macros that construct the desired plots (an annotated example of such a macro is included in this section).

This section of the Macro-PLUS documentation lists these ORA-supplied macros by category. For each macro, we include a brief description of the purpose and the syntax for running it (along with any special requirements or restrictions). Refer to the macro listing itself (or the "MACROLIB.SEQ" on page 25D-15) for a more complete description of what it does (supplied in most cases in comments that begin each macro).

Using ORA-supplied Macros

Most of the supplied macros expect to receive one or more parameters on the command line (the number of parameters expected depends on the macro and is shown in this section in the syntax for each macro). These parameters may be either numeric data or text data. In the syntax descriptions that follow, numbers to be included are shown by their variable name and text to be included is shown by the variable name in quotes; you must use quotes as indicated (either " or ').

The syntax descriptors for each macro follow the usual CODE V command syntax convention where items in square brackets [ ] are optional, and will have default values associated with them. Items not in brackets are required. For many of the macros, calling the macro with no parameters (even if required) will result in a help message being printed giving the syntax for the macro and the defaults (if any).

When you run these macros, you will usually NOT want to see the macro commands displayed during execution (the default is to display the macro commands). Use the command:

```
VER ALL NO
```

to turn off the display of macro commands (only needed once per CODE V session - you may put this in your DEFAULTS.SEQ file).

In most cases, the ORA-supplied macros are self-contained (i.e., they do not call any additional macros). You can run these macros directly from the standard macro directory (CV_MACRO:) without first copying them to a local directory:

```
IN CV_MACRO:macro_name          (followed by any required parameters)
```

You can also include the CV_MACRO directory into your sequence path (PTH SEQ CV_MACRO:); then the CV_MACRO is not needed on the calling line. If you use these macros, this is highly recommended.

```
PTH SEQ CV_MACRO:
in fl                                ! Calls CV_MACRO:FL.SEQ
```

A few of the supplied macros call upon other supplied macros as "subroutines," and if CV_MACRO: is not in the sequence path these subroutines must be in your local directory from where you are running CODE V. These macros are indicated by the notation "Calls:", followed by a list of subroutine macros called.

You may copy any of the ORA-supplied macros to your local directory for ease of use (avoids the need to type CV_MACRO: each time) or to allow modification of a supplied macro for your own use. You should NEVER modify the ORA-supplied macros on the CV_MACRO: directory itself. These are "master" copies that other users on your system may need. Always make modifications on a local copy of the macro. To copy a macro to your local directory you can use the CODE V LIB option.

```
LIB
COPY CV_MACRO:macro_name.SEQ macro_name.SEQ
```

Examples:

```
LIB
COPY  CV_MACRO:SETDECEN.SEQ SETDECEN.SEQ
COPY  CV_MACRO:CHECKRAY.SEQ CHECKRAY.SEQ
```

You can also use the file copy command provided by your operating system. For example, to copy the macro PERTURB.SEQ to your local directory, use the following command:

```
C:\CVUSER> COPY C:\CODEV102\MACRO\PERTURB.SEQ
```

Finding a Supplied Macro

In the GUI, the **Macro** manager dialog box (**Tools** > **Macro Manager** menu) lists all CODE V supplied macros. In the navigation tree in this dialog box, click a macro category to list the associated macros. Click the macro you want to run, and then click the **Run** button.

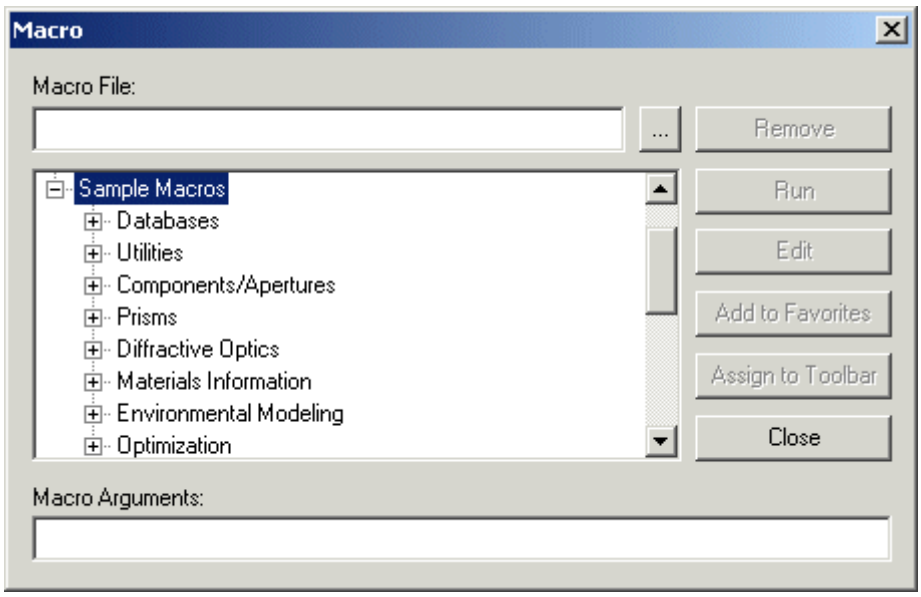


Figure 1. Macro manager dialog box

A special macro, MACROLIB, has been written to help you find a supplied macro when the name is not known. This macro is interactive and prompts you for inputs. It lists on-line the categories of supplied macros, sorted by the same categories as are in this section of the Macro-PLUS documentation. You select a category, and the macro lists the supplied macros included in that category. You can then select a macro, and then that macro's purpose, syntax, input descriptions, and explanatory notes are listed. Since the macro is interactive, a Q (for quit) must be entered to exit the macro. Note that the listings for each macro from MACROLIB are more extensive than the descriptions in this section of the Macro-PLUS documentation.

Generalized Plotting Macros

"Generalized plotting" is the name given to a set of utility macros that are used to create CODE V plotting files (filename.PLT) directly under macro control. They provide tools for drawing lines, labels, rectangles, plot frames, and other line plotting elements, allowing you to create totally new types of plots based on CODE V data.

Since each one of these macros performs a small, specialized task (like drawing a line between two points), you typically need to call several of them from a control macro (written by you) to create a plot. While you can call the macros by their actual names, it is more convenient to use short identifiers ($dra, $mov, etc) that are defined for each macro. To use these identifiers, the cv_macro:INITPLT.SEQ macro must be run first (this macro creates global string variables containing the identifiers - you only need to run it once per CODE V session, before using the plotting macros).

Structure of Generalized Plotting User Macros

These generalized plotting macros are usually called by other macros. The calling macro will generally first call the macro which initializes a new plot file (STARTPLT.SEQ or $pltfile). It will then call several of the generalized plotting macros to create the elements of a plot, and will end with a call to the macro which closes the plot file (CLOSEPLT.SEQ or $endplot). This structure is shown in the very simple SAMPLT.SEQ shown below. Note that the generalized plotting macros only CREATE the plot (.PLT) file. To display this file or send it to a printer or plotter, you must use the CODE V commands:

```
DPL filename.PLT      ! To display plot on screen
PLT filename.PLT      ! To send plot to current plot device.
```

Note that since you are creating the .PLT file directly, and not through CODE V, the plot is independent of the GRA command. For additional examples of this usage, refer to the sample macros which use generalized plotting, listed below.

Generalized Plotting Example

The following macro is not optically useful, but it is a brief, complete example of the use of generalized plotting macro commands. This macro simply plots a random walk of a given number of steps. Before running this example, you must initialize the generalized plotting commands:

```
in CV_MACRO:INITPLT
```

and type the macro into a file using your standard system text editor (there is no need to actually do this now since a version of SAMPLT.SEQ also exists on CV_MACRO - you can just copy it or run it directly with IN CV_MACRO:SAMPLT 10). The generalized plotting commands are indicated in bold in the macro listing.

```
! SAMPLT.SEQ

! Assumes INITPLT.SEQ has been run
! Sample plot macro to try PLT commands (plot #1 random
!   segments)
! Command : $sample
! Usage: IN SAMPLT <NUMBER>
!    Ex: IN SAMPLT 10
!    or: $SAMPLE 10

lcl num ^i ^numSteps      ! Declare local variables
^numSteps == #1           ! Number of random steps is param. #1

$pltfile "samplot"        ! Open plot file samplot.plt
```

```
! The following sets the plotting range of x (-5 to 5) and y (-4
!  to 4). This maps these coordinates to the long (x) and short
!  (y) width of the paper or screen.
$scale -5 5 -4 4          ! Set plot coords and scale

! The following commands draw vertical and horizontal axis lines
!  and place a label near each to indicate the scale
$mov 0 -3     ! Move pen to x=0, y=-3 without drawing
$dra 0 3      ! Draw to x=0, y=3 (move with pen down - vertical
              !  line)
$lab "3.0"    ! Draw the label "3.0" (a string) at the current
              !  pen position
$mov -4 0     ! Move to x=-4, y=0
$dra 4 0      ! Draw horizontal line to x=4, y=0
$lab "4.0"    ! X axis label

! Now move pen to upper left quadrant and plot a title
$mov -3 2.5           ! Move pen to x=-3, y=2.5
$lab "Random Walk"    ! Place label at current pen position

! Finally move pen to (0,0) and start loop for random walk.
!  Points are generated randomly and scaled to lie in plotting
!  range.  Note that the built-in function num_to_str(^i)
!  converts the number ^i to a string so it can be used as
!  a label.
$mov 0 0
for ^i 1 ^numSteps      ! Loop to draw random segments
  $dra (-4+8*randf) (-3+6*randf)    ! Draw to random point (x,y)
  $lab num_to_str(^i)               ! Label point with step
                                    !  number
end for

! Finally end the plot -- this closes the plot samplot.PLT (but
!  doesn't display it).
$endplt

! Display the plot on screen with standard CODE V command
dpl samplot
```

The resulting plot is shown on Figure 2. The other sample macros listed below include longer and more optically interesting examples of generalized plotting, but all of them follow this same basic structure demonstrated in SAMPLT.SEQ.

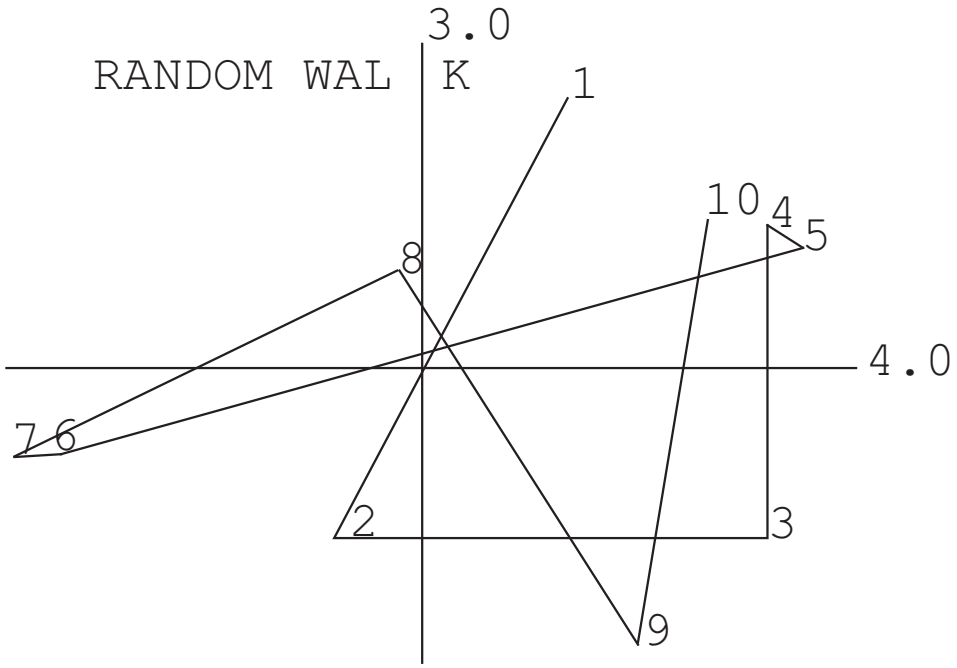


Figure 2. Output of the SAMPLT Macro

Examples of Generalized Plotting User Macros

These macros are examples of plots that can be created by using the generalized plotting macros. Before using many of these, INITPLT must be run.

DIST.SEQ

Plots a diagram of the two-dimensional distortion for the specified zoom position (default 1) and specified field of view. The default for yfov is xfov. if 'file' is blank (default), the plot file is deleted after display. Refgrid is the reference grid drawn (0 = no grid, 1 = outline only, 2 = full grid, 3 = fiducial marks only). Default for 'color' is 'RED'. Range for num_lines is 2 to 21, default is 11. If 'list' is 'Yes', the distortion output are also listed; default is 'No'.

```
in dist xfov [yfov ['file' [refgrid['color'[num_lines[zoom ['list']]]]]]]
Def:     req'd xfov   ''       2     'RED'     11      1    'No'
```

FRAMEPLT.SEQ

An example of a plot in standard CODE V style.

```
in frameplt
```

HISTPLT2.SEQ

Plots a histogram of spherical aberration for all zoom positions.

```
in histplt2 axis_scale_value
$histsa axis_scale_value
```

SAMPLT.SEQ

A sample of generalized plotting. Plots a random walk of n steps (default is 10 steps).

```
in samplt [number]
$sample [number]
```

SPHIST.SEQ

Plots a histogram of spherical aberration for the Double Gauss lens.

```
in sphist
```

Generalized Plotting "Primitive" Macros

The identifiers ($circle, $dra, etc) are defined by INITPLT.SEQ which must be run before using them (only needed to be run once per CODE V session).

CIRPLT.SEQ

Draws a circle of a specified radius centered at the specified location. Default for delta_angle is 10°.

```
in cirplt x_center y_center radius [delta_angle]
$circle x_center y_center radius [delta_angle]
```

CLOSEPLT.SEQ

Closes the generalized plotting file.

```
in closeplt
$endplot
```

COLORPLT.SEQ

Changes color in the generalized plot file. Range of n is 0 to 8 (default 0).

```
in colorplt [n]
$col [n]
```

DRAWPLT.SEQ

Draws a line from the current location to the specified location.

```
in drawplt x y
$dra x y
```

GLABPLT.SEQ

Global labeling. Inserts the text in global string variable ^labtext into the plot file at the current location.

```
in glabplt
$glab
```

HELPLT.SEQ

A help file for generalized plotting macros and their use. Explains scaling.

```
in helplt
$phelp
```

INITPLT.SEQ

Initializes generalized plotting commands.

```
in initplt
```

LABELPLT.SEQ

Adds a label into the plot file at the current location.

```
in labelplt "text"
$lab "text"
```

LDIRPLT.SEQ

Defines the orientation of the labels in the plot file. Angle in degrees, 0° corresponds to horizontal labels.

```
in ldirplt angle
$ldir angle
```

LSIZPLT.SEQ

Changes the size of the labels (in inches) in the plot file.

```
in lsizplt label_size
$lsiz label_size
```

MOVEPLT.SEQ

Moves the pointer location to the specified location.

```
in moveplt x y
$mov x y
```

ORAPLT.SEQ

Draws an ORA style plot border.

```
in oraplt "title"
$frame "title"
```

PLUSPLT.SEQ

Adds a plus sign of given size (in inches) at the specified location.

```
in plusplt x y size
```

RDRAWPLT.SEQ

Draws a line to a location relative to the current one.

```
in rdrawplt delta_x delta_y
$rdra delta_x delta_y
```

RECTPLT.SEQ

Draws a rectangle of given size (in inches) with the lower left corner at the current location.

```
in rectplt x_side y_side
$rect x_side y_side
```

RMOVEPLT.SEQ

Moves the pointer relative to the current location.

```
in rmoveplt delta_x delta_y
$rmov delta_x delta_y
```

SCALEPLT.SEQ

Defines the x and y limits for the generalized plot. These limits correspond to a plot size of 9 inches by 7 inches.

```
in scaleplt xmin xmax ymin ymax
$scale xmin xmax ymin ymax
```

STARTPLT.SEQ

Initializes a generalized plot file with the given name.

```
in startplt "filename"
$pltfile "filename"
```

SYMBPLT.SEQ

Adds a special symbol at the current location. Symbols are as used in SPO (DOT command).

```
in symbplt n
$sym n
```

Databases

CATALOG.SEQ

Enters a specified lens from a given manufacturer's catalog. The catalog is specified by a catalog code `'XX'` (ES - Edmund Scientific, LP - LightPath Technologies, MG - Melles Griot, NC - Newport Corporation, OS - OptoSigma, RO - Rolyn Optics, SH - Spindler & Hoyer). `'nnnnnn'` is the lens catalog part number. The optional surface number is the first surface of the catalog lens; the default is the surface AFTER the current surface. Refer to "Using the Catalog Lenses" on page 2-37 for more information on this macro.

```
in catalog 'XX' 'nnnnnn' [surface]
```

DATABASE.SEQ

CODE V provides a lens database that includes a wide variety of complete lens systems. The DATABASE.SEQ macro allows you to search the lens database for a suitable lens, based on various criteria including patent number, spectral range, and system type. You can then select a lens system from the list presented by the macro, and the lens system will be opened in CODE V.

Note that the CODE V lens database is also accessible using the CODE V patent lens search (**Tools** > **Patent Lens Search** menu or PATENT.SEQ macro), or in the sample lenses listed in the New Lens Wizard (**File** > **New** menu).

```
in DATABASE ['database1' ['database2']]
```

where:

'database1'	–	First database file to be searched. Quotes must be used. Default database file is `cv_lens:database`. Default file extension is .DAT.
'database2'	–	Second database file. Quotes must be used. Default file extension is .DAT.

You can also access the database lenses directly in the LENS subdirectory in the main CODE V installation directory. The database lenses are categorized by lens type; their sequence names are indicative of their lens type. All the database lens sequence names have the form ORAcategoryxx.SEQ. The ORA in the first part of the name allows easy identification of the sequence as a CODE V database lens. The second part of the name of the lens sequence follows from the lens category, and the third part is the number of the entry in that particular category. Thus, the third entry in the category TRP (for triplet lenses and derivatives) would have the sequence name ORATRP03.SEQ. The category names used and number of lenses in each category so far are as follows (more lens categories and lenses will be added in future releases of CODE V):

Category	No.	Description
CAT	1	Catadioptric lenses (including catoptric lenses)
DBG	30	Double Gauss and derived forms
EYE	2	Models of the human eye
EYP	12	Eyepieces, oculars, and magnifiers
FSH	1	Fisheye lenses (very wide angle lenses)
PHO	19	Photographic objectives
PTZ	5	Petzval type lenses
RTL	11	Reverse telephoto lenses (including some wide angle lenses)
SYM	11	Symmetrical or nearly symmetric forms
TLP	7	Telephoto lenses
TRP	10	Triplets and derived forms
WID	4	Wide angle lenses
Total	113	

The lens data in each sequence have not been normalized, and reproduce the patent data, if the lens was obtained from a patent. Most lenses use the visible spectrum (C, d, and F wavelengths) unless they are specifically for a different spectrum (such as an IR lens). The EPD, fields of view, and stop location are as specified in the patent, or if not specified in the patent, then reasonable values are used. Dimensions are as specified in the patent (inches or mm), but lenses normalized in the patent to unity focal length are given dimensions of inches. Vignetting has been included as necessary to eliminate negative edge thicknesses. Unless catalog glass names are specified in the patent, fictitious glass forms are used in the sequences. Thus, it will normally be up to the users to select real glasses from the glass catalogs of their choice to replace these fictitious glasses. In some instances, the replacement of the fictitious glasses by real glasses will affect the secondary spectrum.

To use a given sequence in the CODE V lens database, open the file using the **File** > **Open** menu, or the IN command on the command line:

```
CODE V> IN CV_LENS:ORATRP03
```

The PTH command can be helpful here. If the CV_LENS directory is in the sequence path (PTH SEQ CV_LENS:), then only the sequence name needs to be given with the IN command. Note that these sequences are complete lens sequences, and replace the lens currently in memory.

MACROLIB.SEQ

Macro to list all the supplied macros by category, including program purpose, syntax, input descriptions, and explanatory notes.

```
in macrolib
```

PATENT.SEQ

Macro to interactively search the patent lens database and sort lenses according to user-entered criteria for application, spectral range, object distance, image distance, etc. Optional search criteria allow a search to be refined by entering minimum and maximum values for number of elements, magnification, f/number, semi-FOV, etc. Any of the first four criteria may be bypassed and default values will be assumed. Lenses meeting the search criteria can be compared in a table, and then either previewed or selected. Preview generates an unlabeled 2D view (DRA). Previewing or selecting a lens will restore it, overwriting the current lens.

```
in patent
```

SEARCH.SEQ

Macro to search a specified lens manufacturer's catalog for a lens based on lens type, focal length, and diameter. The lens can then be inserted into the current lens.

```
in search
```

Utilities

ABERRATIONGENERATOR.SEQ

Aberration generator. Creates a perfect f/2 lens with specified stop shifts, reduction ratio, and third-order aberrations. All inputs are optional and have defaults of 0.0.

```
in aberrationgenerator stop_shift RED SA coma Astig Dist Ptz
DEF:     0     0   0   0   0     0    0    0
```

BUFDEL.SEQ

Deletes a buffer.

```
in bufdel buffer_number
```

MACRO.SEQ

Allows interactive input of a macro at the command prompt. If a filename is given, the macro is saved in the named file. If it is omitted, the macro is deleted after execution.

```
in macro ['filename']
```

MAKEGLBS.SEQ

Converts a range of surfaces toa global reference (GLB) decenter type with respect to a prior surface.

```
in MAKEGLBS first_surf_to_convert last_surf_to_convert global_ref_surf
```

MAKEGLBS.SEQ

The MAKEGLBS macro converts a range of surfaces to GLB decenter type with respect to a prior surface.

```
in MAKEGLBS first_surf_to_convert last_surf_to_convert global_ref_surf
```

MINSPHERICAL.SEQ

Changes the radii to make the lens bent for minimum spherical aberration (for infinite conjugates). Surface j is the first surface of the lens.

```
in minspherical surf_j focal_length
```

PLOTWL.SEQ

Plots a macro-calculated value vs. wavelength. By default, this macro plots CEF as a function of wavelength.

```
in PLOTWL value_seq [min_wvl max_wvl num_wvl [values for imag_seq]]
```

PROLITH.SEQ

Macro to transfer Fringe Zernike data from CODE V to Prolith/2 via a file.

```
in prolith 'input_file' 'output_file'
```

QUICKVIEW.SEQ

Gives quick dialog box access to common VIEW drawing features.

```
in quickview
```

REFCHECK.SEQ

Attempts to trace all five reference rays for specified fields (default all fields) and specified zoom position (default all zoom positions) and reports whether each trace was successful or where and why the ray trace failed.

```
in refcheck [start_field [end_field [zoom]]]
Def:                        0          0
```

REFRAYS.SEQ

Lists the coordinates of all reference rays for all fields and zooms for a specified surface, and computes limiting ±x and ±y values and maximum radial dimension.

```
in refrays surface
```

REFLIMIT.SEQ

Macro to list reference ray limits (±X and ±Y and maximum radius) for a given surface range (default all surfaces) and a specified zoom position (default all zoom positions).

```
in reflimit [start_surface [end_surface [zoom]]]
Def:              1            0           0
```

RSIVIEW.SEQ

Lists and plots the results of an RSI trace.

```
in rsiview pupil_choice xpup ypup field view_type
```

SETVIG.SEQ

Performs the equivalent of a SET VIG command on a lens which contains decentered apertures. There are no parameters.

```
in setvig
```

UNVIG.SEQ

Deletes all vignetting factors (i.e., resets them all to 0.0).

```
in unvig
```

UNWRAP.SEQ

Macro to "unwrap" a .INT file into a specified buffer so each row in the buffer corresponds to a complete row of .INT data. Default for buffer number is first available buffer. 'no_data_flag' is either 'Yes' or 'No' (default 'Yes') and specifies to replace all no_data values in the .INT file with zeroes. SSZ is used to override the SSZ value in the .INT file (default = 0 which means use SSZ in .INT file).

```
in unwrap 'int_file' [buffer_num ['no_data_flag' [SSZ]]]
```

VSAV.SEQ

Macro to save a lens file with a version number. The macro actually saves the file twice. When the file is saved the first time, it is not assigned a version number, according to the CODE V file naming syntax. When the file is saved the second time, the first saved file is assigned the highest version number, and the second saved file is not assigned a version number, according to the CODE V file naming syntax. The second saved file is then deleted, which leaves the first saved file as the most recent version. Therefore, the most recent version of the file has the highest version number. The macro then echoes the filename of the first saved file with its version number. The filename parameter is the name of the lens file to be saved with a version number.

```
in vsav filename
$vsa filename
```

ZEMAXTOCV.SEQ

Converts a lens file from the ZEMAX® Optical Design Program (ZEMAX Development Corporation) to a CODE V-compatible lens file format.

```
in ZEMAXTOCV Zemax_file
```

Components/Apertures

CONE.SEQ

Computes the radius and conic constant of an elliptical surface to make the current surface a cone of a given half angle. If angle is positive, radius is positive and cone “points” left. If angle is negative, radius is negative and cone “points” right.

```
in cone angle
```

ELLIPSE.SEQ

Computes the radius and conic constant to make the current surface an ellipse of a given major semi-axis length a and minor semi-axis length b. If major axis length (a) is positive, the radius is positive; if (a) is negative, the radius is negative.

```
in ellipse a b
```

FIBER.SEQ

Inserts an NSS fiber ahead of a specified surface. Fiber is assumed to be used in TIR (unless AIR is specified). Default for glass is BK7.

```
in fiber surface fiber_diameter fiber_length [glass]
Def:     req'd        req'd          req'd     BK7
```

FIBERCLD.SEQ

Similar to the FIBER macro, but enters a cladded fiber. Fiber material defaults to BK7 and clad material defaults to SILICA.

```
in fiberclad surface fiber_dia clad_dia length [fiber_matl [clad_matl]]
```

FIBERSCA.SEQ

Scales fiber entered with FIBER to a new diameter and length.

```
in fibersca start_surface fiber_diameter fiber_length
```

IRREG_POLYGON_APE.SEQ

IRREG_POLYGON_APE creates a set of overlapping rectangular apertures that simulate a convex polygon aperture. It reads a list of polygon vertex X, Y coordinates from a file or buffer.

```
in IRREG_POLYGON_APE surface_num [filename [aperture_type [print_flag]]]
```

where:

surface_num	–	Surface number on which to put apertures.
filename	–	Name of the file from which to read polygon vertex data. Default extension is .DAT. If filename starts with Bn where n is a number, a buffer number is assumed. Vertices are entered (x y) on separate lines in counter-clockwise order. The file can have comment lines in it. Any line with numbers in first two columns is assumed to be legitimate data. If the filename is blank, you are prompted for the vertices. Default: blank.
aperture_type	–	Choice of CLR, EDG, or BOTH. Default: CLR.
print_flag	–	Y|N. Flag that indicates whether to print results or not.

POLYAPE.SEQ

Inputs a polygon aperture of a given inscribed radius and number of sides to a given surface.

```
in polyape surface inscribed_radius num_side
```

POLYPIPE.SEQ

Inserts a polygon light pipe of the specified inscribed radius, length, and number of sides (default 4) ahead of a specified surface. Default for glass is SILICA.

```
in polypipe surf ins_radius length [num_sides [glass]]
```

ZCIR.SEQ

Used in zoom apertures to define zoomed apertures on a surface or range of surfaces, where each zoomed aperture is sized to pass the reference rays for the corresponding position.

```
in zcir start_surface [end_surface]
Def:        req'd       start_surface
```

Prism Macros

These are macros for automatically inserting various prism types into your lens at a specified location. All prisms except one are built with sequential surfaces (the exception is a non-sequential Pechan prism model).

These macros insert a prism into the current lens in front of surface j. The prism is automatically centered in the airspace between surface j-1 and surface j, and the airspaces around the prism are adjusted to account for the optical thickness of the prism. If these effects are not desired, then manually change the airspaces after running these macros. For each prism, you must specify the orientation of the prism as either UP, DOWN, RIGHT, or LEFT. For all these macros, the default for glass is BK7 and the default for orientation is 'down' (quotes must be used around the orientation).

These prisms are as defined in MIL-HDBK-141 (except the Delta prism).

ABBE.SEQ

Adds an Abbe prism into the current lens in front of surface j.

```
in abbe surf_j face_length [glass ["orientation"]]
```

DELTA.SEQ

Adds a delta prism to the current lens in front of surface j.

```
in delta surf_j prism_height apex_angle [glass ["orientation"]]
```

DOVE.SEQ

Adds a dove prism to the current lens in front of surface j.

```
in dove surf_j prism_height [glass ["orientation"]]
```

FRANK3.SEQ

Adds a Frankford Arsenal prism type 3 to the current lens in front of surface j.

```
in frank3 surf_j face_length [glass ["orientation"]]
```

FRANK4.SEQ

Adds a Frankford Arsenal prism type 4 to the current lens in front of surface j.

```
in frank4 surf_j face_length [glass ["orientation"]]
```

FRANK5.SEQ

Adds a Frankford Arsenal prism type 5 to the current lens in front of surface j.

```
in frank5 surf_j face_length [glass ["orientation"]]
```

PECHAN.SEQ

Adds a NSS Pechan prism to the current lens in front of surface j.

```
in pechan surf_j face_width [glass]
```

PECHAN1.SEQ

Adds a Pechan prism to the current lens in front of surface j.

```
in pechan1 surf_j face_width gap [glass ["orientation"]]
```

PECHAN2.SEQ

Adds a backwards Pechan prism to the current lens in front of surface j.

```
in pechan2 surf_j face_width gap [glass ["orientation"]]
```

PENTA.SEQ

Adds a penta prism to the current lens in front of surface j.

```
in penta surf_j face_length [glass ["orientation"]]
```

PORRO.SEQ

Adds a Porro prism to the current lens in front of surface j.

```
in porro surf_j face_length [glass ["orientation"]]
```

REVERS.SEQ

Adds a reversion prism to the current lens in front of surface j.

```
in revers surf_j face_length [glass ["orientation"]]
```

RTANG.SEQ

Adds a right angle prism to the current lens in front of surface j.

```
in rtang surf_j face_length [glass ["orientation"]]
```

Diffractive Optics (Binary Optics) Macros

These macros are all associated with the design and analysis of kinoforms (binary optics). Kinoforms are usually modeled in CODE V with zero power HOEs with aspheric phase defined with the HCO coefficients.

Some of these macros make use of special user-defined functions for HOE calculations, based on technical reports from MIT Lincoln Laboratory.[1,2] These FCTs are defined in the macro HCODEFS; this macro must be run first before running any HCO macro which calls these functions. These macros are identified by a Calls HCO FCTs.

ACHROMAT.SEQ

Macro to input a sample lens/HOE hybrid achromat. The macro also plots the MTF with and without the HOE effect.

```
in achromat
```

BINFAB.SEQ

Macro to output binary optics HOE data in a form useful for binary optics fabricators. Filename (must be enclosed in quotes) will be given a .BIN extender. (This macro was written by Digital Optics Corporation, Charlotte, NC, 704-549-5556.)

```
in binfab surface 'filename'
```

DOCFLAT.SEQ

Macro to input a sample lens which converts input Gaussian intensity profile to a square top profile. The macro also performs a PSF to show the results.

```
in docflat
```

DOESET.SEQ

Used to set the diffractive surface type by specifying a surface label. The type refers to a fabrication and encoding method used by the other macros.

Surface is the surface number of the DOE. Type is the type of DOE to be manufactured and is entered as a text string. The possible choices for type are:

'2 phase level'
'4 phase level'
'8 phase level'
'16 phase level'
'Mixed N'
'DS'

The type 'Mixed N' uses varying phase levels as needed until the minimum feature size is below a given level (set in other macros). 'DS' means direct sampling and also uses a varying number of phase levels

as needed up to a maximum number of phase levels. This macro must be run before the other macros in this series are run.

```
in doeset surface 'type'
```

DOEPARM.SEQ

Displays all the relevant phase parameters and a brief description of the selected manufacturing process.

```
in doeparm surface
```

DOEFFIC.SEQ

This macro computes the diffraction efficiency based on scalar and extended scalar diffraction theories. Efficiency is computed for both the construction wavelength and the system wavelengths.

```
in doeffic surface [num_steps_x [num_steps_y]]
```

Surface is the surface number of the DOE surface. The number of steps defaults to 5.

DOESAG.SEQ

This macro generates a sag table for a diamond-turned kinoform lens and plots the phase. Use only with rotationally symmetric phase polynomial (HCT R), DIF HOE or DIF DOE, zoom position. Use only for refractive HOEs, diffraction order HOR = +2 or -2. Sag steps occur at multiples of 1 wave at HWL wavelength. The blaze depth (BLD) is not used. Sag table is in uniform radial increments (dr), with two extra points inserted at each 1-wave phase transition step. The radial increment is set such that the phase change per increment is 1/4 wave or less, unless overridden by user. Works only with Code V 9.30 or higher. Prior to 9.30, a HOE was a separate surface type.

```
in DOESAG surf_num [max_radius [sampling_incr]]
```

Surf_num is the surface number of the HOE or DOE surface.

Max_radius is the clear aperture radius (default = MAP).

Sampling_incr is the radial sampling increment "dR" in lens units (default = increment which corresponds to 1/4 wave phase at steepest zone).

Buffers: Utilizes the first three empty buffers: phase in waves, phase modulo 1 wave, and sag table for diamond turning. Data in buffers can be exported with BUF EXP Bn filename, or listed with BUF LIS Bn. Phase modulo 1 wave is in buffer B^B2. Sag for diamond turning is in buffer B^B3.

DOESLICE.SEQ

Plots the phase or efficiency of a specific surface along a specified radius, or for non-rotational DOEs, along a specified slice. The start points default to 0 and the end points default to the maximum aperture of the surface. The default is for efficiency.

```
in doeslice surface ['EFF'|'PHA' [x_start [y_start [x_end [y_end]]]]]
```

DOE2DPLT.SEQ

Displays a false color raster map of the diffraction efficiency or phase derivative for any HCT type. The default plot is for efficiency.

```
in doe2dplt surface ['EFF'|'DER']
```

F1HYBRID.SEQ

Macro to input a plano convex lens and flat binary lens to form an f/1 hybrid lens.

```
in f1hybrid
```

HCODEFS.SEQ

Defines FCTs @HCOphi, @HCOfindR, @HCOscEff, @HCOscEff2, and @HCOsxScEff and global variables ^dPHIdr and ^d2PHIdr2. This must be run before running many of the other HCO macros.

```
in hcodefs
```

HCODOUB.SEQ

Sample of the design of a 2-element germanium binary lens.

```
in hcodoub
```

HCOEFFIC.SEQ

Computes diffraction efficiency at a specified radius r on the HOE (default is semi-diameter). Number of mask levels must be input (default 16), and wavelength number (default reference wavelength number). MSRF is surface number of substrate if HOE was input with air on both sides. Calls HCO FCTs.

```
in hcoeffic surf  [r    [N_levels [WL_num [msrf]]]]
Def:        req'd (SD)     16       (REF)    0
```

HCOEFPLT.SEQ

Plots diffraction efficiency for HCO-only kinoforms on a specified surface, for a given number of mask levels (default 16), in N steps (default 50), from minimum radius (default 0) to maximum radius (default semi-diameter). MSRF is surface number of substrate if HOE is input with air on both sides. Calls HCO FCTs.

```
in hcoefplt surf [N_levels [N_steps [rmin [rmax [msrf]]]]]
Def:        req'd   16        50       0    (SD)  0
```

HCOMASK.SEQ

Creates a PostScript file for a binary optics mask for a HCO-only HOE on a given surface. Default for filename is MASK.PST; default for scale factor is 1.0; default for radius is semi-diameter. Calls HCO FCTs, and required HCOPHTAB be run first.

```
in hcomask surf ['filename' [scale_factor [radius]]]
Def:       req'd 'mask.pst'      1.0        (SD)
```

HCOPHTAB.SEQ

Compute and print a table of phase (phi) and phase derivative and locate radii where phase is a multiple of π/M for mask level M. Default starting radius is 0; default ending radius is semi-diameter; default number of radius points is 200, default radius tolerance is 0.00001; default mask number is 1. PrintAll is 'Y' for full table (default) or 'N' for ring radii and phase information. Calls HCO FCTs.

```
in hcophtab surf [rmin [rmax [num_radii [rad_tol [mask_num
  ['PrintAll']]]]]]
Def:         req'd  0   (SD)    200       0.00001  1     'Y'
```

HCOPLOT.SEQ

Plots HCO part of HOE phase as a function of radius (default minimum 0, default maximum semi-diameter, default number of points 50). Final Yes or No controls output on screen of UGR listing. Calls HCO FCTs.

```
in hcoplot surf [rmin [rmax [num_points [Yes|No]]]]
Def:        req'd   0    (SD)   50           NO
```

HCORTOXY.SEQ

Computes and enters 10th order radial phase data into proper x-y HCO coefficients, all other HCO coefficients are set to zero. If all coefficients are zero, then coefficients C1 to C5 are used.

```
in hcortoxy surf [a(2) [a(4) [a(6) [a(8) [a(10)]]]]]
Def:         req'd   0      0      0      0      0
```

HCOSAMPL.SEQ

Sample HOE design session using HCO macros. The macro designs an f/2 germanium singlet with a HOE on it. Calls HCO FCTs. Also calls HCOVAR, HCOPLOT, HCOEFFIC, HCOPHTAB, and HCOMASK.

```
in hcosampl
```

HCOSCAN1.SEQ

Sample macro to design an f-theta scanner using a HOE on a BK7 flat plate. (Described by Buralli and Morris, *Applied Optics*, June 1991, pg. 2151.) Calls HCOVAR and HCOSCAUT.

```
in hcoscan1
```

HCOSCAN2.SEQ

Sample macro to design an f-theta scanner using a HOE on a plano-convex BK7 lens. (Described by Buralli and Morris, *Applied Optics*, June 1991, pg. 2151.) Calls HCOVAR and HCOSCAUT.

```
in hcoscan2
```

HCOSCAUT.SEQ

AUTO optimization sequence called by HCOSCAN1 and HCOSCAN2 macros. Input parameter is aspheric HOE order.

```
in hcoscaut order
```

HCOSCVIEW.SEQ

Plots overlaid VIEW plot of scanner designed with HCOSCAN1 or HCOSCAN2.

```
in hcoscview
```

HCOUVSIN.SEQ

Sample macro to set up and optimize an achromatic fused silica HOE for the UV (247-249 nm).

```
in hcouvsin
```

HCOWLEFF.SEQ

Macro to plot HCO scalar diffraction efficiency vs. wavelength. Inputs are HOE substrate index (default 1.5), reference wavelength (default 500 nm), wavelength minimum (default 250 nm), and wavelength maximum (default 1000 nm). Calls HCO FCTs.

```
in hcowleff [index [ref_wl_nm [max_wl_nm [min_wl_nm]]]]
Def:         1.5      500        1000        250
```

HCOXYTOR.SEQ

Macro to compute and print radial phase equation for a given HOE surface. (No check is made to verify rotational symmetry.) Converts HCT XY to HCT R.

```
in hcoxytor surface
```

Menu
Contents
Indexes
Search

Materials Information

ARCHERPRV.SEQ

ARCHERPRV loads the Archer Molded Glass catalog into a lens private catalog.

```
in ARCHERPRV
```

BIRMAT.SEQ

BIRMAT provides a drop down list of uniaxial crystal birefringent materials that can be applied to a selected surface. The macro automatically computes and enters the appropriate NEO values based on the system wavelengths. Material dispersion data is from the *OSA Handbook of Optics*.

```
in BIRMAT
```

GLASSFIT.SEQ

Macro to find closest glasses in user-selected glass catalogs to fictitious glasses. Options are to evaluate only or fit the glasses. Optional parameter v_factor controls emphasis on dispersion vs. index (default 1.0).

```
in glassfit [v_factor]
Def:           1.0
```

PLASTICPRV.SEQ

Creates a Private Catalog of 17 optical plastics based on information in the *OSA Handbook of Optics*, data measured and fit to a dispersion equation by Honeywell France, and vendor supplied data from B. F. Goodrich, Hitachi Chemical Co., and Hoechst Celanese. Two data sets (from separate sources) are included for the common optical plastics: acrylic (PMMA), polycarbonate, polystyrene, and styrene acrylonitrile (SAN). Background information for each Private Catalog entry is displayed when the macro is run.

```
in plasticprv
```

SPECMAT.SEQ

Macro to output the Abbe (V) partial dispersion value (P), and reference index for materials in the SPECIAL catalog if the material is defined in the designated wavelength band. For example, IN SPECMAT 650 550 450 would not list any information for GERM, SILICN, etc., because their indices are not defined in the visible region. This allows the macro to be used to determine what materials from the SPECIAL catalog have index properties defined in the specified band.

The syntax for this macro is:

```
in specmat wavelength_1 wavelength_2 wavelength_3
```

The input wavelengths for this macro are in nanometers and must be entered in descending order (similar to CODE V's WL command).

UMICOREPRV.SEQ

Macro to enter Private Catalog Information for the Umicore IR glasses (http://www.umicore.com).

```
in umicoreprv
```

VP_PLOT.SEQ

Creates a table of data and Index vs. Dispersion and Partial Dispersion vs. Dispersion plots for the materials in the current lens, or one of the CODE V glass catalogs, over any wavelength region. Replaces the VP_DATA macro.

```
in vp_plot
```

Environmental Modeling

DNDTCALC.SEQ

DNDTCALC.SEQ calculates absolute dn/dT values for Schott and Ohara glasses, for use in CODE V's Environmental Change (ENV) option (**Lens** > **Environmental Change** menu). This macro uses Schott and Ohara formulas for more precise dn/dT values, rather than the default lookup table values that are based on dn/dT for a temperature range of 20° to 40° Celsius. The macro can save the appropriate DN commands in a sequence file in the current working directory, or list the requested dn/dT data.

Schott and Ohara report that the temperature coefficients of refractive indices are accurate within a temperature range of -40° to +80° Celsius, and within a wavelength range of 0.6438 µm to 0.4358 µm.

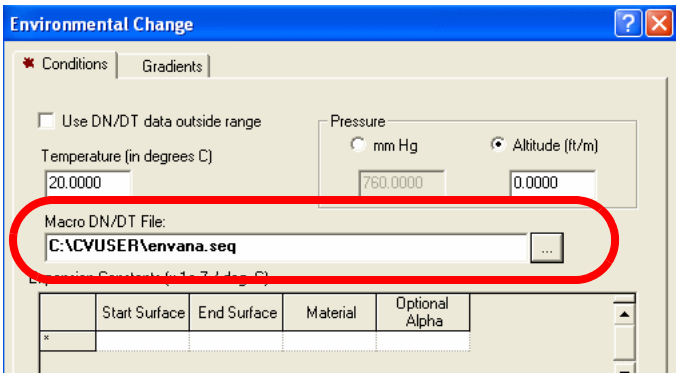
The syntax is:

```
in DNDTCALC [1 or 2] [temperature] [glass name] [# wavelength]
  [wavelength 1] [wavelength 2] [wavelength 3] [wavelength 4] [wavelength 5]
```

where:

1 or 2	–	1 = create an ENV sequence file; 2 = list dn/dT data. Choosing 1 generates a sequence file named *envana.seq* in your current working directory, which contains the dn/dT data for the specified glass and system temperature. Choosing 2 displays the dn/dT data in a text output window. Default: 1
temperature	–	System temperature. Default: 40 (in degrees celsius)
glass name	–	Schott or Ohara glass name. Default: F2 (Schott)
# wavelength	–	Number of wavelengths. When the wavelength number is 0, the macro uses the current design wavelength. You can enter up to 5 wavelengths.
wavelength	–	If a non-zero number of wavelengths is entered, specify each wavelength in nanometers (nm).

For command-line use, you can edit the sequence file (*envana.seq*) with additional ENV commands along with the generated DN commands. In the GUI, you can select the ENV sequence file in the **Macro DN/DT File** field, as shown in the following figure.



When you select an ENV sequence file in the **Environmental Change** dialog box, the system temperature in the **Temperature** field will be overridden by the system temperature specified in the sequence file, so you may wish to delete or comment out the TEM line in the sequence.

DNDTCALC.SEQ Usage Notes

When calculating the temperature coefficient of refractive index for a particular glass, you generally want to use the average of the system temperature change. One way to do this is to enter the average system temperature in DNDTCALC, comment out the temperature in the *envana.seq* file, enter the ending system temperature in the **Environmental Change** dialog box, and then select the modified *envana.seq* file in the **Macro DN/DT File** field.

For example, for a temperature change from +20° to -20° Celsius, do the following:

1. Enter 0 for the system temperature in the DNDTCALC macro dialog box.
2. In the *envana.seq* file, comment out the temperature command (a sample is shown below), and save the file.

```
!tem              0
!WL(nm) 500
dn s1 1.3696897
```

   By default, the *envana.seq* file is saved by the DNDTCALC macro in your working directory.

3. In the **Environmental Change** dialog box (**Lens** > **Environmental Change** menu), type **-20** in the **Temperature** field.
4. In the **Macro DN/DT File** field, navigate to and select the modified *envana.seq* file.
5. Click **OK** to run the ENV option.

THERMPIK.SEQ

The THERMPIK macro sets up an optical system for athermal optimization. It creates a zoom model at nominal, low, and high temperatures. The lens is zoomed for the three temperatures. The syntax is:

```
in thermpik [min_temp] [max_temp] [A|R]
```

where:

min_temp	–	Sets the minimum temperature value for the temperature range across which the system will be optimized.
max_temp	–	Sets the maximum temperature value for the temperature range across which the system will be optimized.
A | R	–	A - sets min/max temperatures in terms of absolute indices. R - sets min/max temperatures in terms of relative indices. Default is R.

THERMPIK calls the following two supporting macros:

The THERM_N macro computes the index of glass at a user-defined nominal, low, and high temperature, and stores the index data.The index data is used to create a private catalog and assign it to the zoomed glass types.

The THERM_ENV macro runs the Environmental Analysis (ENV) option using the parameters generated by the THERMPIK and THERM_N macros.

In the **Macro** manager dialog box (**Tools** > **Macro** menu), you can access these macros in the **Sample Macros** list, in the **Environmental Analysis** category.

Optimization Macros

AUT_GRID_RET.SEQ

The AUT_GRID_RET macro uses a combination of CODE V's default merit function and a user merit function composed of the retardance evaluated on a rectilinear grid using the RAYPOL macro function. Before running this macro, you need to run the AUT_GRID_RET_FCT macro once during the current CODE V session to define the user-defined functions used to build the merit function.

```
in AUT_GRID_RET nrays cmd wavelength
```

where:

nrays	–	Number of rays across the diameter.
cmd	–	Optional. User-specified Automatic Design (AUTO) commands. Each command is separated by a semi-colon character (;).
wavelength	–	Wavelength number (0 for reference wavelength).

AUT_GRID_RET_FCT.SEQ

The AUT_GRID_RET_FCT macro defines the user-defined functions used to build the merit function that the AUT_GRID_RET macro uses. Note that you can only run this macro once during the current CODE V session.

```
in AUT_GRID_RET_FCT
```

AUT_GAUSS_POLGRID.SEQ

The AUT_GAUSS_POLGRID macro uses a Gaussian quadrature to compute the merit function to higher accuracy than is possible with a rectilinear grid with the same number of rays. The merit function is composed entirely of data from the POLGRID macro function. Before running the macro, you need to run the AUT_POLGRID_FCT macro once during the current CODE V session to define the user-defined functions used to build the merit function.

```
in AUT_GAUSS_POLGRID nrings nspokes cmd zoom wavenumber
```

where:

nrings	–	Number of rings.
nspokes	–	Number of spokes.
cmd	–	Optional. User-specified Automatic Design (AUTO) commands. Each command is separated by a semi-colon character (;).
zoom	–	Zoom position to optimize.
wavenumber	–	Wavelength number (0 for reference wavelength).

AUT_POLGRID_FCT.SEQ

The AUT_POLGRID_FCT macro defines the user-defined functions used to build the merit function that the AUT_GAUSS_POLGRID macro uses. Note that you can only run this macro once during the current CODE V session.

```
in AUT_POLGRID_FCT
```

AUTOGRID.SEQ

Plots the ray grid used by AUTO for a given field, DEL value, zoom position, and obscuration ratio. Default DEL = 0.385; default zoom position is 1; default obscuration ratio is 0.0. If plot_file is given, plot is saved in that file; 'SAP' specifies square aperture. (Does not require INITPLT).

```
in autogrid field [del [zoom [obs_ratio ['file_name'
              ['sap'][symbol_size]]]]]
Def:         req'd .385    1      0            ''       ''
```

AUTOGRIDGQ.SEQ

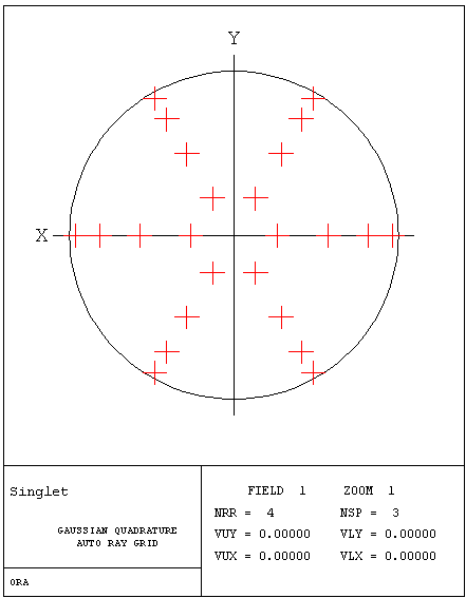
Plots the Gaussian quadrature ray grid used by AUTO for a given field, number of rings, number of spokes, and zoom position. This macro function ignores pupil obscurations (OBS), as well as the following pupil shapes: sagittal slit (SAG), meridional slit (MER), and rectangular (SAP). The plot's perspective is looking into the lens from the object.

```
in autogridgq field [NRR NSP [zoom ['file_name' [symbol_size]]]]
```

where:

field	–	Field number (required).
NRR	–	Number of rings. Default: 3 for transverse ray aberrations 4 for wavefront error types.
NSP	–	Number of spokes. Default: 3.
zoom	–	Zoom position. Default: 1.
'file_name'	–	File name of saved plot file. If blank, file is deleted after it is displayed in CODE V. Default: Blank.
symbol_size	–	Size of plot symbols (+) as a fraction of grid spacing (DEL). Default: 0.4.

The following plot is produced:



PLOT_GL.SEQ

Macro to plot the results of a Global Synthesis run. It draws the lenses and lists the glasses for variable glasses. Inputs are prompted for.

```
in plot_gl
```

TESGLASS.SEQ

Macro to optimize variable glasses, selecting from a pre-specified list of allowable glasses. 'glass_file' is a file containing the names (and catalog names, if necessary) of the allowed glasses, and 'aut_file' is a file containing the sequence of AUTO commands to use. 'strategy' is used to choose fitting strategy: 'nearest' (default), 'furthest', or 'tor' (based on tolerances - requires a TOR run be run first). 'output_flag' selects whether to list the AUTO output (default is 'Yes'), v_factor is importance of V to n (default 1), and 'int_flag' selects interactive mode (default 'No').

The macro saves the output of each fit in a separate lens sequence file. The files are named NAMExx.SEQ, where NAME is the first 6 characters of the lens name and xx is the number of the fit 01, 02, etc.

```
in tesglass 'glass_file' 'aut_file' ['strategy' ['output_flag'
               [v_factor ['int_flag']]]]
```

1st Order Analysis Macros

These macros form a series (except for YYBAR) to define a lens by its y-ybar data. The lens specification data is defined by YYBSPC and it is then converted via YYBCONV to lens modules for the thin lens equivalent of the y-ybar data.

To use these macros, first the macro YYBSETUP must be run to define the global variables and macro functions needed by the other YYB* macros. To get more detailed instructions on the use of these macros, run YYBHELP.

BFLPLOT.SEQ

Creates a plot of back focal length vs. wavelength for the specified zoom position (default 1). If 'list_flag' is 'Yes' (default), the output are also listed.

```
in bflplot [zoom_pos ['list_flag']]
```

FL.SEQ

Prints a table of focal lengths of each refractive optical component (components being elements or cemented elements surrounded by air).

```
in fl
```

FLPLOT.SEQ

Creates a plot of focal length vs. wavelength for the specified zoom position (default 1). If 'list_flag' is 'Yes' (default), the output are also listed.

```
in flplot [zoom_pos ['list_flag']]
```

NODP.SEQ

Calculates the nodal point locations for any specified surface range. Default for end surface is next to last surface (I-1) and default for start surface is 1.

```
in nodp [start_surface [end_surface]]
```

YYB.SEQ

Macro to change y-ybar data, list the data, plot, etc. The macro passes 'commands', which can be built up (such as 'dlp' - delete, list, and plot). Calls YYBHELP, YYBCONV, and YYBPLOT.

```
in yyb 'command' [elem_num [new_y_value [new_ybar_value]]]
Def:     req'd        0          0               0
```

YYBAR.SEQ

Separate macro from this series to use UGR to plot a y-ybar diagram for the current lens.

```
in yybar
```

YYBCONV.SEQ

Converts y-ybar data into thin lens form using lens modules. The print flag controls printing of messages (default 1 = print messages).

```
in yybconv [print_flag]
```

YYBHELP.SEQ

Help macro for YYB* macros. Default is to print information on all y-ybar topics.

```
in yybhelp ['topic']
```

YYBNEW.SEQ

Macro to set up new y-ybar data. Short command is $YNEW. Calls YYBCONV, YYBPLOT, and YYBSAVE.

```
in yybnew
$ynew
```

YYBPLOT.SEQ

Macro to use UGR with stored y-ybar data to plot a y-ybar diagram. Default is to autoscale the plot.

```
in yybplot [max_aperture]
```

YYBREAD.SEQ

Read a data file containing y-ybar data. Short command is $RES. Calls YYBHELP, YYBCONV, and YYBPLOT.

```
in yybread 'filename'
$res 'filename'
```

YYBSAVE.SEQ

Save y-ybar model in data file for later use. Short command is $SAV.

```
in yybsave 'filename'
$sav 'filename'
```

YYBSETUP.SEQ

Sets up variables and defines macro functions (FCTs) for YYB* macros. Must be run first before running any other YYB* macros.

```
in yybsetup [num_windows [num_elements]]
Def:            1             25
```

YYBSPC.SEQ

Macro to define specification data for y-ybar models. Commands are three letter mnemonics, similar to standard SPC data ('epd', 'nao', 'yob', 'yan'). Calls YYBHELP, YYBCONV, and YYBPLOT.

```
in yybspc 'cmd' value ['cmd' value ['cmd' value]]
Def:      req'd req'd  'xxx'   0    'xxx'   0
```

Geometrical Analysis

DIST.SEQ

Plots a diagram of the two-dimensional distortion for the specified zoom position (default 1) and specified field of view. The default for yfov is xfov. if 'file' is blank (default), the plot file is deleted after display. Refgrid is the reference grid drawn (0 = no grid, 1 = outline only, 2 = full grid, 3 = fiducial marks only). Default for 'color' is 'RED'. Range for num_lines is 2 to 21, default is 11. If 'list' is 'Yes', the distortion output are also listed; default is 'No'.

```
in dist xfov [yfov ['file' [refgrid['color'[num_lines[zoom ['list']]]]]]]
Def:     req'd xfov   ''       2     'RED'     11      1    'No'
```

FIFTHDEF.SEQ

Defines @FIFTH and @FIFTHP (surface), which return the value of fifth-order spherical aberrations and calculate and store all other fifth-order aberrations into global variables. @FIFTHP is specifically for returning pupil aberrations. Arguments are surface number 1 through I (surface numbers outside this range give the sums at the image plane), and zoom position. See "Fifth-order Optimization" on page 25C-12 for examples of how to use this macro in optimization.

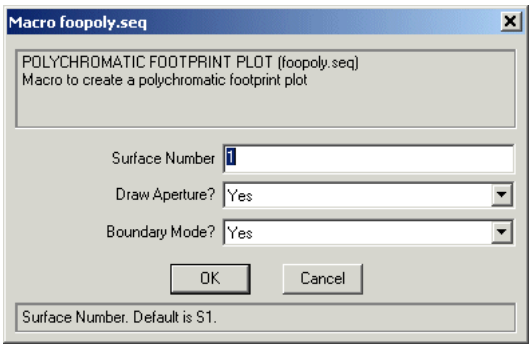
```
in fifthdef
^x == @fifth(6,1)! Image aberration contributions for surface 6, zoom
     ! position 1

^y == @fifthp(6,1)! Pupil aberration contributions for surface 6, zoom
     ! position 1
```

FOOPOLY.SEQ

FOOPOLY (Polychromatic Footprint plot) can be used to automatically perform a footprint analysis at all system wavelengths and plot the results. This macro adds to the functionality provided by the Footprint (FOO) option, which currently traces rays at the reference wavelength only.

```
IN FOOPOLY <surface_number> <draw_aperture_Y/N> <boundary_mode_Y/N>
```

If run from the GUI, CODE V will launch an input dialog box, as shown in the following figure.



FORDER.SEQ

Prints a table of fifth-order aberration contributions for a given surface range (default all surfaces plus image sums). Default for end surface is image surface, default for start surface is 1. Fourth parameter (Y) lists the wave aberrations instead of the transverse aberrations. Fifth parameter (Y) returns pupil aberrations instead of image aberrations. Requires FIFTHDEF to be run first (once per CODE V session).

```
in forder [start_surface [end_surface [zoom_pos [Y [Y]]]]]
```

GHOST_VIEW.SEQ

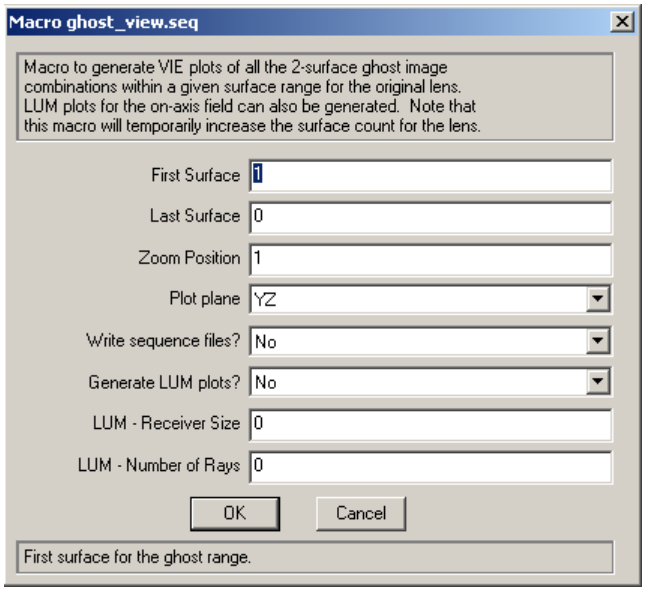
Generates lens view plots of all the two-reflection ghost image combinations within a given surface range. LUM plots for the on-axis field can also be generated. Note that this macro will temporarily increase the surface count for the lens.

```
in ghost_view surface1 surface2 zoom_# VIE_plane WRL? LUM? RCV NRA
```

where:

surface1	–	First surface in ghost range; default is 1
surface2	–	Last surface in ghost range; required entry (0 = SI-1)
zoom_#	–	Zoom position for analysis
VIE_plane	–	Cross-section for plot, either YZ or XZ ; default is YZ
WRL?	–	Either Yes|No; if Yes, write ghost lens setups to .sez files. Default is No.
LUM?	–	Either Yes|No; if Yes, creates on-axis LUM plots for ghosts. Default is No.
RCV	–	Receiver size for LUM plot; default is based on ray trace
NRA	–	Number of rays for LUM plot; default is 50,000

If run from the GUI, CODE V will launch an input dialog box, as shown in the following figure.



LATCOLOR.SEQ

Macro to plot the lateral color vs. field for the specified zoom position (default 1). If 'list_flag' is 'Yes' (default), the output are also listed. A separate plot is made for each zoom position. The wavelength span plotted is from the first to the last wavelengths (for each zoom position) which have non-zero wavelength weights. The lateral color is computed as the chief ray image position of the short wavelength minus the chief ray image position of the long wavelength. If the reference wavelength is not the same as either the short or long wavelength, the difference between the short wavelength and the reference wavelength is also plotted (secondary lateral color).

```
in latcolor [zoom_pos ['list_flag']]
```

LUMSPOT.SEQ

Macro to use the LUM option to get a 2D color raster output of a spot diagram. It only produces output for one field at a time.

```
in lumspot [field [num_rays]]
Def:          1      30000
```

SPOT2D.SEQ

Creates a square array of spot diagrams as a function of field on a single plot. The input syntax for this macro is:

```
in spot2d [ssi [num_fields [max_xfield [max_yfield [min_xfield [min_yfield
[zoo_pos]]]]]]]
```

SPOTDET.SEQ

Draws the outline of a specified detector or Airy disk size on a spot diagram.

```
in spotdet ssi det_x_width|AIRY [det_y_width [nrd [dot [dsi [lab]]]]]
```

Diffraction Analysis

MTFDET.SEQ

Calculates the optical MTF of your system cascaded with the MTF of a specified detector over a range of spatial frequencies. The macro outputs the MTF of the optical system, detector, and the cascaded results. The results are output in both tabular and graphical form.

```
in mtfdet max_freq incr_freq x_det_width [y_det_width [list_flag]]
```

MTFDOF.SEQ

Computes the MTF depth of focus for a specified spatial frequency and MTF value.

```
in mtfdof sp_freq mtf_value
```

MTFTAB.SEQ

Runs the MTF option and lists the MTF performance for all fields in a single table.

```
in mtftab mfr ifr [plot_code [nrd [sqw_flag [imx [imy]]]]]
```

where:

`mfr`	Required. Maximum frequency (line pairs/mm) for computation, measured on the image surface.
`ifr`	Required. Increment in frequency (line pairs/mm) between each successive computation of response, measured on the image surface.
`plot_code`	Plot response versus frequency graphs (Default: Yes).
`nrd`	Number of rays across diameter (Default: 60).
`sqw_flag`	Square wave response (Default: No).
`imx`	X component of image motion (in millimeters) for all wavelengths (Default: 0).
`imy`	Y component of image motion (in millimeters) for all wavelengths (Default: 0).

MTFVSFLD.SEQ

Plots MTF vs. Y-field specification for up to three given spatial frequencies for a given zoom position (zoom default is 1).

```
in mtfvsfld freq1 [freq2 [freq3 [zoom]]]
Def:        req'd    0      0      1
```

OFEXPORT.SEQ

The ofexport.seq macro exports the complex amplitude data from BPR and BSP to a standard CODE V data exchange (.dat) file.

```
in ofexport buffer_number filename
```

where:

buffer_number	–	Specifies the worksheet buffer number where the BPR data has been saved.
filename	–	The name of the .dat file into which the data is written.

Go to "Importing and Exporting Optical Field Data" on page 19-49 for more details about using this macro.

PSFPLOT.SEQ

Macro to plot the PSF cross-section in the X or Y direction. All fields and defocus positions are plotted on the same plot. Each zoom position is plotted on a separate plot. The PSF must be run first, with the output sent to buffer B1 (WBF B1). The field to plot can be specified (default = 0 = all fields). The width of the plot can be specified (default TGR -1). The cross-section can be offset by the specified number of GRI pixels.

```
in psfplot ['x'|'y' [field [width [offset ['NOR'|'LOG']]]]]
Def:          'x'      0      0      0        'NOR'
```

Photonics Analysis Macros

CEF_VS_FOCUS.SEQ

CEF_VS_FOCUS analyzes fiber coupling efficiency through focus and plots the results.

```
in CEF_VS_FOCUS '<CEF command string>' [Y|N] [NFO] [FFO] [IFO] [Y|N]
```

where:

CEF command string	–	Series of CEF commands to be executed.
Y|N	–	Specifies whether to define through-focus values (Y), or use LDM values (N).
NFO	–	Number of focus values.
FFO	–	First focus position.
IFO	–	Focus increment.
Y|N	–	Specifies whether to apply new NFO value to the lens (Y), or not (N).

For example:

```
in CEF_VS_FOCUS 'CEF;MPR GAU .0052;GO' 'Y'  5  -.1  .05
```

CEF_VS_WL.SEQ

CEF_VS_WL analyzes fiber coupling efficiency through wavelength and plots the results.

```
in CEF_VS_WL '<CEF command string>' num_wavelengths short_wavelength
wavelength_increment
```

where:

CEF command string	–	Series of CEF commands to be executed.
num_wavelengths	–	Number of wavelengths.
short_wavelength	–	Shortest wavelength.
wavelength_increment	–	Wavelength increment.

For example:

```
in CEF_VS_WL 'CEF;MPR GAU .0052;GO'  5  1530  10
```

DB_CONVERT.SEQ

DB_CONVERT is a new utility macro that provides a quick conversion to and from fraction and dB.

```
in DB_CONVERT value
```

where the value (required) is either the coupling efficiency or insertion loss. If the value is positive, it is converted to insertion loss; if negative, it is converted to coupling efficiency.

GAUSS_APOD_SINE.SEQ

GAUSS_APOD_SINE calculates an apodization INT file that is Gaussian in the sine of the input angle (CODE V Gaussian apodization is Gaussian in the tangent of the input angle).

```
in GAUSS_APOD_SINE NAOx NAOy xoffset yoffset zx zy pupil_size filename pixels
```

where:

NAOx	–	Numerical aperture of the source in the X direction; default is 0.4
NAOy	–	Numerical aperture of the source in the Y direction; default is 0.4
xoffset	–	Offset of the pattern in the X direction; default is 0.0

yoffset	–	Offset of the pattern in the Y direction; default is 0.0
zx	–	Distance (absolute) from the apodized surface to the X object waist; default is 1.0
zy	–	Distance (absolute) from the apodized surface to the Y object waist; default is 1.0
pupil_size	–	Physical size of the INT file (radial); default is 0.4364
filename	–	File name of the INT file, including pathname; default is 'GAUSS.INT'
pixels	–	Size of the INT file in pixels

GRIN_COUPLE.SEQ

GRIN_COUPLE takes a set of previously calculated fiber modes for a GRIN fiber and calculates the individual and total coupling efficiency.

```
in GRIN_COUPLE buffer_number base_file_name
```

where:

buffer_number	–	Number of the buffer that contains the complex amplitude file (typically generated using BPR).
base_file_name	–	Base name of the files that contain the fiber b values and calculated mode profile data (generated by GRIN_FIBER.SEQ).

GRIN_FIBER.SEQ

GRIN_FIBER takes the physical specification of a GRIN fiber and generates the amplitude mode profile of all the supported modes at a given wavelength. It then writes out an INT file for each mode. These files can be used for CEF calculations.

```
in GRIN_FIBER wavelength core_radius core_index parabolic_coeff file_size
factor file_name header options
```

where:

wavelength	–	The wavelength at which the modes will be calculated in microns.
core_radius	–	Radius of the fiber core in microns.
core_index	–	Index of the fiber core.
parabolic_coeff	–	Parabolic index coefficient: n1 = n0*(1 - 2*D*(r/a)^2)^1/2
file_size	–	Size of one dimension of the .int files generated. The generated .int files are square.
factor	–	Factor to determine the array semi-diameter: ^factor*^a = array semi-diameter
file_name	–	The base filename for the .int files. This string includes the pathname, but no extension.
header	–	The base header name for labeling the .int files.
options	–	String for determining which file types are written and if graphs are given for each mode. The string is a 4-letter input, either a Y or an N. The letters stand for Amplitude, Intensity, Phase, and Graphs, in that order.

INPUT_FIBER_APOD.SEQ

INPUT_FIBER_APOD computes the proper pupil apodization given either a numerical aperture, or an initial wavelength and Gaussian waist. It assumes a fiber in air.

```
in INPUT_FIBER_APOD naox naoy waistx waisty wavelength intensity
```

where:

naox, naoy	–	The $1/e^2$ numerical aperture of the fiber; if this value is left 0, then the macro looks for the wavelength and waist and calculates the NAO.
waistx, waisty	–	Gaussian waist at the object/fiber. Not needed if the NAO is specified.
wavelength	–	Wavelength used to calculate the nao, not needed if NAO is specified.
intensity	–	Intensity at the edge of the pupil for the largest NAO.

MULTIMODE_COUPLE.SEQ

MULTIMODE_COUPLE takes a set of previously calculated fiber modes for a step index fiber and calculates the individual and total coupling efficiency.

```
in MULTIMODE_COUPLE buffer_number base_file_name
```

where:

buffer_number	–	Number of the buffer that contains the complex amplitude file (typically generated using BPR).
base_file_name	–	Base name of the files that contain the fiber b values and calculated mode profile data (generated by MULTIMODE_FIBER.SEQ).

MULTIMODE_FIBER.SEQ

MULTIMODE_FIBER takes the physical specification of a step index fiber and generates the amplitude mode profile of all the supported modes at a given wavelength. It then writes out an INT file for each mode. These files can be used for CEF calculations.

```
in MULTIMODE_CALC wavelength core_radius core_index cladding_index factor
file_size file_name header
```

where:

wavelength	–	Wavelength at which the modes will be calculated in microns
core_radius	–	Radius of the fiber core in microns
core_index	–	Index of the fiber core
cladding_index	–	Index of the fiber cladding.
factor	–	Factor to determine the array semi-diameter
file_size	–	Size of one dimension of the INT files generated. The generated INT files are square.
file_name	–	Base filename for the INT files. This string includes the pathname, but no file extension.
header	–	The base header name for labeling the INT files.

Polarization Analysis Macros

The Physics Department of the University of Alabama, Huntsville has written a series of macros for polarization analysis of optical systems. These macros enhance the output available from the RSI polarization ray trace. They allow the propagation of polarized light to be displayed as Jones vectors, Stokes vectors, or 3-D polarization vectors. The polarization properties of the ray paths can be displayed as Jones matrices, Mueller matrices, or polarization matrices, and the diattenuation, retardance, eigenvalues, and eigenpolarizations are tabulated. The basis vectors used for describing the X and Y components and the s and p components are available also.

Before using these macros, polarization ray tracing must be enabled first (POL Y) and the polarization state must be defined (unpolarized, or polarized as defined by geometrical parameters, Jones vectors, or Stokes parameters). For all these macros (except PupAvg_Stokes_FWZ and PAVE_MM), an RSI must be run first. This can be done for some of them by running the POLAR macro, which then gives a menu for further processing.

DRTAB.SEQ

Calculates and tabulates cumulative diattenuation and retardance associated with a single ray at each surface.

```
in drtab
```

EPJTAB.SEQ

Calculates and tabulates the eigenvalues and eigenpolarizations as Jones vectors from object space through each surface for a single ray.

```
in epjtab
```

EPSTAB.SEQ

Calculates and tabulates the eigenpolarization vectors in EPJTAB into Stokes vectors for a single ray. EPJTAB must be run first.

```
in epstab
```

IJTAB.SEQ

Tabulates i and j basis vectors calculated by POLAR for each surface. These are the basis vectors which define the X and Y components of all Jones, Stokes, and geometrical polarization parameters. Surface Jones matrices and Mueller matrices are defined respect to the basis vectors on the incident side of the surface for incoming light and on the exiting side of the surface for emergent light. Cumulative Jones matrices and cumulative Mueller matrices are defined with respect to the object space i and j basis vectors and the i and j basis vectors on the exiting side of each surface.

```
in ijtab
```

JMCTAB.SEQ

Multiplies surface Jones matrices calculated by POLAR for a single ray to produce cumulative Jones matrices for each surface.

```
in jmctab
```

JMTAB.SEQ

Tabulates surface Jones matrices computed by POLAR for a single ray. Jones matrices relate input and output Jones vectors.

```
in jmtab
```

JVEC.SEQ

Produces a table of Jones vectors at each surface for a ray specified by RSI, using the incident polarization state specified in the LDM.

```
in jvec
```

MMCTAB.SEQ

Multiplies surface Mueller matrices calculated by POLAR for a single ray to produce cumulative Mueller matrices for each surface.

```
in mmctab
```

MMTAB.SEQ

Tabulates surface Mueller matrices computed by POLAR for a single ray. Mueller matrices relate input and output Stokes vectors.

```
in mmtab
```

PAVE_MM.SEQ

Calculates the pupil averaged Mueller matrix by tracing an 11 x 11 ray grid uniformly over the pupil, using the polarization state specified for the last field.

```
in pave_mm x_fract_max_x_field y_fract_max_y_field
```

PupAvg_Stokes_FWZ.SEQ

Calculates pupil-averaged Stokes vector for a specified field by tracing 11 x 11 rays uniformly distributed in the pupil. Rays with non-zero BLS and RER are excluded.

```
in PAVE_SV [f] [w] [z]
```

PMCTAB.SEQ

Multiplies surface polarization ray tracing matrices calculated by POLAR for a single ray to produce cumulative polarization ray trace matrices for each surface.

```
in pmctab
```

PMTAB.SEQ

Tabulates surface 3-D polarization matrices calculated by POLAR for a single ray.

```
in pmtab
```

POLAR.SEQ

Calculates and stores various polarization related quantities for a ray specified by RSI. These quantities include i and j basis vectors, s and p vectors, surface and cumulative Jones matrices, surface and cumulative Mueller matrices, and surface and cumulative polarization matrices. This macro presents a menu for the user to select from additional polarization macros for calculation and output. The following 12 macros can only be run after running POLAR.

```
in polar
```

POLDSP.SEQ

Produces a plot of the polarization state across the pupil. A separate plot is produced for each field and zoom position. The map is for the reference wavelength, unless a different wavelength number is specified. The default is to plot for five points across the pupil, unless a different number is specified.

```
in poldsp [wavelength_num [nrd [type [output]]]]
```

where:

wavelength_num	–	Wavelength number (default is the reference wavelength, which will also be used if 0 is entered).
nrd	–	Number of rays across the pupil (default: 5)
type	–	Type of output plot: R for retardance D for diattenuation Any other character plots the polarization state (default).
output	–	Text output control: F for full S for summary N for none

PVTAB.SEQ

Calculates the surface 3-D polarization vectors for a single ray and polarization state. The macro prompts for an incident Jones vector.

```
in pvtab
```

SPTAB.SEQ

Tabulates s and p vectors which establish the s and p components of the light at each surface.

```
in sptab
```

SVEC.SEQ

Produces a table of Stokes vectors at each surface for a ray specified by RSI, using the incident polarization state specified in the LDM.

```
in svec
```

Fabrication Support

CHINAPLOT.SEQ

Creates a lens drawing according to the Chinese national standard. This macro only works for spherical, conic, or aspheric surfaces, and assumes a circular aperture.

```
in Chinaplot start_surf [dia [scale_fac ['save_file'  ['data_file' [x_offset
[y_offset ['co_name' [frame&table]]]]]]]
```

where:

start_surf	–	First surface of the lens or reflector (required).
dia	–	Edge diameter (default: 0).
scale_fac	–	Plot scale factor.
'save_file'	–	Filename for plot file. If blank (default), plot is deleted after being displayed. Default file extension is .plt.
'data_file	–	Filename for data file containing inputs. If blank (default), the macro prompts for inputs. Default file extension is .dat.
x_offset	–	Offset in X of plot (in inches). Default: 0.0.
y_offset	–	Offset in Y of plot (in inches). Default: -0.5.
'co-name'	–	Company name, 20-character maximum. Default: 'ORA'
frame&table	–	1 for both, 2 for frame only, 3 for table only, 4 for none.

ISOPLOT.SEQ

This macro makes a lens drawing in ISO 10110 format. It only works for spherical, conic, or aspheric surfaces, and assumes a circular aperture. Surface is the number of first surface of the element; dia is the outer edge diameter (default = 0 = add default increment); scale is the drawing scale (default = 0 = autoscale); 'save_file' is the name of the plot file to save the drawing in; 'data_file' is an optional file with inputs for the macro (if it is blank, you are prompted for inputs); x_offset and y_offset are offsets for the plot on the hard copy. The macro ISODEFS must be run first. A sample data file is stored in ISODATA.DAT (on the CV_MACRO directory).

```
in isoplot surface [dia [scale ['save_file' ['data_file'
                   [x_offset [y_offset]]]]]]
```

LENSTABLE.SEQ

Creates a table of lens construction parameters including diameters, sags, real edge thicknesses (and CODE V reference ray based edge thicknesses), and diameter/thickness ratio. This macro will optionally define clear and edge apertures.

```
IN LENSTABLE [ca_increment] [od_increment]
[air_increment][aperture_set_flag]
```

where:

ca_increment	–	Diametrical increment between optically used aperture and clear aperture (CA) definition.
od_increment	–	Diametrical increment between largest CA, and outer diameter of the lens.
air_increment	–	Diametrical increment between largest CA straddling an airspace, and the barrel diameter.
aperture_set_flag	–	Y|N. Default No, if Yes, macro sets clear and edge apertures for all non-dummy surfaces without user-defined apertures.

User-Defined Tolerancing

Two macros, TOLFDIF and TOLMONTE, allow you to perform a sensitivity tolerance analysis (SNS mode) on your lens based on any user-defined quality or performance measure, such as encircled energy, energy on a detector, f-theta scan linearity, etc. Any performance measure that you can compute in CODE V (using options and/or Macro-PLUS processing), can be used as a tolerance criterion. The TOLFDIF macro perturbs the lens for each individual tolerance. This allows the determination of tolerances that are performance drivers. The TOLMONTE macro perturbs the lens randomly by all the defined tolerances, and re-evaluates it to determine the statistical distribution of performance (but does not include any information about which tolerances are performance drivers).

For details about TOLFDIF and TOLMONTE, see “User-Defined Tolerancing” on page 20-107.

User-Defined Features Macros

CODE V has several user-defined features to supplement the built-in forms. These include the ability of the user to program a user-defined surface (four forms, UDS, UD1, UD2, and UD3), interferogram, a user-defined gradient index (UDG), two user-defined diffractive phase functions (HCT USR and HCT US2), a user-defined INT file (UDI), and user-defined surface properties (USP). These user-defined features are externally programmed in either Fortran, C, or another programming language, and linked into CODE V with an ORA-supplied linking routine. The linking processes and usage of the routines are described in detail in “Creating User-Defined Features” on page 1-96.

These user-defined capabilities can also be programmed in Macro-PLUS as user-defined macro functions (UDFs), as an alternative to requiring external Fortran or C subroutines. There are two main advantages to using Macro-PLUS for these capabilities. The first is the ease of modeling and testing of such routines before programming them in Fortran or C, and the second is the ability to program and use these features when you do not have an external compiler. The disadvantage is that these features run significantly slower when programmed in Macro-PLUS than in their corresponding Fortran or C forms.

These UDFs are programmed like any other user-defined macro function. They have no required naming convention; for example, you could name your UDS functions as follows: myuds1, myuds2, etc. CODE V provides default macro functions for each of the user-defined feature types: @USERSUR, @USERSUR1, @USERSUR2, @USERSUR3, @USERGRN, @USERHOE, @USERINT, and @USERUSP. Before you can apply a user-defined macro function to a surface, you must compile it in CODE V; this is done with the IN command.

For details about the search order CODE V uses for user-defined features, go to “How CODE V Searches for User-Defined Features” on page 1-98.

In each of these macro UDFs, there are several passed parameters. They must be passed in the order specified, and the arrays must be the same size as shown below. The actual names of the variables can be different, as long as the corresponding variables are the correct size and in the correct order. In each UDF there are input parameters which cannot be changed by the user, such as position, curvature, direction cosines, and coefficient arrays; these input parameters are computed by CODE V and passed to the UDF routine. Only the output arrays, generally here named ^f and ^flags, are changed by the user in the routine.

User-Defined Surface (UDS)

The user-defined surface (UDS) is defined by the following UDF:

```
FCT @USERSUR(num ^x, num ^y, num ^z, num ^curv, num ^ucodata(84), num ^f(4),
num ^flags(2))
```

x, y, z	–	The coordinates of a point along the ray
curv	–	The base curvature of the surface (need not be used in the calculations, but is used for paraxial calculations).
ucodata	–	An 84-element input array corresponding to the UCO coefficients C1 to C84
f	–	A 4-element output array with the following components f(1) the evaluated function value f(2) the derivative of the surface in x (df/dx) f(3) the derivative of the surface in y (df/dy) f(4) the derivative of the surface in z (df/dz)
flags	–	A 2-element output array with the following components flags(1) indicates whether derivatives are computed 0 = no (use finite differences) 1 = yes (derivatives are programmed) flags(2) indicates an error in the calculation 0 = no error non-zero is the sign of the function value at the location where the function will evaluate

Sample UDS Sequences

Two sample UDS macro functions are supplied with CODE V. Each one corresponds to the Fortran and C subroutines supplied with CODE V.

USERSUR.SEQ

This macro function duplicates the default USERSUR subroutines supplied with CODE V. It defines a surface by standard Zernike coefficients (up to sixth order). The syntax is

```
in usersur
UDS Sk
```

The UCO coefficients are as follows:

UCO	C1	k
UCO	C2..C29	Zernike coefficients Z1..Z28

UDSFRESN.SEQ

This macro function defines a finite width Fresnel surface defined by a tenth order polynomial aspheric. Each zone has equal widths, and is flat with slope corresponding to the slope of the center of the zone. The effects of the back cuts are ignored. This type of surface is modeled more completely with the default UD2 surface. The syntax is

```
in udsfresn
UDS Sk
```

The UCO coefficients are as follows:

UCO	C1	k
UCO	C2	A
UCO	C3	B
UCO	C4	C
UCO	C5	D
UCO	C6	zone width (default = 0.1)

User-Defined Surface Type 1 (UD1)

The user-defined surface type 1 is similar to the UDS surface, except that UD1 supports unlimited coefficients, where UDS has a limit of 81 coefficients. The UD1 surface is defined by the following UDF:

```
FCT @USERSUR1 (NUM ^X,NUM ^Y,NUM ^Z,NUM ^CURV,&
           NUM ^OUT_F(4), NUM ^FLAGS(2), num ^COEFS(84), num ^numcoefs)
```

x, y, z – Coordinates at a point along the ray.

curv – Surface vertex curvature.

f – A 4-element output array with the following components

- f(1) the evaluated function value
- f(2) the derivative of the surface in x (df/dx)
- f(3) the derivative of the surface in y (df/dy)
- f(4) the derivative of the surface in z (df/dz)

flags – A 2-element output array with the following components

- flags(1) indicates whether derivatives are computed
  - 0 = no (use finite differences)
  - 1 = yes (derivatives are programmed)
- flags(2) indicates an error in the calculation
  - 0 = no error
  - non-zero is the sign of the function value at the location where the function will evaluate

coefs – Input data array containing the UMO coefficients for the current zoom position. Double precision.

numcoefs – Integer; number of coefficients in the coefficient data array.

Sample UD1 Sequence

A sample UDS macro function, USERSUR1, is supplied with CODE V.

User-defined Surface (UD2)

The user-defined surface (UD2) is defined by the following UDF:

```
FCT @USERSUR2 (num ^x, num ^y, num ^z, num ^curv, num ^ucodata(84), num ^f(4),
num ^flags(5) ^num ^el, num ^em, num ^en, num ^lchg,
num ^current_opl, num ^work (see note on ^work below for size))
```

x, y, z – The coordinates of a point along the ray

curv – The vertex curvature of the surface (does not need to be used in the calculations, but is used for paraxial calculations).

ucodata – An 84-element input array corresponding to the UCO coefficients C1 to C84

f – A 4-element output array with the following components

- f(1) the evaluated function value
- f(2) the derivative of the surface in x (df/dx)
- f(3) the derivative of the surface in y (df/dy)
- f(4) the derivative of the surface in z (df/dz)

flags – A 5-element output array with the following components

- flags(1) indicates whether the derivatives are programmed
  - 0 the derivatives are programmed
  - 1 derivatives are not programmed (use finite differences)
- flags(2) indicates an error in the surface calculation
  - 0 no error

  non-zero is the sign of the function value at the location where the function will evaluate
- flags(3) the intersection number (I/O flag)
- flags(4) the miss code
- flags(5) -1 if a new lens has been input since the last call to this routine. 0 if no changes have been made to any lens parameters since the last call to this routine. +1 if one or more lens parameters of the current lens have been changed since the last call to this routine. This flag permits USERSUR2 to perform initialization calculations only when something about the lens changes. Should be set back to 0 after its input value is checked.

el, em, en – The x, y, z direction cosines of the ray

lchg – A flag to tell USERSUR2 when the lens has changed

current_opl – The reduced distance (t/n) from the prior surface to the current ray position

work – A user-requested persistent work vector. Currently, this work vector must be defined for a size of 2, regardless of the actual UD2 surface implementation.

Sample UD2 Sequence

There is only one sample USERSUR2 macro sequence supplied with CODE V. It corresponds to the Fortran and C versions supplied with CODE V.

USERSUR2.SEQ

This macro function duplicates the type 1 default USERSUR2 subroutines supplied with CODE V. It defines a Fresnel surface on a general toroidal substrate. Refer to “Creating User-Defined Features” on page 1-96 for details on how to use this surface. The syntax is

```
in usersur2
UD2 Sk 2    !Where 2 (after the surface Sk) is the work vector.
            !The work vector 2 is required, regardless of the actual UD2
            !surface implementation

The UCO coefficients are as follows:
```

UCO C1	RDX (X radius of curvature; 0 means flat)
UCO C2	s (slope of surface relative to y-axis, entered as tangent of the angle)
UCO C3	k (conic constant)
UCO C4	A
UCO C5	B
UCO C6	C
UCO C7	D
UCO C8	E
UCO C9	F
UCO C10	G
UCO C11	H
UCO C12	I
UCO C13	J
UCO C14	Distance from the surface vertex to the common origin of the back cut zones.
UCO C15	Maximum number of discrete zones to allow in the positive Y direction (Fresnel zones and back cut zones are counted separately). Rays which strike the surface at a height corresponding to a larger zone number are treated as if they fail by missing the surface.
UCO C16	Maximum number of discrete zones to allow in the negative Y direction.
UCO C76	0 or 1
UCO C78	Zone sag at the edge of all zones except the center zone. Sag in the YZ plane is defined as perpendicular to the line with slope given by UCO C2 that passes through the surface vertex.
UCO C79	Zone sag at the edge of the center zone.

User-Defined Surface Type 3 (UD3)

The user-defined surface type 3 (UD3) is defined by the following UDF:

```
FCT @USERSUR3 (num ^X,num ^Y,num ^Z,num ^CURV,&
               num ^f(4), num ^flags(5),&
               num ^el,num ^EM,num ^EN, num ^CUR_PL,num ^WORK(613), &
               num ^uco_data(84), num ^numCoefs)
```

x, y, z – The coordinates of a point in space along the ray.

curv – The vertex curvature of the surface.

f – A 4-element output array with the following components

- f(1) the evaluated function value
- f(2) the derivative of the surface in x (df/dx)
- f(3) the derivative of the surface in y (df/dy)
- f(4) the derivative of the surface in z (df/dz)

flags – A 5-element output array with the following components

- flags(1) indicates whether the derivatives are programmed
  - 0 the derivatives are programmed
  - 1 derivatives are not programmed (use finite differences)
- flags(2) indicates an error in the surface calculation
  - 0 no error
  - non-zero is the sign of the function value at the location where the function will evaluate
- flags(3) the intersection number (I/O flag)
- flags(4) the miss code
- flags(5) -1 if a new lens has been input since the last call to this routine. 0 if no changes have been made to any lens parameters since the last call to this routine. +1 if one or more lens parameters of the current lens have been changed since the last call to this routine. This flag permits USERSUR2 to perform initialization calculations only when something about the lens changes. Should be set back to 0 after its input value is checked.

el, em, en – The optical direction cosines of the ray to iterate; $el^2 + em^2 + em^2 = n^2$ where n is the refractive index of the medium of the incoming ray.

cur_pl – The reduced distance from the prior surface to the current ray position

work – A user-requested persistent work vector. 613 elements are allocated by default, but with the UD3 command, the user can request as many locations as needed, within the limitations imposed by the system resources.

uco_data – Input data array containing the UMO coefficients for the current zoom position. Double precision.

numCoefs – Integer; number of coefficients in the coefficient data array.

Sample UD3 Sequence

A sample USERSUR3 macro sequence is supplied with CODE V.

User-Defined Gradient Index (UDG)

A user-defined gradient index is defined by the following UDF:

```
@FCT @USERGRN(num ^brind, num ^udgdata(20), num ^s(3), num ^f(5))
```

brind	–	base refractive index
udgdata	–	A 150-element array corresponding to the UDG coefficients C1 to C150
s	–	A 3-element vector giving the x, y, z location in the gradient medium
f	–	A 5-element output array with the following components:

f(1)	error flag
	0 = no error
	non-zero = error
f(2)	calculated index of refraction
f(3)	the x component of n*grad(n) = n*dn/dx
f(4)	the y component of n*grad(n) = n*dn/dy
f(5)	the z component of n*grad(n) = n*dn/dz

Sample UDG Sequence

Only one sample user-defined gradient index macro function is supplied with CODE V. It corresponds to the Fortran and C versions supplied with CODE V.

USERGRN.SEQ

This macro function models a Luneberg type of gradient index (LUN type). The refractive index is computed from the equation

$$n^2(r) = n_0{}^2\left(2 - \frac{p^2}{a^2}\right)$$

where $p^2 = x^2 + y^2 + (z - r)^2$ and r is the distance from the point of symmetry. The values of n0, a, and r must be defined for each wavelength used in the Private Catalog definition. If only one value of a or r is entered, it will be used at all wavelengths.

```
in usergrn
PRV
PWL  w1 w2...! wavelengths
'glass_label'  n01 n02...! base index
UDG
UDG C1  a1  a2...! parameter a
UDG C2  r1  r2...! parameter r
```

User-defined HOE Aspheric Phase (HCT USR)

A user-defined HOE aspheric phase profile is defined by the following UDF:

```
@FCT @USERHOE(num ^x, num ^y, ^hcodata(65), num ^f(4))
```

x, y	–	The x and y coordinates on the surface
hcodata	–	A 65-element input array corresponding to the HCO coefficients C1 to C65
f	–	A 4-element output array with the following components

f(1)	the value of the phase function of the HOE (in lens units)
f(2)	the derivative of the HOE phase in x
f(3)	the derivative of the HOE phase in y
f(4)	the equivalent quadratic term for the paraxial power of the HOE

Sample USERHOE Sequence

Only one sample user-defined HOE aspheric phase profile macro function is supplied with CODE V. It corresponds to the Fortran and C versions supplied with CODE V.

USERHOE.SEQ

This macro function defines a rotationally symmetric HOE aspheric phase. It duplicates the HCT AR form of the aspheric phase.

```
in userhoe
HCT Sk USR
```

The HCO coefficients are as follows:

HCO	C1	coefficient of r
HCO	C2	coefficient of r^2
...		
HCO	C20	coefficient of r^{20}

User-defined HOE Aspheric Phase Type 2 (HCT US2)

A user-defined HOE aspheric phase profile type 2 is defined by the following UDF:

```
FCT @USERHOE2 ( num ^x, num ^y, num ^f(4), num ^hco_data(84), num ^numCoefs)
```

x, y	–	x,y coordinates on the HOE substrate.
f	–	A 4-element output array with the following components

f(1)	the value of the phase function of the HOE (in lens units)
f(2)	the derivative of the HOE phase in x
f(3)	the derivative of the HOE phase in y
f(4)	the equivalent quadratic term for the paraxial power of the HOE

hco_data	–	Input data array containing the UMO coefficients for the current zoom position. Double precision.
numCoefs	–	Integer; number of coefficients in the coefficient data array.

Sample USERHOE2 Sequence

A sample user-defined HOE aspheric phase profile type 2 macro function is supplied with CODE V.

User-defined INT File (UDI)

A user-defined INT file is defined by the following UDF:

```
FCT @USERINT (num ^x, num ^y, num ^el, num ^em, num ^en, num ^cosi,
num ^udi_data(2), num ^f(3), num ^flags(2))
```

The @USERINT macro function evaluates the function and the derivatives for a user-defined INT file. The following is a description of the parameters in the call list. If the parameter is designated as “input,” its value is passed to the subroutine by the calling program; if it is designated as “output,” its value is calculated or set by this subroutine and passed back to the calling program.

x, y	–	Coordinates of the current position of the ray with respect to the origin of the surface (input from CODE V)
el, em, en	–	Geometric direction cosines of the ray (input from CODE V)
cosi	–	Cosine of angle of incidence (input from CODE V)
udi_data	–	Array containing the user-defined interferogram parameters (input from user); they are the data contained in the INT file.
f(1)	–	Value of the function (output to CODE V)
f(2), f(3)	–	Values of the (x , y) derivatives of the function (output to CODE V); (Not needed for FIL interferogram files).
flags(1)	–	Error code; nonzero if there is an error in the function calculation (output to CODE V)

Sample USERINT Sequence

Only one sample user-defined INT file macro function is supplied with CODE V. It corresponds to the Fortran and C versions supplied with CODE V.

USERINT.SEQ

This macro function defines a Gaussian intensity apodization filter. The syntax is

```
in userint
INT  Sk  Gaussian
```

The INT file entitled Gaussian.int has the following contents:

Gaussian apodization

```
UDI  1      FIL    SSZ    1.0
1.5
```

User-Defined Surface Properties (USP)

A user-defined surface properties function is defined by the following UDF:

```
FCT @USERUSP (NUM ^RAY_POS(3), NUM ^RAY_DIR_BEFORE(3),
NUM ^RAY_DIR_AFTER(3),NUM ^SURF_NORMAL(3), NUM ^LCL_GRATING_VECTOR(3), NUM
^DIFFRACTED_ORDER, NUM ^WAVELENGTH, NUM ^INDEX_BEFORE,
NUM ^INDEX_AFTER, NUM ^POL_TRACING, NUM ^JONES_MATRIX(8),
NUM ^USP_OTHER_DATA(4), NUM ^JONES_COEFFS(100), NUM ^JONES_NUM_COEFFS)
```

ray_pos – x,y,z coordinates of the point at which the ray intersects the surface

ray_dir_before – Direction cosines of the ray that is incident on the surface

ray_dir_after – Direction cosines of the ray after reflection/refraction or diffraction. The value of ray_dir_after will always be set prior to calling this routine. It will be set to the ray direction that is computed based on the standard raytracing equations. If ray_dir_after is modified within this function, then that modified ray direction will be picked up for the ray as it exits this surface.

surf_normal – Vector that is normal to the surface at the point at which the ray intersects the surface.

lcl_grating_vector – Grating vector at the intersection point of the ray with the diffractive surface.

diffracted_order – Diffracted order for the ray traced from a diffracted interface

wavelength – Wavelength of light in nanometers

index_before – Index of refraction of the medium in which the incident ray propagates

index_after – Index of refraction of the medium that is on the opposite side of the interface from the incident ray. This value will be set to zero for REFL surfaces in sequential surface ranges

pol_tracing – Nonzero if polarization raytracing is active

jones_matrix – (Output to CODE V.) Eight elements that represent the four complex elements of the Jones matrix for the device. The order of the elements is:
^JONES_MATRIX(1) –> Real(J11)
^JONES_MATRIX(2) –> Imag(J11)
^JONES_MATRIX(3) –> Real(J12)
^JONES_MATRIX(4) –> Imag(J12)
^JONES_MATRIX(5) –> Real(J21)
^JONES_MATRIX(6) –> Imag(J21)
^JONES_MATRIX(7) –> Real(J22)
^JONES_MATRIX(8) –> Imag(J22)

ncoordinatetype – Allowable values are 0, 1, or 2:

0 Device Jones matrix is applied in a PCS COL coordinate system

1 Device Jones matrix is applied in a PCS STD coordinate system

2 Device Jones matrix is applied in an S/P coordinate system.

usp_other_data – (Output to CODE V)
^USP_OTHER_DATA(1)
nCoordinateTyp - Indicates which coordinate system the Jones matrix should be applied in. The possibilities are:
0 - Device Jones matrix is applied in a PCS COL coordinate system
1 - Device Jones matrix is applied in a PCS STD coordinate system
2 - Device Jones matrix is applied in an S/P coordinate system.
^USP_OTHER_DATA(2)
dOPL - The OPL increment (waves) added by this device
^USP_OTHER_DATA(3)
dTransmission - The device transmittance. (The true transmittance of the device will also include the effects of the Jones matrix values.)
^USP_OTHER_DATA(4)
nErrorFlag - An error flag. A value of zero indicates that no errors occurred during the processing within this MACRO.

jones_coeffs – Array of coefficients that the user can set within CODE V. Note that there is a limit of 100 coefficients.

jones_num_coeffs – Largest non-zero coefficient number

Sample USERUSP Sequence

The sample USERUSP sequence supplied with CODE V implements a quarter-wave retarder with a horizontal fast axis. The syntax is

```
in userusp
POP Sk USP
```

Macros to Define User-Defined Macro Functions

These macros all define a user-defined macro function. Each macro can be run only once per CODE V session. After running them, the macro function is used in a similar manner to any other predefined macro function.

To access sample macros that define user-defined macro functions, choose the **Tools** > **Macro Manager** menu, go to the **Sample Macros** list, and select the **Macro Functions** category.

DEFINE_JMRCC.SEQ

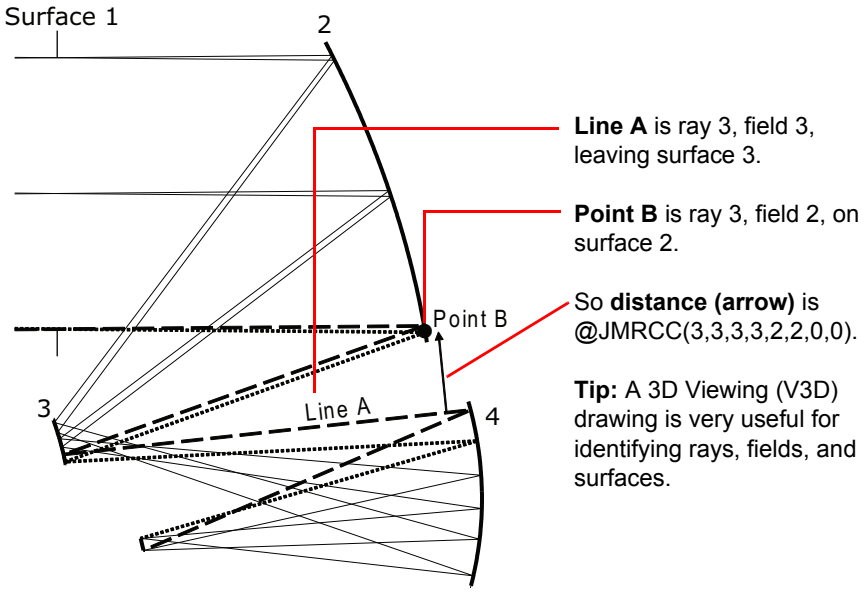
Defines the function @JMRCC, which computes the signed perpendicular distance (in the YZ plane) from a line A to a point B. @JMRCC is useful for creating clearance or interference constraints when you are optimizing off-axis reflective systems, such as three-mirror anastigmat (TMA) configurations.

There are eight integer arguments. The syntax is as follows:

```
in cv_macro:DEFINE_JMRCC
^clearance == @JMRCC(RA,FA,SA,RB,FB,SB,RC,FC)
```

The first six function arguments are reference ray number, field number, and surface number, first for the line A and then for the point B. The line A is a reference ray (ray RA) at field FA after it leaves a specified surface SA, and the point B is the intersection of any reference ray (ray RB) at field FB with a given surface SB, or optionally the intersection of two reference rays. The last two arguments, if non-zero, are the reference ray number RC and field number FC of a second line; the "point" is then defined to be the intersection of the reference ray RB at field FB leaving surface SB with reference ray RC at field FC leaving the same surface SB. The sign convention is such that for a ray (line A)

propagating generally in the +Z direction (N direction cosine > 0), the signed distance is positive if the "point" is on the +Y side of the line, as shown in the following figure.



The function returns a value that is the signed distance from point B to line A, measured perpendicular to the line.

Example

The following commands show use of the @JMRCC macro function for clearance control in an unobscured TMA.

```
in cv_macro:define_jmrcc
res cv_lens:threemir
yde s2 (yde s2)-(yde s1)
del dda s1  ! deletes superfluous decenter on stop
thc s1..2 0  ! these don't need to be coupled
bc s2 0
bc s3 0
bc s4 0
del ape sa
vie;nbr;aap;go
!
aut
!     Use clearance control UDMF in 3 places:
@s3_clr == @jmrcc(3,2,1,2,3,3,0,0)
@s3_clr < -25
@field_stop_clr == @jmrcc(3,1,2,2,3,3,3,3)
@field_stop_clr > 5
@lyot_stop_clr == @jmrcc(2,1,3,3,1,4,3,3)
@lyot_stop_clr < -6
!     Image centration
y r1 f2 si = 0
!    Focal length (image size) constraint
!     Paraxial EFL isn't accurate for tilted system.
@image_height == (y r1 f1 si)-(y r1 f3 si)
@image_height = 4.6
!     You need these to keep system from getting too big.
```

```
@system_length == (z r1 f2 s4 g1)-(z r2 f3 s3 g1)
@system_length < 175
@system_height == (y r2 f3 s2 g1)-(y r2 f1 s4 g1)
@system_height < 200
dra
imp .01
go
vie;nbr;aap;go
```

DEFLENG.SEQ

Defines @LENG(i,j) which is the absolute value of the vertex lengths from surface i to surface j.

```
in defleng
^x == @leng(3,7)
```

DEFMAXB.SEQ

Defines @MAXB which is a maximum value function used in the supplied macro BEAMREAD.

```
in defmaxb
```

EXSTNDEF.SEQ

Defines @EXSTN('string'). This extracts the first number found in the string, and returns the rest of the string into global variable ^STR.

```
in exstndef
^x == @exstn("abc25.7")
```

FLGINT.SEQ

Defines a function @LAGINT to perform a Lagrange interpolation for three points, equally spaced in X. The three Y values are input, plus a normalized X value at which the Y interpolated value is needed. The X value is normalized such that $X_1 = 0$ and $X_3 = 1$. The syntax is

```
in flgint
@lagint(x,y1,y2,y3)
```

FLNINT.SEQ

Defines a function @LININT to perform a linear interpolation. Two points X_1,Y_1 and X_2,Y_2 are input and the interpolated Y point is input. The function returns the X value corresponding to the Y value. The syntax is

```
in flnint
@linit(y,y1,y2,x1,x2)
```

FPEAK.SEQ

Defines a function @PARPEAK which fits a parabola to three points equally spaced in X. The return value of the function is the X value of the peak of the parabola; it is also stored in global variable ^x_peak. The Y value of the peak is stored in global variable ^y_peak. If the points are in a straight line, ^x_peak and ^y_peak are both set to 0. The syntax is

```
in fpeak
@parpeak(x1,y1,x2,y2,x3,y3)
```

GHOSTDEF.SEQ

Defines a function @GHOST used to analyze ghost reflections in an optical system. The function can also be used as an optimization target in AUTO.

```
in ghostdef

@GHOST(first_refl, second_refl, ghost_surface, zoom_position)
```

where:

first_refl	–	Surface number of the first reflection. Range: 2 to (NUM S). A value of 0 will use (NUM S).
second_refl	–	Surface number of the second reflection. Range: 1 to first_refl-1. A value of 0 will use 1.
ghost_surface	–	Surface number of the surface on which you wish to evaluate the size of the ghost image. Range: second_refl to (NUM S). If ghost_surface=second_refl, then there is effectively only one reflection made. A value of 0 will use (NUM S).
zoom_position	–	Zoom position to use. Range: 1 to (NUM Z). A value of 0 will use 1.

The return value of @GHOST is the paraxial marginal ray height at the specified ghost image surface. A global variable ^GHOST is also filled with the return value. If there are any errors in the parameters, the return value is set to 0 and a global variable ^GERROR is set to 1 (otherwise ^GERROR is set to 0). Note that if the surfaces between first_refl and second_refl or between second_refl and ghost_surface are in NSS ranges, an error is generated.

```
AUTO
@ghost == @ghost(3,1,1,1)
@ghost > 1.0
```

HOEPHASE.SEQ

Macro function @HOEPHASE to compute the phase of a HOE (in waves at the HOE construction wavelength) at a point X,Y for a given zoom position. The HOE phase is also stored in global variable ^HOEPHASE. If an error occurs, global variable ERROR is set to 1. The syntax is

```
in HOEPHASE
@HOEPHASE(surface, zoom_position, X,Y)
```

INDBOUND.SEQ

Defines a function @INDBOUND(mode, surface, zoom) to calculate the index at the extreme left or right side of a lens using LightPath Technologies glass. If mode is negative, the left side is examined; if positive the right side is examined. This is intended to be used in generating UDCs in AUTO. An example showing the syntax is

```
in indbound

AUTO
@indleft  == @indbound(0,^surf,^zpos)
@indright == @indbound(1,^surf,^zpos)
@indleft  >1.6  <1.8
@indright >1.6  <1.8
```

ISODEFS.SEQ

Defines a function @STRNO0 which takes a number in string format and returns the length of the string, not including trailing zeros. This function is called by ISOPLOT. The syntax is

```
in isodefs
@strno0('string')
```

NTSF.SEQ

Defines a macro function @NTSF(value,n,d) to perform a formatted number-to-string conversion. The resulting string is in global string variable ^NTSF, and is a string of n characters with d digits after the decimal. An example showing the syntax is

```
in ntsf
^x == @ntsf(123.45,8,3)
^y == ^ntsf                    ( = string ' 123.450')
```

READLINE.SEQ

Defines a macro function @READLINE('string') to take an input string (such as from a READ command) and extract space delimited entries, placing them in global arrays. the syntax is

```
in readline num_items 'read_prompt'
@readline('input_string')
```

READTEST.SEQ

Macro to demonstrate usage of READLINE.

```
in readtest
```

SPLINE.SEQ

Defines a user-defined macro function @SPLINE to perform a generalized cubic spline interpolation. Two input arrays are needed to define the x and y values of the input data; these arrays must be dimensioned at 100. Two other arrays are used for the interpolated x and y values; these arrays must be dimensioned at 1000.

If only one interpolated point is requested, the routine finds the y value corresponding to the specified x point. If more than one interpolated point is requested, the routine equally spaces the interpolated points across the span of x input points. The syntax is

```
in spline

@SPLINE(num_input_pts, x_input, y_input, num_output_pts,
        x_output, y_output)

usage example:
    num ^x1(100) ^y1(100) ^x2(1000) ^y2(1000)
    ^x1(1) == 1; ^x1(2) == 2; ^x1(3) == 3
    ^y1(1) == 1; ^y1(2) == 4; ^y1(3) == 1
    ^x2(1) == 1.5
    ^ok == @spline(3,^x1,^y1,1,^x2,^y2)
    wri "Interpolated value at" ^x2(1) " is" ^y2(1)
```

Macros to List Various Lens Data

LIST.SEQ

Prints a list of the value of a given numeric parameter for each surface for a specified surface range. Default for end surface is image surface, default for start surface is 1.

```
in list "parameter" [start_surface [end_surface]]
```

LISTDEC.SEQ

Prints a list of tilt/decenter data in the lens for a specified surface range. Default for end surface is image, default for start surface is object surface.

```
in listdec [start_surface [end_surface]]
```

SPCFLD.SEQ

Macro to print the field and vignetting data in a more readable format for more than five fields. Default is for all zoom positions.

```
in spcfld [zoom]
```

SUR.SEQ

Lists the basic surface data (radius, thickness, glass only) along with codes for surface type, solves, zooms, and decenters for a surface range. Default for start surface is object and default for end surface is image.

```
in sur [start_surface [end_surface]]
```

SURC.SEQ

Lists the basic surface data (curvature, thickness, glass only) along with codes for surface type, solves, zooms, and decenter for a surface range. Default for start surface is object and default for end surface is image.

```
in surc [start_surface [end_surface]]
```

ZLI.SEQ

Macro to list the zoom data (similar to a ZLI command) for a surface range, but sorted by surface. Default is for all surfaces.

```
in zli [start_surface [end_surface]]
```

Macros to Change Lens Data

APSET.SEQ

Inputs the same circular or rectangular aperture for a range of surfaces. Default for start surface is 1, default for last surface is next-to-last surface (I-1). If rey is 0 or omitted, a circular aperture is entered.

```
in apset cir|rex [rey [start_surf [end_surf]]]
Def:     req'd     0        1       image-1
```

BAUSCHPRV.SEQ

Loads the Bausch & Lomb glass catalog into a lens private catalog.

```
in bauschprv
```

BENDBACK.SEQ

Bends a lens. Changes the front radius for a given new back radius to hold focal length. Surface j is the radius you are changing.

```
in bendback surf_j new_radius
```

BENDFR.SEQ

Bends a lens. Changes the back radius for a given new front radius to hold focal length. Surface j is the radius you are changing.

```
in bendfr surf_j new_radius
```

BICONVEX.SEQ

Computes the radii required to make an equi-convex lens of a given focal length. Surface j is the first surface of the lens.

```
in biconvex surf_j focal_length
```

BUMP.SEQ

Bumps any lens parameter by a given increment. Do not use quotes on parameter.

```
in bump parameter surf increment
```

CRADJ.SEQ

Macro to adjust the chief ray for a specified field (default all fields) and a specified zoom position (default all zoomed positions) to center the chief ray in the vignetted pupil

```
in cradj [field [zoom]]
Def:        0     0
```

EDGVIG.SEQ

Sets EPD and vignetting factors to eliminate negative edge thicknesses. Default for edge thickness target is 1 mm in lens units, default for air gap target is 0.0001.

```
in edgvig [edge_thi_target [air_gap_target]]
Def:       0.04| 0.1| 1.0           0.0001
```

NODPADJ.SEQ

Computes the nodal points for any two surfaces j and k, and adjusts the thicknesses of surfaces j-1 and k to account for the nodal distances.

```
in nodpadj j k
```

REZOOM.SEQ

Macro to delete specific zoom positions and keep the rest. Selection is done with the POS command; positions set to ON are kept, OFF are discarded.

```
in rezoom
in schott_2000
```

SETXYOB.SEQ, SETXYIM.SEQ, SETXYAN.SEQ

Adjust, respectively, XOB and YOB, XIM and YIM, or XAN and YAN of one specified field, such that the reference wavelength chief ray passes within 10^{-7} lens units (the threshold hard-coded in the macro) through the user-specified coordinates on the image surface. This behavior is similar to XRI and YRI, except that the field will not be updated if the lens is altered.

```
in SETXYOB field_num x_target y_target [zoom_pos]

in SETXYIM field_num x_target y_target [zoom_pos]

in SETXYAN field_num x_target y_target [zoom_pos]
```

UDSFIT.SEQ

Converts a surface to a sixth-order Zernike polynomial form to fit user-specified sag data stored in a file. The data in the file are x,y,z data points, one per line, separated by blanks. Note that this macro sets the surface curvature to zero, and thus first-order data will be incorrect; use real rays to compute first-order properties.

```
in udsfit surface filename
```

Animation Macros

These macros are mainly demonstration macros to show the power and speed of graphics on engineering workstations. They are called animation macros, because they are used to record multiple plot files with small variations (such as a zooming group) and then are played back in succession.

To use these, first run LOOPINIT to initialize the arrays and global variables needed. Then run LOOPHELP for more detailed instructions on the use of these macros.

LOOPDEM1.SEQ

Demonstration macro to show the use of the LOOPVIEW macro to record 3-D plots for animated playback on workstations.

```
in loopdem1
```

LOOPDEM2.SEQ

Demonstration macro to show use of LOOPZOOM (animation of standard lens MOVIE.LEN zooming).

```
in loopdem2
```

LOOPDEM3.SEQ

Demonstration macro to show use of LOOPOPT macro to record a series of plot files as a specified parameter changes (in this case, RIM plots as the front radius varies from 20 to 40 mm).

```
in loopdem3
```

LOOPHELP.SEQ

Help macro for animation macros. It prints out the file stored in LOOPTEXT.SEQ without comments.

```
in loophelp
```

LOOPINIT.SEQ

Initialization routine for animation macros. This macro should be run before running any other of the LOOP* macros. Default for max_frames is 200; the other inputs are required.

```
in loopinit num_loops plot_window win_scale delay [max_frames]
Def:          req'd      req'd      req'd   req'd     200
```

LOOPLAY.SEQ

Macro to play animation loops of stored plot files (reads names from an array and uses DPL). Short command is $PLAY.

```
in looplay ['filename' ['jump'|'rev' [delay]]]
$play ['filename' ['jump'|'rev' [delay]]]
Def:       ''          'rev'      -1
```

LOOPOPT.SEQ

Macro to record option plots for changing parameters in plot files (saving the names in an array). Default commands are for a spot diagram. Short command is $OREC (option record).

```
in loopopt 'filename' ['cmdfile' ['paramstr' [start [end [step
   [Yes| No]]]]]]
$orec 'filename' ['cmdfile' ['paramstr' [start [end [step
   [Yes| No]]]]]]
Def:   req'd        ''         ''         0     0    0  No
```

LOOPTEXT.SEQ

Notes and instructions on LOOP* macros.

```
in looptext
```

LOOPVCOM.SEQ

Sample macro called by LOOPVIEW.

```
in LOOPVCOM azimuth elevation scale
```

LOOPVIEW.SEQ

Macro to record animation loops of VIE plots. Short command is $VREC (view record).

```
in loopview 'filename' ['cmdfile' [Yes| No]]
$vrec 'filename' ['cmdfile' [Yes| No]]
Def:    req'd        ''        No
```

LOOPZOOM.SEQ

Demonstration macro to save zooming VIE plots for animation. Short command is $ZREC (zoom record).

```
in loopzoom 'filename' ['cmdfile' [viewscale [Yes| No]]]
$zrec 'filename' ['cmdfile' [viewscale [Yes| No]]]
Def:    req'd        ''        2.0        No
```

LOOPZRIM.SEQ

Rim commands for test of LOOPZOOM.

```
in loopzrim 'title'
```

Illumination Analysis Macros

These macros are for aiding in the analysis of illumination systems. The theory for these are described in detail in the ORA document *CODE V Application Notes for Illumination Systems*.

ILFOOGET.SEQ

Performs a footprint analysis (FOO) on the specified surface, extracts the output from the Worksheet Buffer to get the area values for each field and store them in the global array ^fooarea(25). 'Print_flag' is either 'Yes' or 'No' (default is 'Yes').

```
in ilfooget surface ['print_flag']
Def:         req'd     'Yes'
```

ILGRDBUF.SEQ

This macro calls TRAGET to get illuminance values for a rectangular grid of object or image points, saving the values in a specified buffer for display by UGR. 'Type' is either 'OBJ' or 'IMG', xgrid and ygrid are sizes of the grid, luminance is the source luminance, and buffer is the buffer number to store the data in.

```
in ilgrdbuf ['type' [xgrid [ygrid [luminance [buffer]]]]]
Def:          'IMG'    11     11       1.0      1
```

ILINTFIL.SEQ

For a specified grid of object points, creates a grid type pseudo-interferogram file (.INT file) of illuminance values calculated by ILTRAGET. The resulting file can be displayed with FAB or UGR. The illuminance values can be multiplied by a specified multiplier and can have a bias added to them. 'Filename' is the name of the .INT file to be created, and 'title' is an optional 80 character title for the .INT file.

```
in ilintfil 'filename.int' ['title' [bias [multiplier]]]
Def:             req'd         ''     0        100
```

ILREFSPH.SEQ

Inserts a dummy or glass reference sphere with a specified radius in object space for a specified object point. A glass sphere is used for display purposes (VIE) as a curved dummy will not draw. The object distance must be finite.

```
in ilrefsph field radius [glass|'air']
Def:        req'd  req'd     'AIR'
```

ILREVERS.SEQ

General macro to reverse a lens from object to image. The macro sets the specification data and NAO as appropriate. The macro will not work for systems with lens modules, HOEs, or non-sequential surfaces, and may not work properly on systems with no XY symmetry, such as with toroids, etc.

```
in ilrevers
```

ILTRAGET.SEQ

Performs a transmission analysis (TRA) and uses the Worksheet Buffer to extract the projected solid angle and transmittance data and store them in global arrays ^psaobj(25), ^psaimg(25), and ^avgtrans(25). 'Print_flag' is 'Yes' or 'No' and NRD is the number of rays across the diameter.

```
in iltraget ['print_flag' [nrd]]
Def:            'Yes'       20
```

Miscellaneous Macros

BEAMPLOT.SEQ

Plots Gaussian Beam Radius vs. Propagation Distance. Run BEAMREAD before using this macro. Uses the UGR option to make plot (does not use generalized plotting macros).

```
in beamplot
```

BEAMREAD.SEQ

Prepares the data for the BEAMPLOT macro. DEFMAXB must be run first. Input wry is the waist radius. FINDKEY.SEQ, SKIPLINE.SEQ, @maxB

```
in beamread wry
```

BENDABER.SEQ

Plots third-order spherical aberrations, coma, and astigmatism vs. bending for a given f/number, index, wavelength, and image height. The bending start value, end value, and increment is input also. All inputs are optional; defaults: f/no = 2, bending starting value = -2, bending ending value = 4, bending increment = 0.2, index = 2.0, wavelength = 500 mm, image height = 5.0.

```
in bendaber fno bend_start bend_end bend_incr index wl image_ht
Def:         2      -2.0      4.0      0.2     2.0   500    5.0
```

CVNEWLENS.SEQ

Default new lens sequence file that is executed by the LEN NEW command, or, from the GUI, by choosing the **File** > **New** menu or by selecting a blank lens in the **New Lens Wizard**. CODE v searches for this macro first in your current working directory, and then in the standard macro directory (CV_MACRO).

```
in cvnewlens
```

FINDKEY.SEQ

Macro to locate a line in a file starting with a given qualifier. This is called by BEAMREAD. This only works if the string to find is in upper case.

```
in findkey "keystring"
```

FOOAPE.SEQ

Performs a FOOTPRINT on a given surface with a given SSI (default is half the semi-diameter) and also draws the combined outer aperture (the aperture formed by the ANDing of all user-defined apertures on that surface). If no user-defined apertures are defined, draws the default aperture. If NRD is specified, forces BOU NO mode.

```
in fooape surface [ssi [nrd]]
```

INPUT.SEQ

Macro to be called from another macro. It puts the calling macro "on hold" and allows you to enter any CODE V commands. Entry of a blank line returns to execution of the calling macro.

```
in input
```

MUL2COA.SEQ

Macro to export multilayer coating data from the MUL option to a *LightTools* User Coating file. To export a coating, load a coating in the MUL option, and execute the macro. The macro is interactive, and will prompt the user for the following information:

Minimum and maximum wavelengths in nanometers, and the number of wavelengths to be output to the file

Minimum and maximum angles of incidence in degrees, and the number of angles of incidence to be output to the file

File name for the coating data

Once this information is entered, CODE V writes the coating data to the screen and the specified file. The file will consist of reflectance values for the s and p polarization components, and the reflected and

transmitted phase differences between the s and p polarization components for the wavelengths and angle of incidence values specified. Note the following limitations:

Only non-absorbing coatings can be exported (i.e., coatings with extinction coefficients cannot be exported). CODE V does not display the s and p transmittance values, and thus the only way to determine them is to use a non-absorbing coating with ts = 1 - rs, and tp = 1 - rp.

A maximum of 100 wavelengths may be output.

A maximum of 5 angles of incidence may be output.

in mul2coa

MULTIPLT.SEQ

Macro to combine multiple plots into one plot. It works by manipulating .EPS files, thus is only useful for laser entries. Each plot can be scaled, translated, and rotated. The macro prompts for inputs; if an input file is specified, the inputs are taken from the file. Refer to the macro notes for a description of the file format.

```
in multiplt ['input_file']
```

RMIN.SEQ

Computes the minimum sized centered circular aperture required on a given surface to pass all ray bundles from all fields and zooms, and defines a circular aperture of that size on the surface.

```
in rmin surface
```

RTPOLPLOT.SEQ

Generates polarization phase effects plots similar to the **PMA; PPH;YP1** and **PMA; PPH; XP1** commands (see “PMA - Pupil Map” on page 18-127 for details about these commands). The difference is that instead of using x- and y-linear analyzers in the exit pupil, radial and azimuthal analyzers are used. Plots of the phase and irradiance in the exit pupil after passing through the radial and azimuthal analyzers are provided by the macro. The output at the center of the pupil is ill-defined. This macro calls two supporting macros (also supplied with CODE V in the CV_MACRO directory):

RadTanPol_def.seq

RadTanPol_PMA_Plot.seq

The syntax is:

```
in RTPolPlot nrd plot
```

where

nrd	–	Number of rays across the pupil diameter (default = 64).
plot	–	Number indicating the type of plot desired: 1 = false color plot 2 = isometric contour 3 = both plots displayed

SKIPLINE.SEQ

Skips n lines in a file read sequence. Called by BEAMREAD. Default for n is 1.

```
in skipline [n]
```

Microlithography Macros

PAR_BOSSUNG.SEQ

PAR_BOSSUNG is a new macro that plots the CDs and distortions of an imaged feature in PAR through dose and defocus. The plot is a series of 1D plots of CD vs. defocus for the sampled values of the dose computed via the BOS option. Nominal CD is taken to be the specified bar width.

The syntax is:

```
IN PAR_BOSSUNG [NBars Width Period DiffLength ExpRange NExp TGR NFoc
FocRange]
```

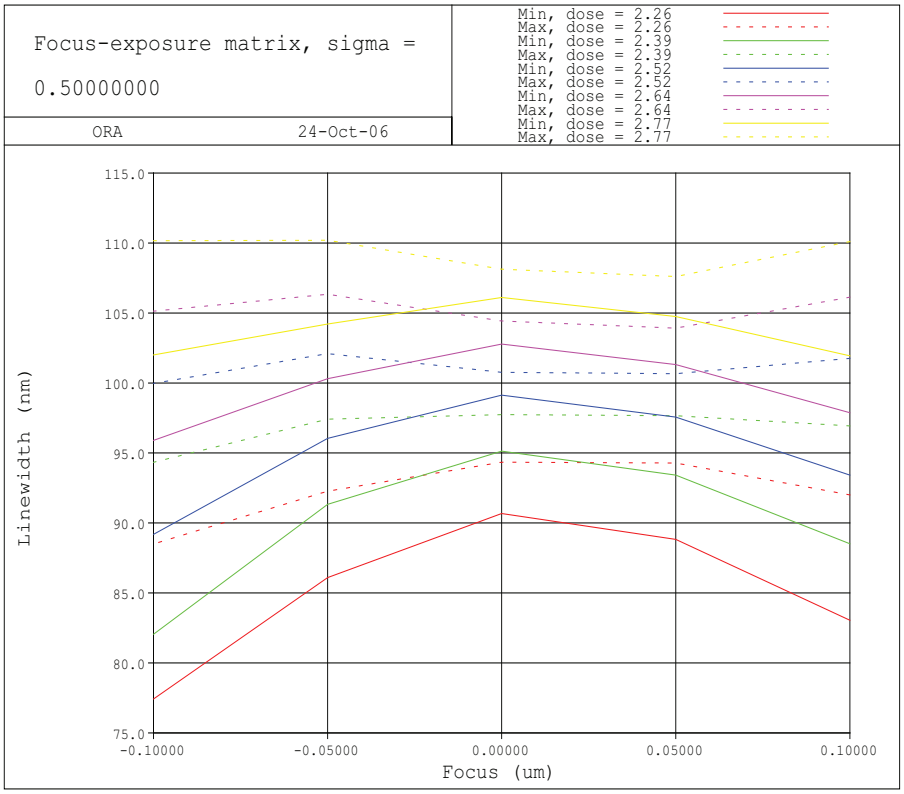
where:

NBars	–	Number of bars in test pattern. Default: 5
Width	–	Width of bars in test pattern. Default: 0.0001 lens units.
Period	–	Number of spokes. Default: 0.0002 lens units.
DiffLength	–	Diffusion length, or width of Gaussian resist blur profile. Default: 0.00002 lens units
ExpRange	–	Range of exposures, expressed as +/- % of nominal. Default: 10.
NExp	–	Number of exposure steps to be plotted. Default: 3.
TGR	–	Transform Grid Region, or number of grid points in PAR computation. Default: 256.
NFoc	–	Number of defocus steps to be plotted. Default: 3.
FocRange	–	The maximum defocus sampled in the +/- directions. Default: 0.0001 lens units.

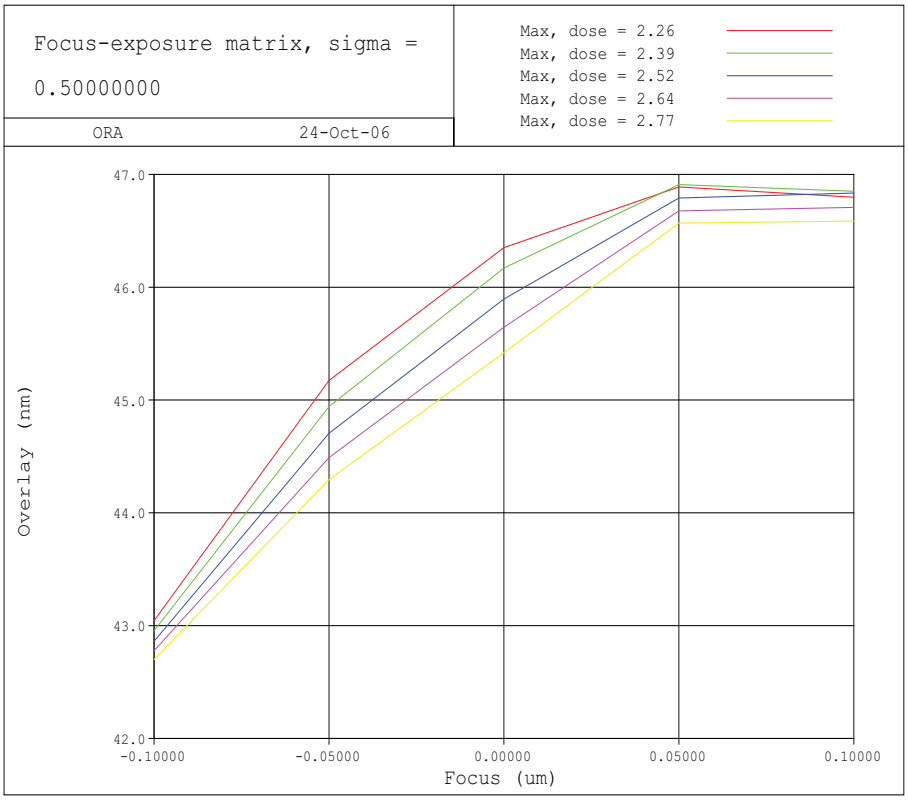
Example:

```
IN PAR_BOSSUNG  5  0.0001  0.00028  0.00002  10  5  256  5  0.0001
```

The output CD is as follows:



The distortion from Par-Bossung is as follows:



PAR_RESIST_PROFILE

PAR_RESIST_PROFILE is a new macro that plots the predicted resist profile from the LPM option against the ideal, geometrical profile. The image is that at dose-to-size (as predicted by DOS) for a nominal width equal to the specified bar width, and the CDs are found at several heights within the resist.

The syntax is:

```
IN PAR_RESIST_PROFILE [NBars Width Period RNA DiffLength Contrast Absorption
ResistThick MinDevRate]
```

Menu
Contents
Indexes
Search

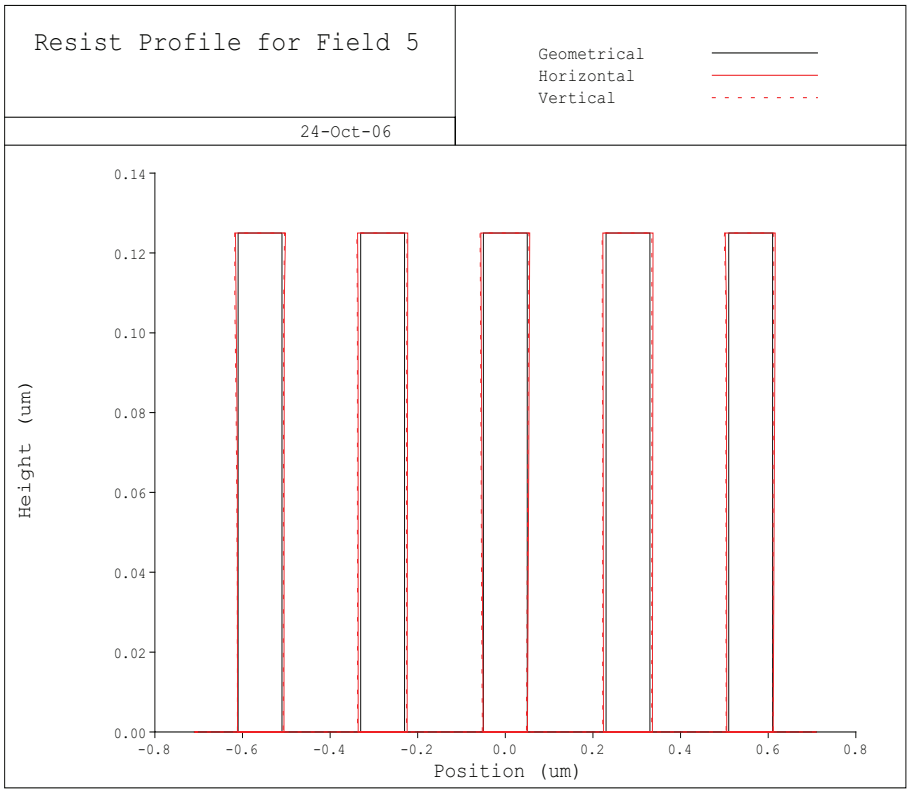
where:

NBars	–	Number of bars in test pattern. Default: 1.
Width	–	Width of bars in test pattern. Default: 0.1 µm.
Period	–	Number of spokes.Default: 0.25 µm; if 0.0, then 2*Width.
RNA	–	Relative Numerical Aperture, or pupil fill. Default: 0.8.
DiffLength	–	Diffusion length, or width of Gaussian resist blur profile. Default: 20 nm.
Contrast	–	The contrast of the resist. Default: 10.5.
Absorption	–	The vertical absorption of the resist per micron. Default: 0.9/ µm.
ResistThick	–	The resist thickness, or the depth within the resist at which the bottom of the feature is defined. Default: 0.1 µm.
MinDevRate	–	The develop rate in nm/sec in the presence of zero light energy. Default: 0.02 nm/sec.

Example:

```
IN PAR_RESIST_PROFILE 5  0.1  0.28  0.75  20  8.0  0.3  0.125  0.02
```

The resist profile from Par_Resist_Profile is as follows:



References

Swanson, Gary J., "Binary Optics Technology: The Theory and Design of Multilevel Diffractive Elements," MIT Lincoln Laboratory Technical Report 854 (August 1989). *NTIS Publ.* AD-A213-404.

Swanson, Gary J., "Binary Optics Technology: Theoretical Limits on the Diffraction Efficiency of Multilevel Diffractive Optical Elements," MIT Lincoln Laboratory Technical Report 914 (March 1991). *NTIS Publ.* AD-A235-404.