MACOS

MACOS Manual

Modeling and Analysis for Controlled Optical Systems

Contact

For information on MACOS contact:
David Redding
dcr@huey.jpl.nasa.gov
Jet Propulsion Laboratory 306-438
4800 Oak GroveDrive
Pasadena CA 91109

Acknowledgment

The MACOS program was developed at the Jet Propulsion Laboratory, California Institute of Technology.

U.S. Government sponsorship of parts of MACOS under NASA Contract NAS7-918 is acknowledged.

MACOS graphical functions are supplied by PGPLOT, an astronomical plotting package. PGPLOT is copyrighted by the California Institute of Technology. PGPLOT is available free of charge from CalTech by anonymous ftp to deimos.caltech.edu, or contact:
Tim Pearson
tjp@deimos.caltech.edu
Astronomy Department, 105-24
California Institute of Technology
Pasadena, CA 91125.

Installation

A separate installation manual is provided with the distribution tape.

Table of Contents

SECTION 1 Introduction

This section introduces the features of MACOS and gives an overview of the manual. New features introduced with this version of the code are summarized.

1.1 Modeling and Analysis for Controlled Optical Systems

The Modeling and Analysis for Controlled Optical Systems software provides tools for modeling and analyzing optical systems. It consists of a stand-alone application program called MACOS and a Fortran subroutine package called SMACOS. SMACOS provides the full functionality of the MACOS application in a form that can be directly integrated with other codes.

MACOS is an optical analysis and model generation program, as opposed to an optical design program. It is focused on generating and exercising fast, accurate, and detailed models combining ray-trace, differential ray-trace, and diffraction techniques. It does not include extensive internal optimization functions or other features to support the optical designer.

MACOS does provide unique modeling capability for *system-level* design and analysis tasks, however. Capabilities include modeling optics on dynamic structures, deformable optics, and controlled optics. These models are easily integrated with standard structures, controls, and user written analysis tools to create an end-to-end system model. Using MACOS linear optical models (or SMACOS subroutines), integrated models can be exercised in end-to-end dynamic simulations, Monte Carlo analyses, covariance analyses, or optimization studies to determine system-level performance.

MACOS (and SMACOS) provide computationally efficient general ray-trace capabilities. These support standard analyses of optical performance such as ray-traces, spot-diagrams, and wavefront (OPD) plots. They also support MACOS diffraction calculations.

MACOS has a differential ray-trace capability which generates linear matrix models of optical systems. These models are accurate for systems that undergo only small perturbations and are essential for tasks such as covariance analysis and control design. Models can be exported to linear analysis programs such as Matlab where they can be exercised alone or in combination with structural, thermal, and control models.

For diffraction calculations, MACOS uses Fresnel propagators driven by exact phases computed by its ray-trace engine. Simulated images (point-spread functions) can be generated using single- or multi-plane diffraction. MACOS supports simulation of multiple source and wavelength images. Digitization (pixilation) can be set to model detector resolution limits.

1.2 Overview of this Manual

This manual provides two services to the user:

- A technical orientation outlining MACOS ray-trace, differential ray-trace, and diffraction modeling capabilities.
- Detailed description of MACOS commands including reference material and numerous examples.

Section 2 of this manual briefly summarizes the technical areas covered by MACOS: geometric optics and physical optics. The purpose is to establish basic definitions and to introduce certain MACOS features. More detailed technical discussion is provided at the beginning of each subsequent section and in the references. Section 3 discusses the basic procedures of starting MACOS, getting help, entering commands, and using files. Sections 4 through 9 proceed in the sequence that a typical MACOS session might:

- Entering the optical prescription from a drawing of the system (Section 4)
- Ray-tracing the system (Section 5)
- Setting up the model for diffraction analysis (Section 6)
- Simulating images (Section 7)
- Building and exporting linear models (Section 8)
- Generating and using SMACOS models (Section 9)

Several example problems are worked in the manual to illustrate specific features of the code. Appendix A contains the prescriptions and journal files for the examples. Appendix B contains selected technical papers.

1.3 New Features of Version 3.2

MACOS 3.2 Beta Version (3.2b) is a **Fortran 90 version** of MACOS, developed using MACOS 2.86 as the baseline. From a user's point of view, the most important improvement in MACOS 3.2b is its ability to support runtime specification of the MACOS model size in both interactive mode and in subroutine MACOS (SMACOS). The usage of this feature is described in Section 3.3 for the interactive mode and in Section 9.3 for SMACOS. Taking advantage of Fortran 90 compiler's strong type-checking, a number of type mismatches between calling and called subroutines in MACOS have been fixed. MACOS 3.2b has been tested on several optical system simulations for space telescope wavefront sensing and control.

MACOS 3.2b retains all improvements that MACOS 2.8 made over earlier versions of MACOS. For users familiar with MACOS before version 2.8, version 2.8 brings some improvements and some changes. New features include additional surface types, additional element types, a new segmentation option, and glass tables. Full backward compatibility with existing MACOS prescriptions and journal files has been preserved. MACOS 2.8 made some changes in its interface to SMACOS functions. These include a changed SMACOS argument list; also some important arguments and internal arrays, such as `OPMmat`, are now `REAL*8` rather than `REAL*4` variables. In additionto these changes, a user program calling SMACOS in MACOS 3.2b needs to include relevant Fortran 90 modules instead of Fortran 77 common block files as a way of communicating with SMACOS. Programs calling SMACOS should be examined to ensure compatibility with Version 3.2.

A summary of changes since MACOS 2.8:

- Toroid surface type was added

- Interpolated data surfaces using general X-Y-Z data are once again supported
- Interpolated data surfaces using square X-Y grids have been added
- Influence function-based surfaces are provided as a “user-defined” surface
- Transmissive and reflective straight-ruled gratings were added
- “Flower” segmentation was added
- A gaussian blur function was added for simulating image jitter and other noise effects
- SMACOS call line arguments were changed
- `OPMmat` and `RaySpot` variables were changed to double precision

SECTION 2 Technical Overview

This section provides a technical orientation to Modeling and Analysis for Controlled Optical Systems, providing brief reviews of geometric and physical optics. Though in the form of a tutorial, it is intended mainly as a summary of definitions and approach.

2.1 Geometry and Coordinate Systems

Optical systems are composed of *elements* — mirrors, lenses, detectors, gratings, holographic optical elements, and others — that have optical properties. These optical elements are distributed in 3-space, which is to say, they are placed at different locations. *Where* they are located is of critical importance. Telescopes, cameras, optical systems in general depend on precise element positioning to achieve good optical performance.

Equally important is element *orientation*—where each element is pointed. Angles of reflection or refraction of a light beam depend on the orientation in 3-space of each surface in the system.

2.1.1 Global Coordinates

MACOS requires the user to specify the locations and orientations of elements using vectors in a single *global coordinate system*, as opposed to the usual approach of specifying each optical element relative to the preceding element. A single user-defined coordinate system is often more convenient when preparing models to be integrated with structural models: the structural coordinates can be used for the optical model as well. It facilitates checking results, especially for decentered and tilted systems. A global coordinate system also eliminates unnecessary calculations from the ray-trace. All MACOS calculations are actually done in the user-specified global coordinate system (Ref. Redding and Breckenridege). Specifying the system in those coordinates eliminates coordinate transformations as well as ray calculations at dummy surfaces. The result is significantly faster execution times.

To illustrate the choice of a global coordinate system, consider the simple 2-mirror Cassegrain telescope drawn in Figure 1.

In this optical system there are three optical elements: the primary mirror (PM); the secondary mirror (SM); and a focal plane (FP). Two possible global coordinate system are

- The structural coordinate frame (*Frame A)*, which would be shared by structural models, simplifying integration of optics and structures models.
- A frame centered at an optical element, such as the primary mirror, and aligned with the optical axis of the system (*Frame B*). This frame is more natural for optical layout. It can also be convenient when comparing results with optical design programs.

The location of each element is defined in MACOS using a 3-vector called `VptElt` (for "vertex point"). For the primary mirror of our example, `VptElt` is drawn from the origin of the global coordinate system to the vertex of the element, which is shown as the small circle. `VptElt` is indexed by 1=x, 2=y, 3=z coordinate values. For planes and

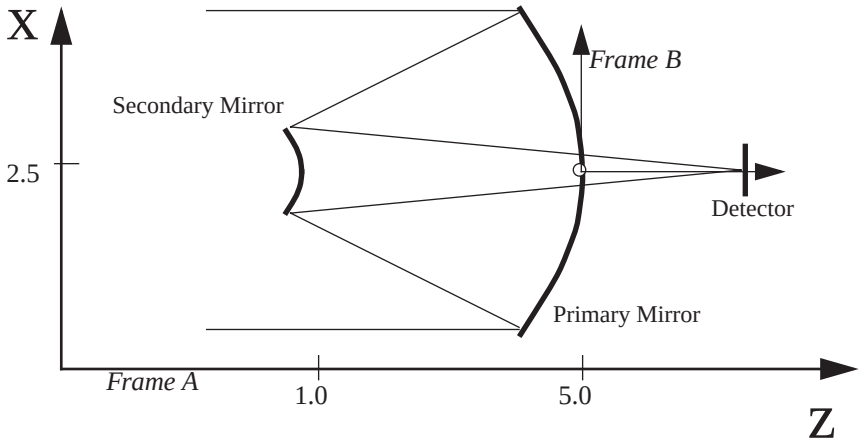


FIGURE 1 Coordinate systems for the Cassegrain telescope example.

spheres, which have no unique vertex, `VptElt` can be any point on the surface. For the two example frames, `VptElt` takes values as listed in Tables 1 and 2

TABLE 1. Primary mirror location specification

Location	Frame A	Frame B
`VptElt(1)`	2.5	0.0
`VptElt(2)`	0.0	0.0
`VptElt(3)`	5.0	0.0

TABLE 2. Secondary mirror location specification

Location	Frame A	Frame B
`VptElt(1)`	2.5	0.0
`VptElt(2)`	0.0	0.0
`VptElt(3)`	1.0	-4.0

Comparing the intervertex spacings of the two mirrors (the difference of the PM and SM `VptElt`), you can see that they are the same in either frame. It does not matter what frame is used, provided the relative location of each element is correct.

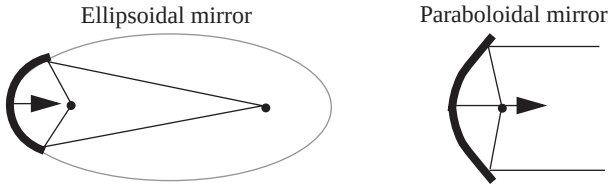


FIGURE 2 Principal axis directions

The orientation of each element is specified by a *unit-vector* (a 3-vector of length 1) called `PsiElt`. For conicoid elements `PsiElt` is the *principal axis direction* of the element surface: the direction of a line drawn from the vertex of the element towards the nearest geometric focus. Figure 2 shows the principal axis for elliptical and paraboloi-

dal mirrors. The principal axes for spheres or flats, which have no unique vertex, are defined from `VptElt`.

For the primary and secondary mirrors in our example, the principal axes are pointed in the negative z direction (Table 3). Since Frame A and Frame B have the same orientation, (they differ only in translation) the values of `PsiElt` are the same in both coordinate frames.

TABLE 3. Mirror orientations

Orientation	Primary Mirror	Secondary Mirror
PsiElt(1)	0.0	0.0
PsiElt(2)	0.0	0.0
PsiElt(3)	-1.0	-1.0

Detailed discussion of how elements are specified is provided in Section 4.

2.1.2 Local coordinates

MACOS allows the user to explicitly specify three types of local coordinates: *element coordinates*, *output coordinates*, and *surface coordinates*. Also important, but not explicitly set at each surface, are *beam coordinates*.

Element coordinates allow the user to simulate optical *actuators*, such as steering mirrors or translation stages, by defining coordinates local to the controlled element and aligned with the actuator axes. Actions of the steering mirror are rotations about, or translations along, that axis, and are easily implemented via the MACOS `PERturb` command (see Section 5.3.7). These *element coordinates* can also be imbedded into a linear model of the system using the MACOS `BUIld` and `EXPort` commands (see Sections 8.2.1 and 3.8.5). This enables creation of optical models directly in terms of actuator actions.

Output coordinates are used when calculating certain graphical outputs, such as spot diagrams. They help define sensing axes for simulating *detectors* (e.g., rows and columns of CCDs). When embedded in a linear model, they define the optical outputs of that model by selecting and transforming the optical states into combinations specified by the user.

Element coordinates are centered at a *rotation point*, specified as a 3-vector `RptElt`, which is the (0,0,0) point of the element coordinate system expressed in global coordinates. All rotations imparted to the element through the `PERturb` function are about `RptElt`.

The element coordinate axes are 6-vectors which describe rotational and translational perturbation to the element. Each 6-vector consists of two 3-vectors designating the rotation and translation, respectively. Usually an element will be actuated in rotation or translation only, but the option exists to use coupled rotation/translation coordinates.

Surface coordinates are required to define propertires of certain surface types, such as Zernike surfaces, and are described in Section 4.

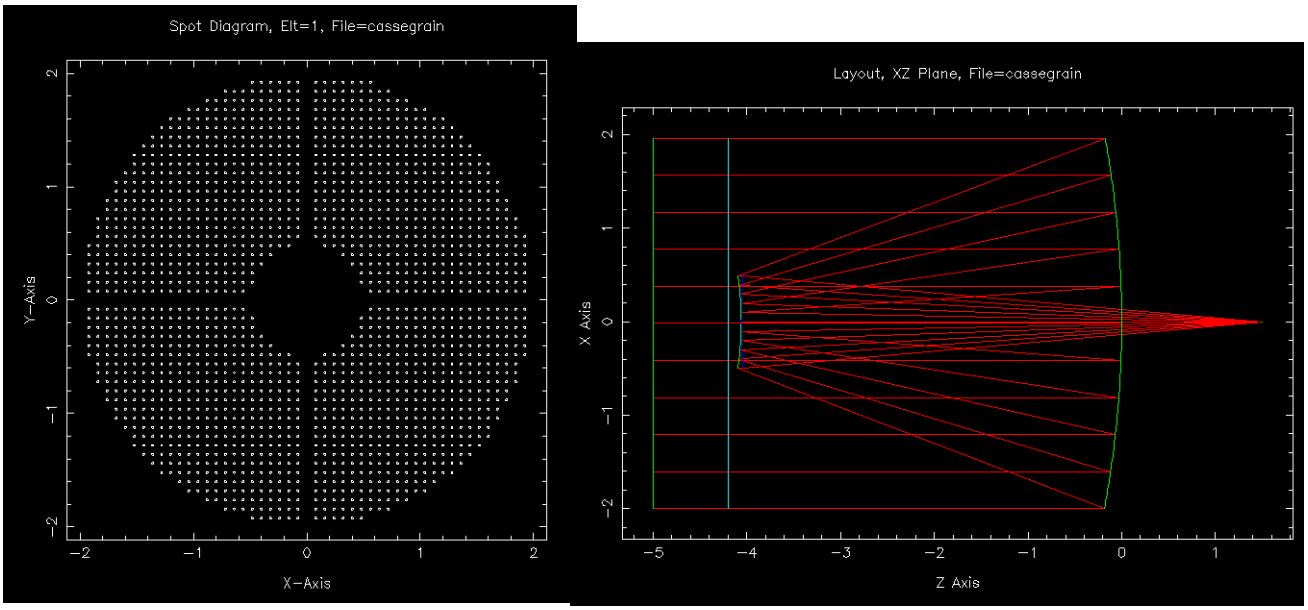
Beam coordinates define the orientation of the ray and diffraction grids at particular elements. They are initially defined at the source, and propagated to each subsequent optic by the ray tracing process. In general they are not the same at each element (due to tilt-

ing, rotating and inverting properties of the optical system). Beam coordinates can be computed at any surface using the MACOS `COORD` command.

2.2 Geometric Optics: Tracing Rays

Geometric optics capture the *particle nature* of light. In geometric optics, light beams are represented as bundles of *rays*, which are the trajectories of individual photons. Rays are generally composed of numerous straight-line segments, with direction changes at the reflective or refractive surfaces of optical elements (curved segments occur in regions of continuously changing refractive index). *Ray-tracing* is the process of propagating a bundle of rays through an entire optical system.

Ray-tracing is an extremely valuable tool for determining the performance of optical systems. For example, characteristics of a well-designed imaging system include an extremely small *spot diagram* at the detector focal plane. Spot diagrams are plots of the pierce points of a ray bundle on a surface, and a small spot indicates a tight focus. An example is provided by the Cassegrain telescope of Figure 3. Two spot diagrams are shown in Figure 4. The unaberrated telescope indeed has a focus at a single spot (diameter is approximately 10^{-10} in arbitrary units). Misaligning the primary mirror causes the spot to spread as shown in Figure 4 to approximately $2x10^{-8}$.



Spot diagram at entrance pupil

Cross-section of ray-trace

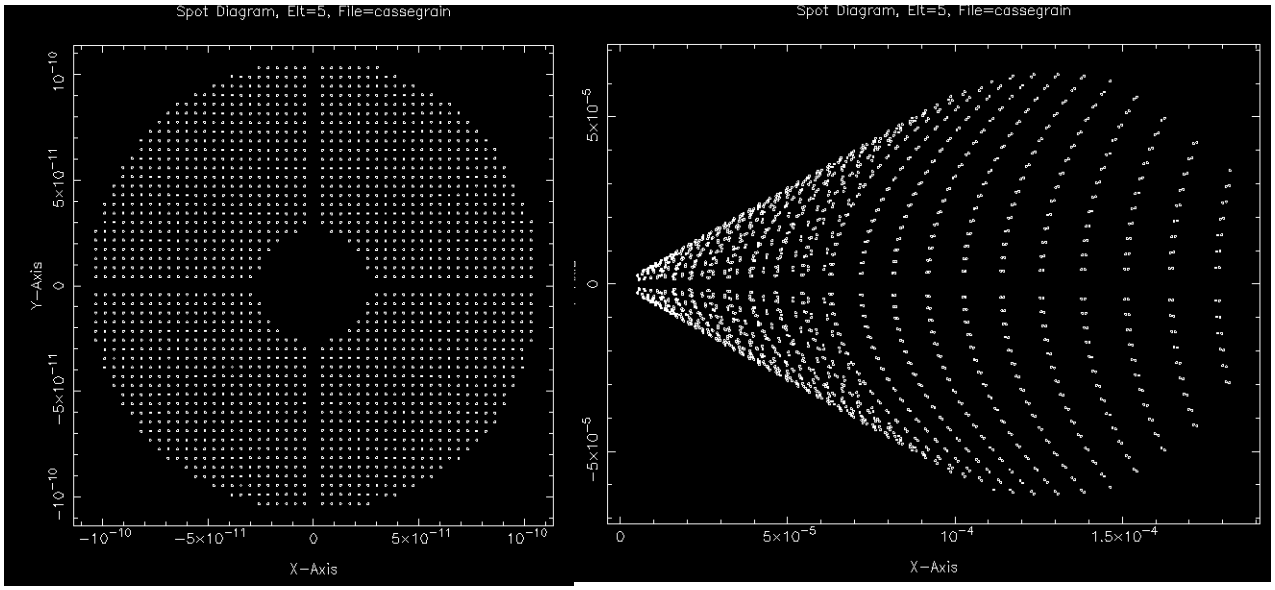
FIGURE 3 Ray trace of a Cassegrain telescope

As another example, interferometer systems can be evaluated by calculating the flatness of two interfering wavefronts at a pupil of a system. The wavefront of an extended beam is accurately represented by an *Optical Path Difference* (OPD) map at a particular surface, such as the beam-splitter where the interference is to take place.

The predictions of geometric optics are exact only in cases where the wavelength of the light is infinitely small. Of course, no such system exists. Geometric optics do provide extremely good predictions in a wide range of cases, where wavelengths are small compared to the dimensions of the optical system. Even in systems with long wavelengths

and large diffraction effects, geometric optics are useful for determining the best design conditions.

One thing that ray-tracing cannot do is predict the intensity pattern of a focused spot.



Focal plane spot diagram
(unaberrated system)

Focal plane spot diagram
(primary mirror tilted 1e-4 radians)

FIGURE 4 Spot diagrams for `Cassegrain.in`.

Near-focus is a region where diffraction effects dominate, as a perfectly-focused spot is itself infinitely small. For this case as well as others, *physical optics* approaches must be employed. Physical optics capture the wave nature of light and can accurately account for diffraction around obstructions, or in tightly focused beams, or at diffraction gratings. The MACOS approach to physical optics is discussed in the Section 6.

The exit pupil is a conjugate (an image) of the aperture stop of the system. For the Cassegrain telescope example, the entrance pupil is located at the primary mirror. A point source at the primary produces a virtual image behind the secondary mirror (see Figure 5). The exit pupil is a spherical “reference” surface at the virtual image point, with center of curvature on the focal plane. This surface coincides with the nominal spherical wavefront converging on the geometric focus.The OPD distribution at the exit pupil determines the image quality. Perfect optical systems have zero OPD; the wavefront leaving the optical system is spherical. In an aberrated optical system the OPD is nonzero as shown in Figure 6.

2.3 Physical Optics: Diffraction

Physical optics capture the *wave nature* of light. In physical optics, Huygen’s Principle states that each point within a beam of light radiates a spherical wave of light. A *wavefront* is the envelope of all the waves emitted at a particular instant of time. Waves of

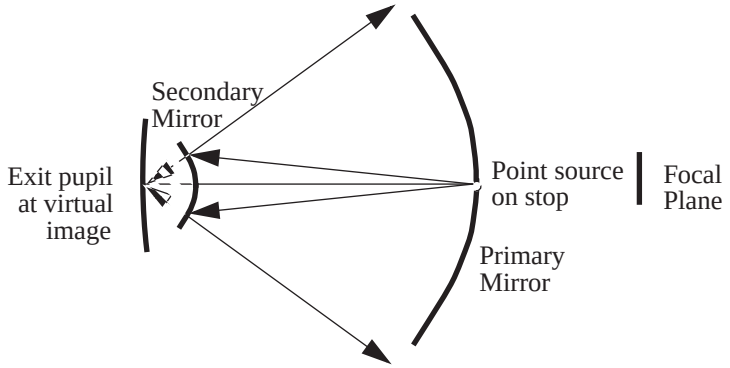
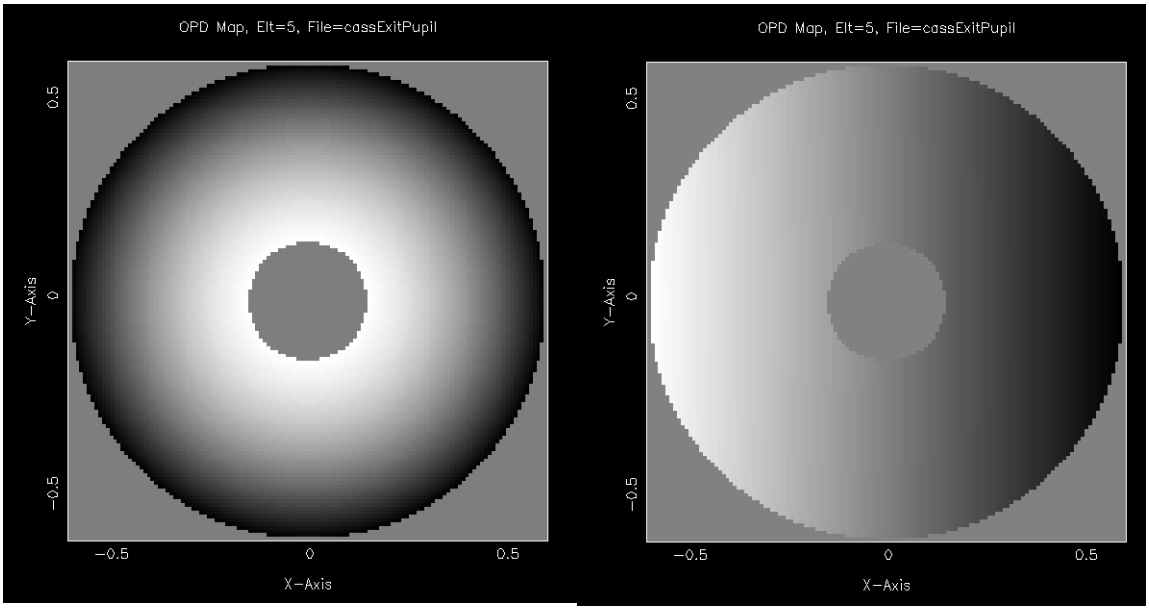


FIGURE 5 Exit pupil for `Cassegrain.in`



Unaberrated (defocused)

Aberrated OPD (coma)

FIGURE 6 OPD at the exit pupil

light propagate through an optical train by bouncing off of mirrors or bending through lenses, much as predicted by geometric optics. Unlike geometric optics, however, physical optics predicts that light can leak back into areas of shadow, or areas outside of the edge of the geometric beam. This process is *diffraction* and is important when modelling images, the propagation of beams over long distances, and other phenomena.

The physics underlying diffraction and the free-space propagation of light are captured in Maxwell's equations. Physical optics theories have simplified these equations, enabling practical solutions to optical problems. *Scalar diffraction theory* assumes time invariance and ignores polarization effects (Ref. Goodman). *Fresnel propagation theory* further assumes that the beam is paraxial, or a small beam divergence/convergence. This allows the expression of the propagation of light from one planar surface to another planar surface in terms of the *paraxial wave equation*.

Geometric optics is the solution of Maxwell's equations in the limit of an infinitesimally small wavelength. This is a good approximation for many practical optics problems.

Diffraction effects start to dominate the propagation of a beam when the beam size approaches its wavelength (i.e., near focus), or when the dimensions of a surface approaches the beam wavelength (i.e., diffraction grating grooves), or when the beam is projected over distances exceeding its Rayleigh length (Ref. Goodman, Siegman).

The paraxial wave equation can be solved using two-dimensional Fourier transformations for free-space propagation between planes. MACOS implements these using computationally efficient fast Fourier transforms (FFTs). Care must be taken when using Fourier optics because of the approximations inherent in the Fresnel assumptions. Using FFTs brings additional concerns of *sampling density, aliasing* and *wrap-around*. Violating the Fresnel conditions, or sampling incorrectly, can affect the accuracy of computed results. MACOS provides several features to help you set problems up correctly.

In a physical optics computer model, the light beam is evaluated on *reference surfaces*. These are "dummy" surfaces located in the beam and aligned to the beam. They have no optical effect and are essentially transparent screens on which the beam is sampled.

The light on the reference surface is represented as a *complex matrix*, each element of which is the complex amplitude of the scalar field at a particular location. These locations are distributed as a regular rectangular grid of points across each reference surface (Fig. 7). Light is *propagated* between these reference surfaces using algorithms based (in the case of Fresnel diffraction) on the solutions to the paraxial wave equation.

The complex amplitudes can be interpreted as having *magnitude* and *phase*. Viewed geometrically, the phases describe the OPD (see Fig. 8). If a reference surface is not aligned with a particular wavefront, the phases will vary "rapidly" across the reference surface grid. On the other hand, if the reference surface is nearly aligned with the wavefront, the phases vary only slowly. The magnitude describes the amplitude of the electromagnetic field across the exit pupil.

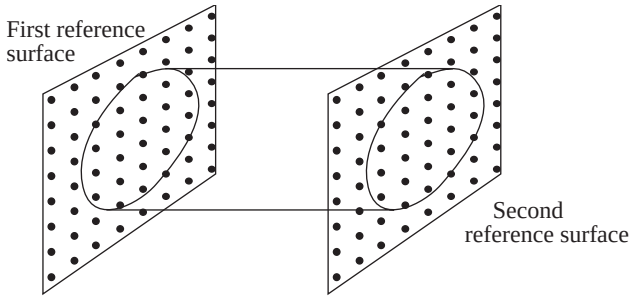
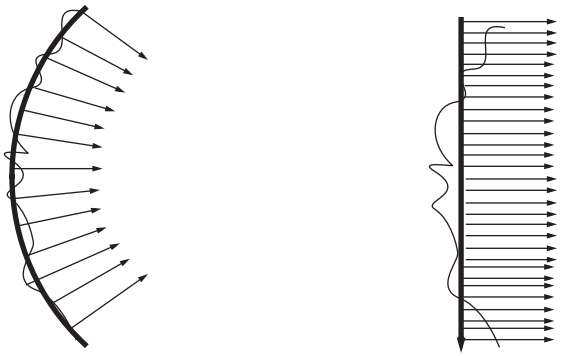


FIGURE 7 Sampling grid for a collimated beam

Sampling refers to the density of samples per phase crossing. The FFT algorithm works quite well when there are many (at least 4 to 8) sampling points per phase crossing. When there are fewer, the ability of the FFT to capture higher spatial frequency effects is diminished. This difficulty can become acute when trying to sample diverging or converging waves using planar reference surfaces. Fortunately, this is not necessary. A coordinate transformation (Ref. Sziklas and Siegman) allows the paraxial wave equation to be applied to converging or diverging beams, by removing the spherical component of phase from the light. This is equivalent to sampling the light on a *spherical reference surface*.



Sampling on a sphere takes out the need to sample the spherical part of the wavefront.

Sampling on a plane must be higher to capture the spherical part of the wavefront.

FIGURE 8 Sampling a converging beam

Using a spherical reference surface in a converging or diverging beam allows the sampling to be determined by the aberrations in the beam, which are the phase distortions, or the deviations from the nominal spherical wave. This vastly reduces the sampling requirement for most optics problems.

Whether planar or spherical reference surfaces are to be used to sample a particular light beam depends on the curvature of the wavefronts that make up that beam, and also on the width of the beam. MACOS has functions that allow the geometric determination of the curvature. These functions fit pairs of reference surfaces into the beam at specified points, setting up near-field propagations.

The width of the beam grid relative to the amplitude grid is set to avoid aliasing. *Aliasing* is an artifact of the FFT algorithm, which transforms amplitudes distributed in space to frequencies distributed in the spatial frequency domain. Because the FFT is a periodic calculation, the frequencies computed for the amplitudes are replicated at higher frequencies. If the beam is too wide compared to the extent of the amplitude matrix, some of the replicated "beams" can leak back into the main beam. This effect is aliasing.

The main symptom of aliasing in a diffraction computation is a "noisy looking" beam intensity pattern. Aliasing causes spikes to appear in the beam. These are usually most pronounced in regions of high intensity. Aliasing can be minimized by *padding* the amplitude matrix so that the gridpoints that coincide with the beam occupy about half of the width, or about a quarter of the area of the grid (Figure 9). The padding has the effect of restricting aliasing to very high-frequency effects which are usually of quite low amplitude. Additional padding can be specified by using smaller values of the nGridPts parameter, which governs the relative size of the ray and diffraction grid. MACOS does not currently provide shaped windowing features.

The width of the beam in regions far from focus is determined geometrically. When the beam is small compared to the wavelength, the width is determined largely by diffraction considerations. Near focus, diffraction effects dominate. MACOS is set up to automatically require adequate padding in diffraction calculations for imaging systems. Nonetheless, the user is urged to be aware of the rules for constructing diffraction models, as discussed in Section 6. Good discussions of these issues are available (Ref. Lawrence).

The shape and quality of the *point-spread function* (PSF) are a good indicator of the performance of the optical system. The PSF is the image or intensity pattern due to a point

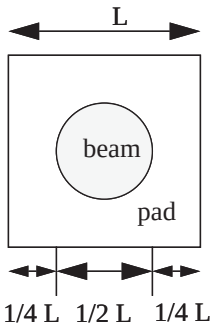


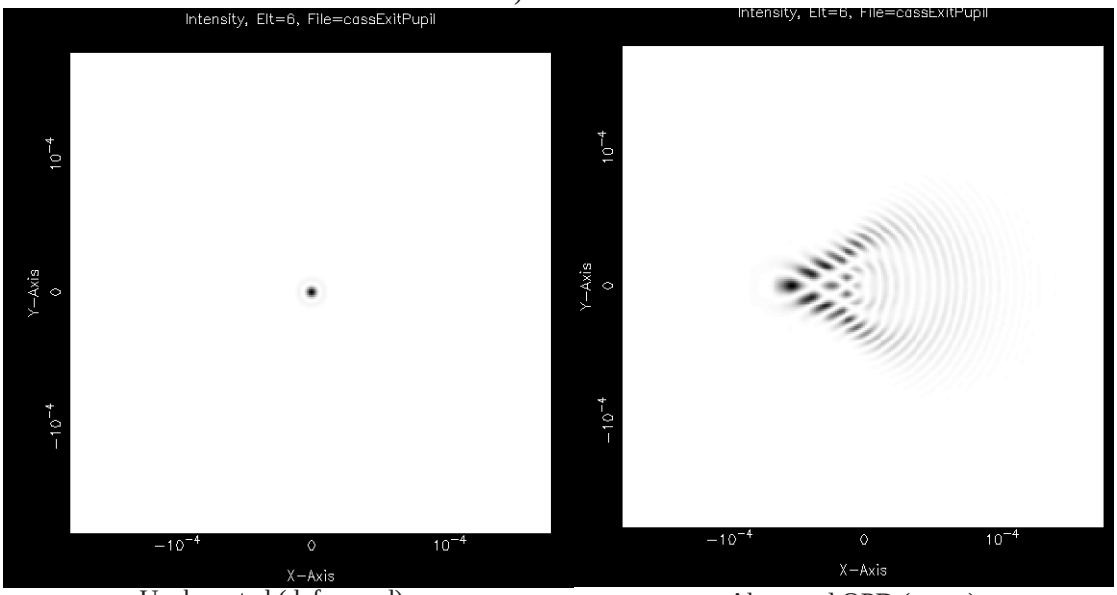
FIGURE 9 Padding the amplitude matrix.

source. Without aberrations, geometrical optics predicts infinitely small images: a delta function of infinite amplitude and zero width. Alas due to diffraction from finite sized apertures of the optical system, the PSF is never a delta function. If the aberrations are small the imaging performance of a system is not degraded, but the performance is *diffraction limited*. As the aberrations in the optical system are increased, the PSF degrades further: its peak value decreases, the side-lobes grow, and the pattern spread out.

2.3.1 Single Plane Diffraction

The simplest and most common PSF calculation uses a ray trace to compute the OPD in the exit pupil of the optical system followed by a single *far-field diffraction propagation* to the focal plane. Figure 10 shows the PSFs computed by MACOS for the Cassegrain telescope calculated in this fashion (see Figure 6 for the OPD maps).

)



Unaberrated (defocused)

Aberrated OPD (coma)

FIGURE 10 Point spread functions for `Cassegrain.in`

2.3.2 Conventional Multi-plane Diffraction

Conventional physical optics modelling of multi-plane diffraction phenomena *unfolds* the optical beam train into an equivalent set of *thin lenses* arranged in linear fashion along the optical axis.[2,4] The effect of the lenses is modeled using the *thin-phase approximation*. This approach assumes that:

- The beam is paraxial, so that phase changes occur as a quadratic function of distance from the center of the array.
- The grid does not change shape or dimension across the lens; the beam out of the lens is the same size as the beam into the lens.
- The lens has no thickness, so all phase changes occur in a plane.

MACOS does not make these approximations. MACOS does not unfold the system nor use the thin-phase approximation.

2.3.3 Multi-plane Diffraction in MACOS

MACOS multi-plane diffraction models are set up by placing reference surfaces immediately before and after the optical elements of a system, or at the start and end of free-space propagations. These reference surfaces are optimized in pairs using MACOS `ORS`, `SRS` and `FEX` functions, discussed in Sections 5 and 6.

MACOS diffraction calculations use four basic propagators:

1. Geometric propagator.
2. Fresnel near-field sphere-to-sphere propagator for converging or diverging beams.
3. Fresnel/Fraunhofer far-field sphere-to-plane propagator.
4. Fresnel near-field plane-to-plane propagator for collimated beams.

The Fresnel propagators are used for the free-space propagations in much the same way as in standard codes. They determine the amplitude on the downstream reference surface based on the amplitude on the upstream reference surface. These propagators are described in more detail in Section 6.

MACOS uses the geometric propagator to determine phase changes across optical elements. The geometric propagator uses rays as *phase probes* aligned with particular amplitude matrix grid points. The phases are determined directly from the exact optical path lengths calculated for the rays from the first reference surface, through the optical elements, to the next reference surface. If necessary due to aberrations through the optical elements, the rays can be regridded to assure alignment with the diffraction grid points. Compared to thin-phase, the MACOS approach

- Does not assume that the beam is paraxial through the optical element (though an assumption of paraxial beams is built into the Fresnel free-space propagators).
- Does not assume that the beam out of the lens is the same size as the beam into the lens.
- Does not assume all phase changes occur in a plane.
- More accurately accounts for geometric effects, such as field-dependant aberrations.

The MACOS geometric propagator does not compute magnitude changes unless polarization ray-trace is being used.

2.4 Polarized Light

MACOS polarized light summary: wht polarized light is; representation of polarized TE fields using rays; polarization rt; interfaces; coatings; vector diffraction.

SECTION 3 User Interface

We begin this section by discussing the mechanics of interacting with the MACOS application. Basic file-oriented functions are discussed, as are various graphics options. Memory requirements are discussed. The commands covered in this section are:

```
Help
EXPort
JOUrnal
EXEcute
```

3.1 Starting and Stopping MACOS

The particular command used to start the MACOS application program depends—of course—on which machine you are using. If your machine is a…

- Macintosh: open the folder containing the MACOS application and double-click on the MACOS 1.0 icon
- Unix: type the command MACOS with appropriate path names to point to the MACOS executable file

On all machines, the response is the same. A window appears with the MACOS command dialog:

```
Modeling and Analysis for Controlled Optical Systems
MACOS Version 2.8, May 31, 1999

This run is limited to  70 surfaces,  69 segments,   4226
rays (    441 for sensitivity calcs). Up to  3 diffraction planes are stored.
Up to 10 interpolated surfaces of up to    255 data points are supported.
Maximum pixel array size is   128 by   128.
Diffraction grid size is   128 by   128.

Glass table /home/comp/macosData/macos.glass read, with 202 glasses included.

For a list of commands, type H

MACOS>
```

The `MACOS>` prompt indicates readiness of the MACOS application to take commands. To exit MACOS, type `Quit` or `BYE` to end a session. The response you get depends on whether a PGPLOT graphics window is open (Unix machines running XWindows only). If so, MACOS will ask you to hit return before it goes away. On Sun machines a warning that “non-standard” floating point results may have been produced will be printed. This can be ignored.

```
MACOS>q
 Type <RETURN> for next page:
```

3.2 Help

A good first command is `Help`:

```
MACOS>h
```

The `Help` command produces a summary of MACOS commands, grouped in categories. MACOS top-level commands, for loading and modifying optical prescription data, creating journal files and executing macro files, and for exporting data and results are:

```
MACOS command summary:
  Commands:   Quit or END MACOS
              NEW: build new optical system dataset
              OLD: load existing optical system dataset
              EXEcute: run a .jou journal file
              JOUrnal: generate a .jou journal file
              SUMmarize optical system data
              STAtus of current calculations
              RESet options and defaults to starting values
              SHOw data for a particular element
              MODify optical system data
              SAVe optical system data
              EXPort specified results in various formats
```

The following commands are for analyzing ray-traced optical systems and to set up diffraction computations. They can be used to trace rays, specify polarization conditions, optimize the source location, perturb optical elements, determine optical path differences, and set other conditions. Most of these commands are described in Section 5:

```
  Commands for ray-trace analysis are:
              MAP prints ray and segment ID maps
              RAYtrace a specified ray
              POLarized light
              NOPOL: no polarization calculations
              OBS: spot diagram obscuration option
              SINT: setup interpolated surface data
              PERturb optical system
              PREad: perturb using input from a file
              OPD plot and analysis
              CENTEr beam in system stop
              STOP defines system stop
              COOrd: compute beam coordinates at an element
              CENTRoid sets centroid as reference pt for FEX
              FFP: find field point to place chief ray at a
              specified point on an element
              PFP: find field point to place chief ray at a
              specified pixel location on a detector
              SPOt diagram plot
```

The following commands are used for generating linear optical models, and are described in Section 8:

```
  Commands for linear model building and analysis are:
               BUild linear model using ray partials
               DMBUild including a deformable mirror
               PARtials: print patrials of a specified ray
               LPErturb optical system using linear model
               LPRead: perturb using input from a file
               LREset: zero previous linear perturbations
               LSPot: perturbed linear system spot diagram
               LOPD: perturbed linear system OPD map
               LPIxilate: plot perturbed pixilated image
               LINtensity: plot perturbed wavefront
  Hit return to continue...
```

The following commands are used to set up diffraction computations, by defining the source beam illumination profile, specifying polarization conditions, setting up appropriate reference surfaces to define start and end-points for the diffraction computations, and setting other conditions. Most of these commands are described in Section 6:

```
  Commands for diffraction analysis are:
              BEAm: set beam type
              VECtor diffraction (with polarized light)
              SCAlar diffraction (with polarized light)
              REGrid at specified element during propagation
              NOREGrid: regridding off at specified element
              ORS to optimize reference surface
              SRS to slave one reference surface to another
              LNEGok allows negative ray lengths at ref surfs
              NOLNeg prohibits negative ray lengths
              COMpose a multiple object/color image
```

The following commands perform diffraction propagation to generate beam amplitude and/or intensity at a specified element. They then produce a visual display (or file output) of the beam TE field, PSF, image or wavefront. These commands are described in Section 7:

```
Commands to plot wavefronts and images are:
              INtensity of the wavefront output
              PIXillated intensity output
              ADD current intensity to composed image
              DADD: display current composed image
              LOg intensity output
              AMplitude and phase output
              REal and imaginary wavefront output
              PFP: find field point to place chief ray at a
                   location specified in pixel coordinates
              PLOcate: specify location of pixel array
              SEEd sets image noise seed value
              NOIse adds noise to a COMposed image
              STRetch specifies display data stretch (LINear, LOG10, or SQRT)
```

When using the Unix version of MACOS, computed results can be displayed in one of several forms: gray-scale, wire-frame, contour plot, or horizontal or vertical data "slices." These different types of display are selected by changing the active "plot type," using one of the following commands. Plot type can also be set to export data as a data-file, in text, binary, fits-format or Matlab format. Where direct graphical output is not available (such as for the Macintosh version of MACOS), output files can be used to pass data to analysis or plotting applications for display. Use of supplied MACOS Matlab scripts to generate graphics from MACOS datasets is discussed later.

```
  Commands to set wavefront plot type are:
              WIRe sets wireframe surface plotting
              SLIce sets "slice" surface plotting
              GRay sets gray-scale surface plotting
              COLumn sets column line-plotting
              ROW sets row line-plotting
              CONtour sets contour surface plotting
              TEXt print output
              BINary image file output
              FITS image file output
              MATlab image file output
  Help repeats this message
MACOS>
```

3.3 MACOS Model Size Specification

The first thing the user will be asked to do after starting MACOS is to specify a model size. The model size determines MACOS computational accuray, such as the number of rays traced and image grid resolution, and obviously also has an impact on the overall computer simulation time. MACOS currently supports the following model sizes: 128, 256, 512, 1024 and 2048 mainly because the fast Fourier transorm routine is optimized for power-of-2 arrays.

The MACOS model size can be changed at any time during a MACOS session. Use the *mreset* command to set a new model size, as shown in the following example to set a model size to 256:

```
MACOS> mreset 256

or

MACOS> mreset
Enter new size (128, 256, 512, 1024, 2048): [128]: 256

Modeling and Analysis for Controlled Optical Systems
MACOS Version 3.2, February, 2003

This run is limited to  70 surfaces,  69 segments,  16385
rays (    500 for sensitivity calcs). Up to  3 diffraction planes are stored.
Up to 10 interpolated surfaces of up to    255 data points are supported.
Maximum pixel array size is   256 by   256.
Diffraction grid size is   256 by   256.
```

In previous versions of MACOS, separate model parameter files are used for different model sizes. For example, the model parameter file *param256.cmn* is used for the model size 256. In MACOS 3.2b, a single model parameter file called *macos_param.txt* is used for all model sizes. *macos_param.txt* contains parameter sets for all supported model sizes. To run MACOS, either *macos_param.txt* or a link to it must exist in the MACOS execution directory.

3.4 Entering Commands

MACOS commands can be abreviated using the first letters of the command. The neccessary letters are denoted in capital letters. However, they do not need to be entered in capitals as MACOS is case insensitive.

MACOS prompts the user for the next input. For example, as shown below, to enter an old file, the user types OLD. MACOS then prompts for the filename.

```
MACOS>old
 Enter file name: lens_array
  Input file lens_array.in being loaded.
 Default used for  SegXgrid
 Optical Train Summary:
  Tracing rays past   4 elements.
  Aperture grid type = Circular         with 512 grid points.
  Elt   1: LensArray   : Conic    with Kc= 0.0000D+00, Kr=-8.0000D-01, nECo-
ords =  -6
  Elt   2: Refractor   : Conic    with Kc= 0.0000D+00, Kr=-1.0000D+18, nECo-
ords =  -6
  Elt   3: FocalPlane  : Flat     with Kc= 0.0000D+00, Kr=-1.0000D+18, nECo-
ords =  -6
```

```
  Elt   4: FocalPlane  : Flat     with Kc= 0.0000D+00, Kr=-1.0000D+18, nECo-
ords =  -6
  Number of output coordinates is   5
  Too many grid points. Resetting npts to  128
MACOS>
```

MACOS also allows commands to be stacked on a single input line. The commands must be separated by a space.

```
MACOS>old lens_array show 2
 Enter file name:  lens_array
  Input file lens_array.in being loaded.
 Default used for  SegXgrid
 Optical Train Summary:
  Tracing rays past   4 elements.
  Aperture grid type = Circular         with 512 grid points.
  Elt   1: LensArray   : Conic    with Kc= 0.0000D+00, Kr=-8.0000D-01, nECo-
ords =  -6
  Elt   2: Refractor   : Conic    with Kc= 0.0000D+00, Kr=-1.0000D+18, nECo-
ords =  -6
  Elt   3: FocalPlane  : Flat     with Kc= 0.0000D+00, Kr=-1.0000D+18, nECo-
ords =  -6
  Elt   4: FocalPlane  : Flat     with Kc= 0.0000D+00, Kr=-1.0000D+18, nECo-
ords =  -6
  Number of output coordinates is   5
  Too many grid points. Resetting npts to  128
MACOS> show
 Enter number of element (0=aperture): [0]: 2

    iElt=    2
 EltName= rs
 Element= Refractor
 Surface= Conic
    fElt=   1.000000000D+18
    eElt=   0.000000000D+00
   KrElt=  -1.000000000D+18
   KcElt=   0.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00  2.000000000D-01
  RptElt=   0.000000000D+00  0.000000000D+00  1.000000000D+00
  IndRef=   1.000000000D+00
  Extinc=   0.000000000D+00
    nObs=    0
  ApType=   None
    zElt=   0.000000000D+00
PropType=   Geometric
 nECoord=   -6
MACOS>
```

MACOS provides defaults for nearly all requested parameters or values. These are usually computed from contextual information. The value of the default for a particular parameter is shown in the prompt line enclosed in square brackets. To indicate that the default is to be accepted, the user can simply hit return when the prompt is displayed. If typing ahead, a semicolon can also be used to indicate that a default should be accepted. In the example below, the default element number 0 is used to select the aperture.

```
MACOS>show;;
 Enter number of element (0=aperture): [0]:
ChfRayDir=  0.000000000D+00  0.000000000D+00  1.000000000D+00
ChfRayPos=  0.000000000D+00  0.000000000D+00 -1.000000000D+00
 zSource=   1.000000000D+22
```

```
  IndRef=   1.000000000D+00
  Extinc=   0.000000000D+00
 Wavelen=   1.000000000D-04
    Flux=   1.000000000D+00
GridType=   Circular
Aperture=   5.000000000D-01
Obscratn=   0.000000000D+00
nGridpts=  128
   xGrid=   1.000000000D+00  0.000000000D+00  0.000000000D+00
   yGrid=   0.000000000D+00  1.000000000D+00  0.000000000D+00
    nElt=    4
MACOS>
```

3.5 Graphics Display

MACOS provides in-line graphical displays on Unix machines using the PGPLOT package from CalTech (see installation notes for information on how to obtain PGPLOT). PGPLOT is started up at the first invocation of a MACOS graphics function, such as `INTensity`. PGPLOT prompts the user to define the desired form for the graphical output:

```
 MACOS>stretch
 Enter image stretch type (LINEAR, LOG10, SQRT): [LINEAR]:  log10
 MACOS>intensity
 Enter number of element where data is to be generated: [9]: 9
 Tracing   3210 rays and propagating   16384 grid points...
  Scalar FF Prop between Elt  8 and Elt  9 to WF  1:
  z1=-9.1481D+01 dx1= 1.0581D-01 min= 1.0581D-01 max= 1.0581D-01 dev=
2.6446D-06 lin=100.00%
  z2= 1.0000D+22 dx2=-1.0996D-03
 Compute time was    1.4453 sec
 Graphics device/type (? to see list, default /NULL): /xwindow
Starting /opt/local/bin/pgxwin_server.
```

The normal selection for Unix machines is "`/XWindow`." PGPLOT then sets up a graphics window in which plots are displayed. Once selected, the graphical output option cannot be changed without quitting and restarting MACOS.

3.6 Off-Line Graphics Display

MACOS graphics can also be displayed and analyzed using other applications. This is the only display option for Macintosh. It is also extremely useful for users on all platforms who wish to study their results in ways not provided by MACOS.

One such application is Matlab®, which provides an easy-to-use, powerful environment for manipulating and displaying data in many forms. Matlab is a product of The Math Works, Inc., 24 Prime Park, Natick, MA, USA (508)653-1415. We have included Matlab ".m-file" functions that provide some of the standard MACOS display types. These scripts import a specified datafile, generate a plot according to specified plot and stretch types, and return the data values. They functions also provide templates for modification or translation to other languages. By keeping the display application (e.g. Matlab) running in one window, and MACOS running in another, nearly seamless generation and display of results can easily be accomplished.

Generating the data for off-line display is as simple as selecting a file output "plot type," using the `BINARY`, `FITS`, `TEXT` or `MATLAB` commands (Section 7). Subsequent use of graphics-generating commands such as `INTensity` then create a file and store the requested data in that file. If the file already exists, the user is asked if the file is to be

overwritten; if not, a new name is requested. Other applications can then access and display the data. Repeating the previous example:

```
 MACOS>binary
  Plot type set to BINARY
 MACOS>intensity
 Enter number of element where data is to be generated: [9]: 9
 Tracing   3210 rays and propagating   16384 grid points...
  Scalar FF Prop between Elt  8 and Elt  9 to WF  1:
  z1=-9.1481D+01 dx1= 1.0581D-01 min= 1.0581D-01 max= 1.0581D-01 dev=
2.6446D-06 lin=100.00%
  z2= 1.0000D+22 dx2=-1.0996D-03
 Compute time was    1.0508 sec
 Wavefront Propagation Data Summary:
 Wavelength= 1.6280000D-04;  Source Flux= 1.0000000D+00
 u-v plane diam= 1.3438007D+01  du= 1.0581108D-01
 x-y plane diam=-1.3965216D-01  dx=-1.0996233D-03
 Peak intensity= 1.1584734D-01; Peak occurs at i=   65, j=   65
 Sum of intensity= 7.6908687D-01
 Writing Intensity, Elt=9

 DIRECT ACCESS File=tmo.int9                          Data file is  "tmo.int9"

 Binary array is of dimension   128 by   128. Magnitude of central pixel is
0.3161060810D-01
 MACOS>
```

The output file name (in this case, "tmo.int9") is nominally generated from the prescription file name, with the datatype ("int" for intensity) and element number appended after a period. Fits, text and Matlab format files have ".fit," ".txt," or ".mat" appended as well, respectively.

The Matlab command required to read the file, generate the plot (at log10 stretch), and return the data for further analysis is:

```
>>intensity=macosIMG('gray','tmo.int9','log10');
```

Here `intensity` is the returned intensity data, in this case a 128 by 128 matrix, `'gray'` is the plot type (gray scale), `'tmo.int9'` is the file name, and `'log10'` is the stretch option. The result is a plot functionally identical to what MACOS would produce with the same settings.

Other applications and environments that you may find useful for analyzing and displaying MACOS data include include SAOImage (available free from the Smithsonian Astrophysical Observatory for Unix only), Spyglass Transform, MatrixX, IDL, plotting packages such as DeltaGraph, and spreadsheets such as Excel.

3.7 PostScript and GIF Graphics Output

MACOS provides PostScript and GIF graphical file output directly from PGPLOT (Unix only). When PGPLOT is started up, at the first invocation of a MACOS graphics function, it prompts the user to define the desired form for the graphical output. Typing `?` prints a list of PGPLOT output options:

```
 Graphics device/type (? to see list, default /NULL): ?
 PGPLOT v5.0.0 Copyright 1994 California Institute of Technology
 Legal PGPLOT device types are:
```

```
/GIF   (Graphics Interchange Format file, landscape orientation)
/VGIF  (Graphics Interchange Format file, portrait orientation)
/NULL  (Null device, no output)
/PS    (PostScript file, landscape orientation)
/VPS   (PostScript file, portrait orientation)
/CPS   (Colour PostScript file, landscape orientation)
/VCPS  (Colour PostScript file, portrait orientation)
/TEK4010 (Tektronix 4010 terminal)
/RETRO (Retrographics VT640 Tek emulator)
/XTERM (XTERM Tek terminal emulator)
/XWINDOW (X window window@node:display.screen/xw)
/XSERVE  (A /XWINDOW window that persists for re-use)
/XDISP (pgdisp or figdisp server)
Graphics device/type (? to see list, default /NULL):
```

The GIF options generate separate GIF files for each graphic, named pgplot.gif, pgplot.gif_2, etc. in sequence.

The PS options generate a single file, pgplot.ps, containing all of the plots generated in a single session. It can be printed (on Unix machines) with the command:

```
% lpr pgplot.ps
```

The file can be renamed if you wish to save it.

Screen capture utilities and applications, such as SAOimage or XV, can also be very useful in recording MACOS graphical outputs.

3.8 MACOS Files

3.8.1 .In-File

An optical system prescription created in MACOS is saved in a "`filename.in`" file. This file stores the prescription in a text format which can be edited offline and reused later. The MACOS prescription files are generated by the `NEW` command (Section 4).

3.8.2 .Out-File

MACOS no longer generates a "`filename.out`" file for recording each result generated in a session.

3.8.3 Glass Tables

Glass data is provided in a text file, named `macos.glass`, which is located in a special MACOS data directory (see next section). The form and content of this file is discussed in Section 4.4.3.

Alternatively, glass data can be provided as part of the optical prescription "`.in`-file."

3.8.4 MACOS Data Directory

Beginning with Version 2.8, certain MACOS data files, such as glass tables, are being placed in a special directory. On Unix systems, the path to this directory can be specified by the user by defining an environment variable, `MACOS_DATA`. This is done in response to the Unix shell prompt, as in:

```
> setenv MACOS_DATA /home/macos/data/
```

The above command establishes the directory `/home/macos/data` as the location of the data files for subsequent MACOS sessions. The user may specify any directory as `MACOS_DATA`.

If `MACOS_DATA` is not specified, 3 other directories will be searched for data files. In order these are:

1. Directory `macosData` in the user's home directory
2. Directory `/home/comp/macosData`
3. The current directory

If the required data file is not found in any of these directories, a warning message is issued. In this event, the user is required to supply all relevant data, such as glass coefficients, through the prescription file.

3.8.5 Data Export

The `EXPort` command writes certain *variables* to a file, in several different formats (Section 7 discusses writing *graphics* data to files). The variables are useful when linking linear models to thermal or structural codes (see Section 8).

The following variables can be exported:

- `CMAtrix`: optical sensitivity matrix in global or local coordinates
- `CmDM`: deformable mirror sensitivity matrix
- `RayPos`: current position 3-vectors for each ray
- `RayDir`: current ray direction 3-vectors for each ray
- `CumRayL`: current optical pathlengths for each ray

Files may be saved in the following formats:

- `BINary`: data is saved to a Fortran binary file, `filename.bin`
- `TEXt`: data is saved to an ASCII text file, `filename.txt`
- `MFILE`: data is saved to a Matlab file, `filename.m`

The following is a sample `EXPort` session.

```
MACOS>build ;
 Enter terminal element number: [1]:
 Tracing  12669 rays but not BUILDing linear model.
 BUILD limited to    500 rays.  MOD npts and recompute.
 MACOS>export
Current element is   1
  Enter EXPORT mode (BINary, TEXt, MFIle, ISM, H or Quit):  [BINARY]: tex
 Exporting data to lens_array.txt
 Export data for CHFRay only, ALLRays, or NOChfray?  [CHFRay]: chfr
  EXPORT (CMAtrix, CmDM, RAYPos, RAYDir, RAYL, H or Q):  [QUIT]: raypos
% RayPos 3-vectors for       1 rays at Element     1
  EXPORT (CMAtrix, CmDM, RAYPos, RAYDir, RAYL, H or Q):  [QUIT]: q
MACOS>
```

3.9 Using Macros

MACOS has a scripting or macro capability. This is provided by keystroke recording and play-back commands, `JOUrnal` and `EXEcute`. Macros can call other macros, up to 10 deep. Comments (anything following a "%" on a line) and blank lines are ignored. Looping, conditionals and other "language" capabilities are not currently supported.

3.9.1 Journal Files

The `JOUrnal` command is used to start and stop a journal of all the commands the user inputs to the terminal. Typing the `JOUrnal` command initiates keystroke recording, which continues until toggled off by typing the `JOUrnal` command a second time. `JOUrnal` allows the user to keep track of the commands entered in a MACOS session. The resulting .jou-file can later be used as a macro command (see Section 3.9.2). The .jou-file can also be edited and modified offline.

The `JOUrnal` command has one argument, the filename. The filename entered by the user will automatically have the file extension `.jou` appended, to denote that the file is a `JOUrnal` file. If the file already exists, then the contents of the file are deleted before the `JOUrnalizing` begins. It is imperative that the user select a unique filename, if is it desired that old `JOUrnal` files be kept.

3.9.2 Executing Macros

The `EXEcute` command allows the user to designate a .jou-file from which further MACOS commands are read. Each command (including other `EXEcute` commands) is executed in sequence, until an END OF FILE symbol is read from the .jou-file. MACOS then returns to interactive mode and the user can once again enter commands from the keyboard.

The `EXEcute` command has one argument, the filename of the batch file. The filename must have the file extension `.jou` to denote that the file is an journal file.

The user can edit a file produced with the `JOUrnal` comand to produce a macro. For example, if the .jou-file was created with an OPD at the 10th element, it is trivial to modify it to calculate the OPD at the 14th element. The user only needs to change the 10 in the .jou-file to a 14 using a text editor.

Examples are provided in the macosDemo directory distributed with the MACOS code.

3.10 Memory Requirements

The MACOS application currently uses a fixed memory allocation set at compile time. Different run modules are supplied for different sized problems, depending on the machine and available memory. Table 4 summarizes requirements for specific platforms.

TABLE 4. Memory Requirements

executable	Sun/Solaris	PowerMacintosh
macos128	9.75 MB	
macos256	18.3 MB	
macos512	38.4 MB	
macos1024	115 MB	

SECTION 4 Describing Optical Systems

A first MACOS session typically begins by invoking the NEW command. This begins a dialog whose function is to define the data used by MACOS in its computations. This data comprises the optical prescription, and is saved by MACOS as an ".in-file," which can be edited and reused later using the OLD command. The contents of the .in-file can be altered within a MACOS session using MODIFY, PERTURB or other commands, and saved as a new .in-file using the SAVE command. The .in-files can also be created and edited off-line using a text editor. The commands covered in this section are:

```
NEW
DRAw
OLD (LOAd)
SUMmarize
STAtus
RESet
SHOw
MODify
BEAM
SAVe
```

4.1 Specifying the Light Source

The NEW dialog begins by asking the user for information about the light source, such as where it is located, in what direction it is oriented, and what its optical properties are. After the source is defined, similar questions are asked about each optical element in turn. This section is intended to help the user provide the appropriate answers to these questions. It is also intended to define MACOS variable names, which are used in the .in-files to specify aspects of the prescription.

MACOS represents light sources as *beams*, consisting of a ray bundle and an associated complex amplitude matrix (or matrices). Beams can be collimated or focused, as per Figure 11. In addition, the illumination profile of a beam can be specified as uniform, gaussian, dipole or cosine-to-a-power, using the BEAM command discussed in a later section. Collimated beams are composed of a bundle of parallel rays, originating at a plane in front of the entrance aperture. Collimated beams are used to represent the light received from a point-source at an infinite distance from the system. They are also used when collimated light sources, such as some lasers, are to be modeled. Focused beams are bundles of rays converging to or diverging from a point in space and represent the light from a nearby point source. Focused beams are defined on a spherical surface whose center of curvature is at the source point.

The properties that must be specified by the user include:

- Source position and direction, defined by the position and orientation of the "chief" ray (really the gut ray – the ray that lies in the center of the beam)
- Aperture size (collimated source) or angle (point source)

- Aperture shape (circular, square, hexagonal, pie, flower)
- Illumination profile (flat, gaussian, dipole or cosine-to-a-power)
- Wavelength
- Flux through the defined aperture
- Polarization state (if using polarization ray-trace analysis functions)
- Segmentation (unsegmented, hexagonal segments, or "pie" segments)
- Coordinates to define the orientation of the beam to the optical elements

4.1.1 Source Position and Orientation

In MACOS, a *light source* is a surface but not an element. The surface is defined by a single wavefront emitted from a single point source. The `zSource` parameter is the distance from the origin of the light to the light source surface, as defined by the location of the chief ray, `ChfRayPos` (see Figure 11). Collimated sources are specified by `zSource=1d22`. For point sources, the convention is that the sign of `zSource` is positive for a converging beam and negative for an expanding beam.

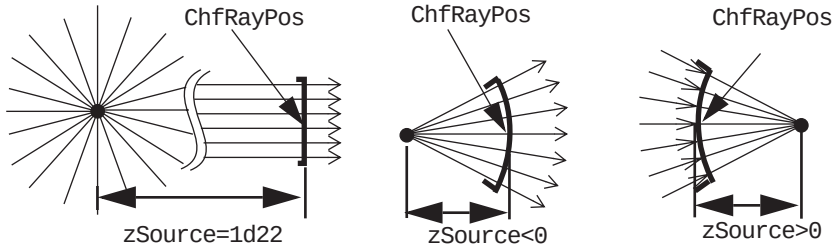


FIGURE 11 Light sources

The location and orientation of the light source is determined from the *chief ray.* The MACOS "chief ray" is the central or gut ray of the beam, as discussed in Section 2.2. It is the center ray of the incident beam bundle. It is normally a true chief ray as well, passing through the center of the system stop. This condition is enforced by the user, using the `STOP` or `CENTER` commands, as described later.

To define the source, the user must specify in the global coordinate system:

- Chief ray position vector
- Chief ray direction vector

These are set by the 3-vector inputs `ChfRayPos` and `ChfRayDir`. If the source is collimated, then every ray in the beam starts out with its direction equal to `ChfRayDir`. The position of each ray is calculated from `ChfRayPos`, the shape and type of aperture, and the specified ray density. If the source is a point source, appropriate position and direction is computed so that each ray starts from the surface of a sphere centered at the source point.

NOTE: Converging or diverging sources **should not be located at the source point**, or, for example, divide-by-zero errors will result when performing diffraction analysis.

4.1.2 Refractive Index and Extinction Coefficient at the Source

The refractive index is, in general, a complex number. The real part is the conventional index, giving the speed of light in the particular medium, and so defining the direction changes a ray experiences at refractive interfaces. The imaginary part is the extinction

coefficient, which defines the attenuation of the field in the medium. Only the index (real part) is significant in MACOS scalar ray-tracing calculations. The extinction coefficient is used only when polarization ray-trace is selected, by use of the POL command.

- The real part of the refractive index of the source medium is specified by IndRef. If ChfRayPos is in a vacuum, then IndRef=1. For many applications it is sufficient to approximate the index of refraction of air as 1, although the exact value depends on the pressure and temperature.
- The negative of the imaginary part of the refractive index, the extinction coefficient, of the source medium is specified by Extinc. For lossless media such as vacuum and air, set ExtInc=0. For perfect metals, set ExtInc=1d22. This parameter is used only when polarization ray-tracing is turned on.

The user must specify the index explicitly for the source only. For other elements, particular glasses can be specified, and the index will be varied as a function of the wavelength of the beam. The source IndRef and Extinc are not changed automatically with changes in wavelength. The user can implement such changes as needed using the MODIFY command.

4.1.3 Wavelength

The wavelength is specified by the wavelen variable. For most calculations, MACOS uses one wavelength at a time. Multiple-wavelength images can be generated using the COMPOSE command to define detector parameters, using use the MODify command to change the wavelen (and Flux) variables (see Section 4.10.2), then using the ADD command to separately propagate and accumulate different wavelength and flux level images at the detector. This procedure can be automated by using ".filt" filter files and the MULtispec command.

4.1.4 Flux

The total flux of the source illumination through the aperture area (defined by the Aperture, Obscuration and GridType parameters) is specified by Flux. This parameter is used in MACOS to facilitate the radiometric modeling of systems. It is useful when constructing composite images of multiple sources or at multiple wavelengths. This procedure can be automated by using ".filt" filter files and the MULtispec command.

4.1.5 Aperture

The aperture size is set by the Aperture parameter, whose interpretation is different for collimated or point sources. If the source is collimated (zSource=1d22), Aperture is the *diameter* of the input beam. If it is a point source(zSource<1d22), Aperture is the full *angle* (entered in radians like all angles in MACOS) subtended by the two marginal rays (see Figure 12). Thus, the numerical aperture of the system is

$$N.A. = n*\sin(Aperture) \tag{4.1}$$

where n is the index of refraction.

Obscurations can be defined in the source beam if desired. Obscuration size is specified in the same way as the aperture, with the parameter Obscratn setting the diameter or angle of a *central* obscuration.

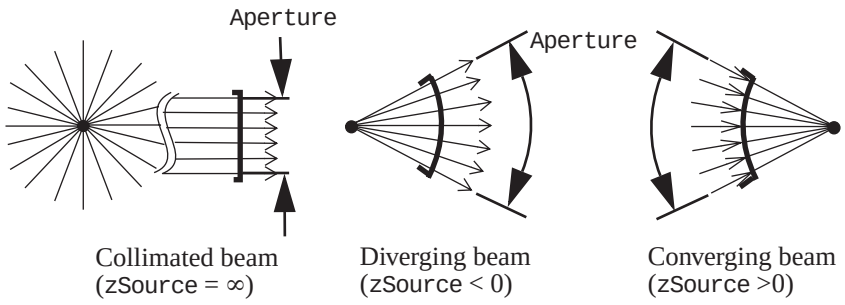


FIGURE 12 Aperture sizing

NOTE: This aperture obscuration option can be useful, but is not the best way to represent most beam-train obscurations. It is better to explicitly place obscurations on the appropriate elements as discussed in Section 4.6.

The entrance aperture can have several shapes, as shown in Figure 13, depending on whether the system is monolithic – in which all optics are of one piece – or segmented – in which at least one mirror is split into separate segments. For monolithic systems, the aperture shape can be round or square, as set by the `GridType` parameter. For segmented systems the options are hexagonal array, round with 6 pie-shaped outer segments, or round with an arbitrary number of outer segments. For computational efficiency, the segmentation is defined at the light source.

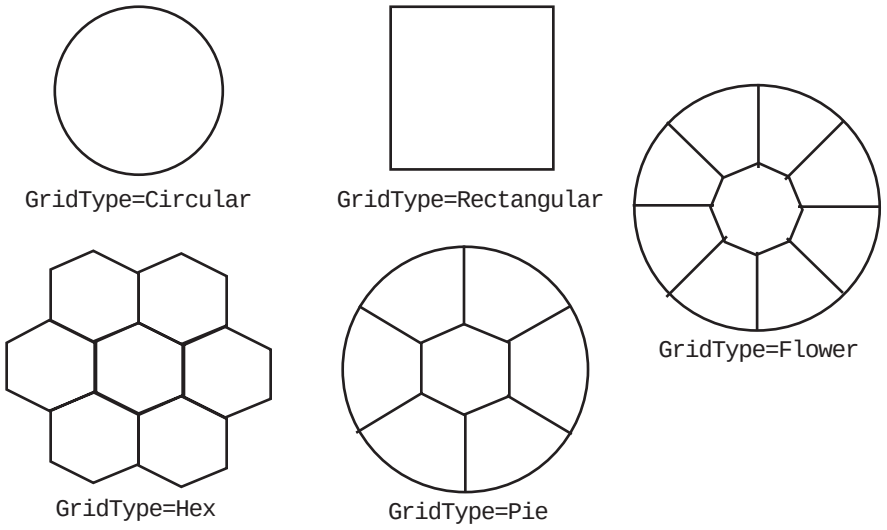


FIGURE 13 Aperture shapes

It is not necessary to segment the aperture to use lenslet arrays, as discussed later.

NOTE: Registration of a segmented aperture on a segmented mirror may shift when the beam incidence angle changes, or when intervening optics are perturbed. This can cause rays that nominally strike one mirror segment to move to another. In these conditions, the user should check to ensure proper registration.

4.1.6 Ray Grids

The density of rays across the beam and the orientation of the ray grid are set by three parameters:

- Number of ray grid points across the aperture is set by `nGridPts`
- Ray grid coordinate vectors `xGrid` and `yGrid` determine the global orientation of the ray grid and so the beam coordinates

The ray grid is laid out on the surface that defines the light source, according to a coordinate system for which `zGrid` equals the negative chief ray direction `ChfRayDir` and `xGrid` is as specified by the user (see Figure 14). The number of rays across the aperture is set by `nGridPts`.

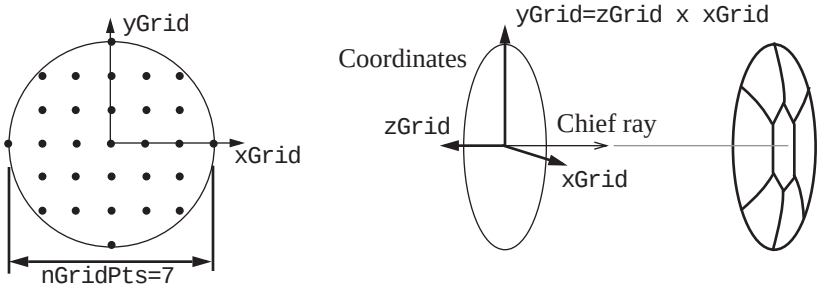


FIGURE 14 Ray grid layout

4.2 Example Light Source

MACOS obtains the information it needs to build an optical prescription (an ".in-file" dataset) through the `NEW` file definition dialog. The first questions in this dialog are about the light source and the latter questions pertain to the optical elements.

For the first example, we start the prescription for a Cassegrain telescope, `Cassegrain.in`, shown in Figure 15. The light source is a point source at infinity so the wavefront incident on the telescope is collimated. In the example dialog, the prompts are exactly as MACOS gives them. The correct user response is indicated in **bold** text, and the name of the variable being set is in ***italic*** text.

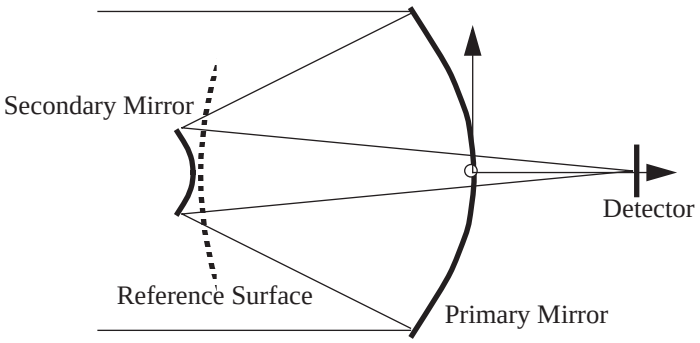


FIGURE 15 Cassegrain telescope layout

```
     Prompt                    Response                                  VariableName
MACOS>new
 Enter file name: Cassegrain
 File Cassegrain.in being created.                                          filenam
 Enter chief ray input direction vector (x,y,z)  [0.,0.,1.0]:0,0,1        ChfRayDir
 Enter chief ray location vector (x,y,z)  [0.,0.,0.]:0,0,-5               ChfRayPos
 Enter distance from source  [1.000000E+22]:1d22                            zSource
 Enter chief ray index of refraction  [1.00000]:1                           IndRef
 Enter chief ray extinction coefficient  [0.]:0                             Extinc
 Enter monochromatic wavelength  [6.000000E-07]:1d-6                        Wavelen
 Enter intensity (flux over aperture area)  [1.00000]:1                        Flux
 +-----------------------------------------+
 | Source ray grid types:                  |
 |   Circular       Square         Hex     |
 |   Pie            Flower                 |
 +-----------------------------------------+
 Enter source grid type  [Circular]: Circular                             GridType
 Enter aperture diameter or angle  [1.00000]:4                            Aperture
 Enter number of grid points  [65]:15                                     nGridPts
 Enter input plane x-axis vector (x,y,z)  [1.00000,0.,0.]:-1,0,0             xGrid
 Enter input plane y-axis vector (x,y,z)  [0.,1.00000,0.]:0,-1,0             yGrid
 Enter number of elements  [no default]: 5                                    nElt
```

This dialog produces the following lines in the prescription file `Cassegrain.in`:

```
ChfRayDir=  0.000000000D+00  0.000000000D+00  1.000000000D+00
ChfRayPos=  0.000000000D+00  0.000000000D+00 -5.000000000D+00
 zSource=   1.000000000D+22
  IndRef=   1.000000000D+00
  Extinc=   0.000000000D+00
 Wavelen=   1.000000000D-06
    Flux=   1.000000000D+00
GridType=   Circular
Aperture=   4.000000000D+00
Obscratn=   0.000000000D+00
nGridpts=   15
   xGrid=  -1.000000000D+00  0.000000000D+00  0.000000000D+00
   yGrid=   0.000000000D+00 -1.000000000D+00  0.000000000D+00
    nElt=    5
```

The `Cassegrain.in` example is finished in Section 4.8.

Now consider the case of a non-collimated input beam, `zSource` not equal to infinity. A Cassegrain telescope has been turned into a beam projector. The light source is now located where the detector was previously located as shown in Figure 16.

```
MACOS>new
 Enter file name:  zsourc
 File zsourc.in being created.
 Enter chief ray input direction vector (x,y,z)  [0.,0.,1.0]:0,0,1        ChfRayDir
 Enter chief ray location vector (x,y,z)  [0.,0.,0.]:0,0,-1               ChfRayPos
 Enter distance from source  [1.000000E+22]:-0.1                            zSource
 Enter chief ray index of refraction  [1.00000]:1                           IndRef
 Enter chief ray extinction coefficient  [0.]:0                             Extinc
 Enter monochromatic wavelength  [6.000000E-07]:1d-3                        Wavelen
 Enter intensity (flux over aperture area)  [1.00000]:1                        Flux
 +-----------------------------------------+
 | Source ray grid types:                  |
 |   Circular       Square         Hex     |
 |   Pie            Flower                 |
 +-----------------------------------------+
 Enter source grid type  [Circular]: Circular                             GridType
 Enter aperture diameter or angle  [1.00000]:0.1076                       Aperture
 Enter number of grid points  [65]:25                                     nGridPts
 Enter input plane x-axis vector (x,y,z)  [1.00000,0.,0.]:-1,0,0             xGrid
```

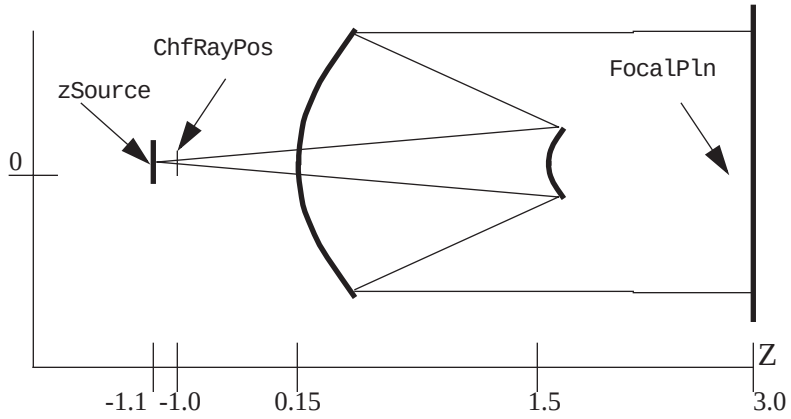


FIGURE 16 Non-collimated light source example

```
 Enter input plane y-axis vector (x,y,z)  [0.,1.00000,0.]:0,-1,0          yGrid
 Enter number of elements  [no default]: 6                                nElt
```

This results in the following lines in the prescription file `zsource.in`:

```
ChfRayDir=  0.000000000D+00  0.000000000D+00  1.000000000D+00
ChfRayPos=  0.000000000D+00  0.000000000D+00 -1.000000000D+00
 zSource=  -1.000000000D-01
  IndRef=   1.000000000D+00
  Extinc=   0.000000000D+00
 Wavelen=   1.000000000D-03
    Flux=   1.000000000D+00
GridType=   Circular
Aperture=   1.076000000D-01
Obscratn=   0.000000000D+00
nGridpts=   25
   xGrid=  -1.000000000D+00  0.000000000D+00  0.000000000D+00
   yGrid=   0.000000000D+00 -1.000000000D+00  0.000000000D+00
    nElt=    6
```

Notice that the `Aperture` entry has changed significantly, since `Aperture` now represents an angle, instead of the diameter of the input beam. The `ChfRayPos` is now a small distance in front of the source, so `zSource` is now negative to indicate an diverging beam.

4.3 Specifying Optics

The next part of the `new` dialog is the specification of the optical elements. The first step in defining the beam train is actually the last step in the source dialog: specifying the number `nElt` of optical elements in the beam train. The number of elements is equal to the number of active optical surfaces, plus any reference (dummy) surfaces, plus 2 return surfaces for an exit pupil, if desired.

The `new` dialog continues by prompting the user for all data needed to define each optic. Optics are specified one at a time, in the sequence in which they are encountered by the light from the source (except nonsequential and segmented optics). Some optics, such as mirrors, are realized by a single *element*, or optically active surface. Other optics require

two or more elements. Lenses require two refracting elements, one for each surface (see the example in Section 4.4.2). Corner cubes require three non-sequential reflecting elements. Each element in multiple-element optics is entered separately.

The first step in specifying an element is to define its *type* (e.g., `Reflector`) and its *surface type* (e.g., `Conic`). Table 5 lists the elements currently supported by MACOS, and Table 6 lists the surface types. Further data required to define the surfaces may be requested, depending on specified surface type.

TABLE 5. Element types

Element Type (`Element=...`)	**Allowed Surface Types**	**Description**
`Reflector`	all	reflective surfaces (e.g., mirrors)
`Refractor`	all	refractive surfaces (e.g., lenses)
`LensArray`	conics or aspheres	lens arrays
`Grating`	all	reflective diffraction gratings
`HOE`	all	reflective holographic optical elements
`NSReflector`	conics	non-sequential reflective surfaces
`NSRefractor`	conics	non-sequential refractors
`Segment`	all	segmented mirrors
`ObsSrf`	conics	obscuring surfaces
`FocalPln`	flats	focal planes
`RefSrf`	conics	reference surfaces
`ReturnSrf`	conics	return surfaces

TABLE 6. Surface types

Surface Type (`Surface=...`)	**Surface Description**
Flat	Planar
Conic	Conicoid of revolution. Precise shape is defined by conic constant `KcElt` and radius `KrElt` parameters (or `fElt` and `eElt`)
Toric	Toric or cylindrical shapes, defined by separate radius parameters in x and y directions.
Anamorphic	“Saddle” shapes, defined by separate conic and radius parameters in x and y directions.
Aspheric	Base conicoid, with 10th-order aspheric deviation.
Monomial	Base conicoid, with 14th-order cartesian polynomial deviation.
Zernike	Base conicoid, with deviation defined by Zernike polynomials.
Interpolated	Base conicoid, with deviation defined by X-Y-Z data triplets
GridData	Base conicoid, with deviation defined by square matrix pixels
User-defined	Deformable mirrors defined using influence functions.

The second step is to define element location, orientation and rotation point, as discussed in Section 2.1:

- Element *location* is specified by the `VptElt` vector. This is a three-vector in global coordinates giving the location of the vertex of the element.

- Element *orientation* is specified by the `psiElt` vector. This is a unit magnitude three-vector in global coordinates giving the direction of the principal axis of the element.
- Element *rotation point*, `RptElt`, defines the pivot point for rotations of the element implemented using the `Perturb` command. It also defines the coupling between rotation and translation for the `Build` command. Together with any element coordinates, `Telt`, `RptElt` governs the element response to perturbations.

The third step is to define any aperture or obscurations that might occur at the element. Apertures define the margins of the element, and can be circular or rectangular. Obscurations are circular or rectangular regions on the surface of the element where light is completely absorbed. Negative obscurations "un-absorb" the light, countering the effect of an obscuration in regions where they overlap. Obscurations and apertures can be placed anywhere on the surface.

The fourth step is to define the `zElt` and `PropType` diffraction propagation parameters, which are needed if the element is at the start or end of a diffraction propagation.

The final step is to specify element coordinates (`Telt`), if they are desired.

In the next several subsections, each of these steps is discussed more carefully, and examples are shown.

4.4 Element Types

Element type and surface shape are each most critical in the determination of the properties of an optic. In this subsection we define the MACOS element types and illustrate their use through a sequence of specific examples.

4.4.1 Reflectors

Mirrors are entered as type `Reflector`. Reflectors can have any surface type or shape; the user will be prompted for different parameters depending on which surface type is chosen. No element-specific parameters are required for reflectors unless polarization ray-tracing is turned on. In that case, `Extinc` and `IndRef` should be carefully specified, as they determine the transmission of the beam past the reflective interface.

This example shows how to enter a flat mirror in MACOS:

```
Enter Element   1 Data:
Enter element name (no blanks)  [Elt]: flatMirror                    EltName
+-----------------------------------------+
| Element types:                          |
|   Reflector      NSReflector    Segment |
|   Refractor      NSRefractor    LensArray|
|   FocalPlane     Reference      Return   |
|   HOE            Grating        Obscuring|
+-----------------------------------------+
Enter element type :reflect                                          Element
+-----------------------------------------+
| Surface types:                          |
|   Flat           Conic          Aspheric |
|   Anamorphic     Zernike        Monomial |
|   Interpolated   UserDefined             |
```

```
+-----------------------------------------+
Enter surface type :flat                                        Surface
Enter element vertex location (x,y,z)  [no default]: 0,0,0      VptElt
Enter element rotation point (x,y,z)  [0.,0.,0.]:0,0,0          RptElt
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,-1   psiElt
Do you want an aperture on this element? [NO]: no
Enter number of obscurations  [0]:0
Enter Fresnel propagation distance  [1.000000E+22]:1d22           zElt
+-----------------------------------------+
| Propagation types:                      |
|   Geometric      GeomUpdate     FarField   |
|   NFSpherical    NFS1surf                  |
|   NFPlane        NFP1surf                  |
|   NF1            NF2                       |
|   SpatialFilter  SF1surf                   |
+-----------------------------------------+
Enter propagation type  [Geometric]: Geometric                  PropType
Do you wish to use element-based coordinates?: [NO]: no
Enter element index of refraction  [1.00000]:1                  IndRef
Enter element extinction coefficient  [1.000000E+22]:1d22       Extinc
```

This results in the following lines in the prescription file:

```
   iElt=    1
 EltName= flatMirror
 Element= Reflector
 Surface= Flat
    fElt=   1.000000000D+22
    eElt=   0.000000000D+00
   KrElt=  -1.000000000D+22
   KcElt=   0.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  RptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  IndRef=   1.000000000D+00
  Extinc=   1.000000000D+22
    nObs=    0
  ApType=   None
    zElt=   1.000000000D+22
PropType=   Geometric
 nECoord=   -6
```

4.4.2 Refractors

A refracting element is the interface between two media of differing index of refraction, such as air and glass. Rays traversing the interface are, of course, bent at the interface. lenses, windows and other refracting optics are actually modeled using 2 optical elements, both of type `Refractor`. The first element defines the front surface of the optic and sets the index for the ray traveling inside the optic. The second element defines the back surface, and resets the index to the value outside the optic.

Some refracting elements, such as prisms, use more than two elements, and may require use of nonsequential reflectors and refractors as well – see Section 4.4.13. Arrays of lenslets are covered in Section 4.4.4.

As with reflectors, refractors in MACOS may have any surface shape. The index of refraction `IndRef` can be set directly in the prescription, or indirectly, by specifying a particular glass for the element. To set the index directly, select `FixedIndex` glass type and then specify the index. No other element-specific parameters are required, unless polarization ray-tracing is turned on. In that case, `Extinc` must be specified as well.

This example lens has a spherical front surface and a planar rear surface, as illustrated in Figure 17:

```
 Enter Element   1 Data:
 Enter element name (no blanks)  [Elt]: lens_front                    EltName
 +-----------------------------------------+
 | Element types:                          |
 |   Reflector      NSReflector    Segment |
 |   Refractor      NSRefractor    LensArray|
 |   FocalPlane     Reference      Return   |
 |   HOE            Grating        TrGrating|
 |   RfPolarizer    TrPolarizer    Obscuring|
 +-----------------------------------------+
Enter element type :refractor                                        Element
 +-----------------------------------------+
 | Surface types:                          |
 |   Flat           Conic          Aspheric |
 |   Anamorphic     Zernike        Monomial |
 |   Interpolated   UserDefined    Grid data|
 |   Toric          InfluenceFcn           |
 +-----------------------------------------+
Enter surface type :conic                                            Surface
 Enter element vertex location (x,y,z)  [no default]: 0,0,0           VptElt
 Enter element rotation point (x,y,z)  [0.,0.,0.]:0,0,0               RptElt
 Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,-1        psiElt
 Enter e&f or Kc&Kr (1=e&f,2=Kc&Kr)?  [1]:1
 Enter element focal length  [1.000000E+22]:1                           fElt
 Enter element eccentricity  [0.]:0                                     eElt
 Do you want an aperture on this element? [NO]: no
 Enter number of obscurations  [0]:0
 Enter Fresnel propagation distance  [1.000000E+22]:1d22                zElt
 +-----------------------------------------+
 | Propagation types:                      |
 |   Geometric      GeomUpdate     FarField |
 |   NFSpherical    NFS1surf               |
 |   NFPlane        NFP1surf               |
 |   NF1            NF2                    |
 |   SpatialFilter  SF1surf                |
 +-----------------------------------------+
 Enter propagation type  [Geometric]: Geometric                      PropType
 Do you wish to use element-based coordinates?: [NO]: no
 Enter glass type [FixedIndex]: BK7                                  GlassElt


Enter Element   2 Data:
 Enter element name (no blanks)  [Elt]: lens_back                    EltName
 +-----------------------------------------+
 | Element types:                          |
 |   Reflector      NSReflector    Segment |
 |   Refractor      NSRefractor    LensArray|
 |   FocalPlane     Reference      Return   |
 |   HOE            Grating        TrGrating|
 |   RfPolarizer    TrPolarizer    Obscuring|
 +-----------------------------------------+
Enter element type :refractor                                        Element
 +-----------------------------------------+
 | Surface types:                          |
 |   Flat           Conic          Aspheric |
 |   Anamorphic     Zernike        Monomial |
 |   Interpolated   UserDefined    Grid data|
 |   Toric          InfluenceFcn           |
 +-----------------------------------------+
```

```
Enter surface type :flat                                              Surface
Enter element vertex location (x,y,z)  [no default]: 0,0,0.1          VptElt
Enter element rotation point (x,y,z)  [0.,0.,0.]:0,0,0                RptElt
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,-1         psiElt
Do you want an aperture on this element? [NO]: no
Enter number of obscurations  [0]:0
Enter Fresnel propagation distance  [1.000000E+22]:1d22                 zElt
+-----------------------------------------+
| Propagation types:                      |
|   Geometric      GeomUpdate     FarField |
|   NFSpherical    NFS1surf                |
|   NFPlane        NFP1surf                |
|   NF1            NF2                     |
|   SpatialFilter  SF1surf                 |
+-----------------------------------------+
Enter propagation type  [Geometric]: Geometric                        PropType
Do you wish to use element-based coordinates?: [NO]: no
Enter glass type [FixedIndex]: FixedIndex                             GlassElt
Enter element index of refraction  [1.00000]:1                          IndRef
Enter element extinction coefficient  [0.00000]:0                       Extinc
```

NOTE: It is not uncommon to “lose” rays, if they miss the properly defined area of a lens, that is, if the rays miss the region where the 2 surfaces of the lens are sequential. If rays extend beyond the intersection point of the two elements, the rays become undefined and are not traced further as shown in Figure 17.

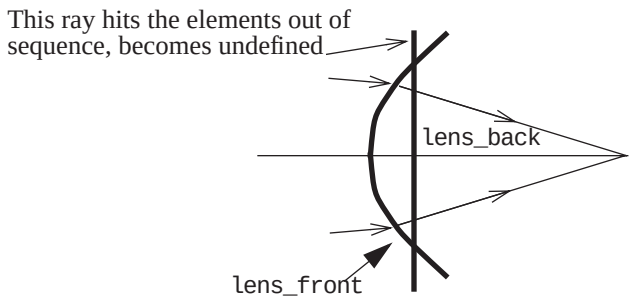


FIGURE 17 Ray bundle must pass within surface intersection points of the refracting elements for all rays to be traced properly

4.4.3 Glass Tables

Version 2.8 of MACOS comes with a single glass table, which incorporates the Schott glass types as of May 1999 (from `http://schottglass.com`). This table is a text file, `macos.glass`, located in the MACOS data directory. See Section 3.8.4 for details on how to specify this directory.

The user may specify the index of glasses or other optical materials in several ways:

- By selecting from the materials in the provided `macos.glass` file, as in the first part of the example in Section 4.4.2.
- By using a different `macos.glass` file, located in the MACOS data directory. The user can easily add other materials to a private copy of `macos.glass`, using the `MACOS_DATA` environment variable to define its location.
- By directly specifying the 6 Sellmeier glass coefficients in the prescription. These coefficients are `B1`, `B2`, `B3`, `C1`, `C2`, `C3` from the manufacturers catalog. Note that the MACOS `SAVE` command saves the coefficients of each glass in the prescription file, even if the glass data was originally derived from the MACOS glass table.

- By declaring a material to be FixedIndex, and then specifying the index, in the prescription. This approach uses a single value of the index for any wavelength of light.

The form of the glass table is simple. Each glass is defined in a single row of the file, with the first entry being the name of the glass, followed by the 6 Sellmeier coefficients. The following shows the first few entries from macos.glass:

```
Air    0d0  0d0  0d0  0d0  0d0  0d0
BaF13  1.57068590E+00  1.51295152E-01  1.07638026E+00  9.39377487E-03  4.49479866E-
02  1.13934126E+02
BaF3  1.32064267E+00  1.33572683E-01  8.85521821E-01  8.87798715E-03  4.20290346E-
02  1.11729167E+02
BaF4  1.37492590E+00  1.46354500E-01  9.12643359E-01  9.29554343E-03  4.40426654E-
02  1.16607899E+02
```

4.4.4 Lens Arrays

Lens arrays (a.k.a. lenticular arrays or lenslet arrays) are refracting optics. Usually, a lens array will have one flat surface and one surface figured into a mosaic of regularly distributed lenslets (Figure 18). Lens arrays are used most often in Hartmann wavefront sensors. An example adaptive optics wavefront sensor is provided in Appendix A.4.

MACOS uses a LensArray element type to model the bumpy surface of a lens array. LensArray elements are specified much as Refractor elements are, with the addition of a few new parameters. LensArray elements can have conic or aspheric lenslet shapes. Enter shape parameters that describe a single lenslet, and all the other lenslets will be automatically replicated. The location and orientation parameters VptElt and psiElt define the plane containing all of the lenslet vertices, and specify the orientation of each lenslet.

The first new parameter is LensArrayType, which can have two values: LensArrayTyp=1 for hexagonal arrays; and LensArrayTyp=2 for rectangular arrays. The second new parameter is LensArrayWidth, which is the width of a single lenslet (flat to flat for hexagonal lenslets) (Figure 18).

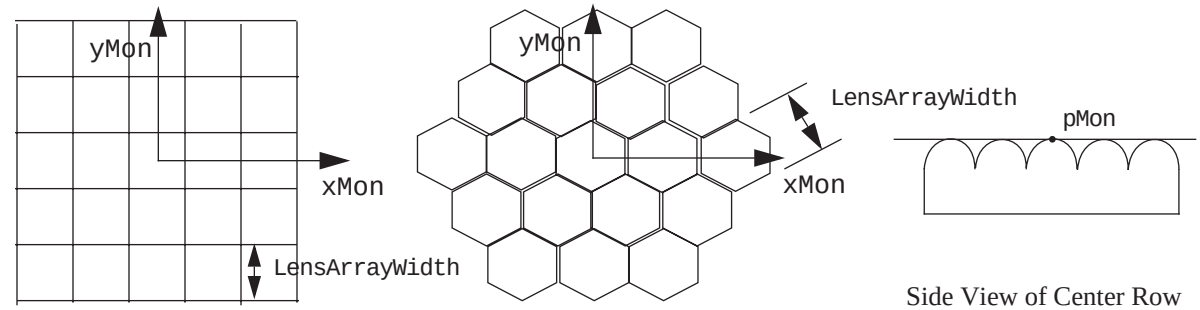


FIGURE 18 Hexagonal lens array

LensArray elements use a surface coordinate system to define the location and orientation of the lenslet grid on the underlying plane. The variable pMon is the center of the coordinate system, defining the vertex position of the "central" lenslet. pMon can differ

from VptElt, though only the projection of pMon onto the plane is significant. The variables xMon, yMon, and zMon define the surface coordinate frame as shown in Figure 27.

An additional example of the use of lenslet arrays is provided in Appendix A.4.

```
Enter element name (no blanks)  [Elt]: LensArray                          EltName
+-----------------------------------------+
| Element types:                          |
|   Reflector      NSReflector    Segment |
|   Refractor      NSRefractor    LensArray|
|   FocalPlane     Reference      Return   |
|   HOE            Grating        Obscuring|
+-----------------------------------------+
Enter element type :LensArray                                            Element
+-----------------------------------------+
| Surface types:                          |
|   Flat           Conic          Aspheric |
|   Anamorphic     Zernike        Monomial |
|   Interpolated   UserDefined             |
+-----------------------------------------+
Enter surface type :Conic                                                Surface
Enter element vertex location (x,y,z)  [no default]: 0,0,0                VptElt
Enter element rotation point (x,y,z)  [0.,0.,0.]:0,0,0                    RptElt
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,1              psiElt
Enter e&f or Kc&Kr (1=e&f,2=Kc&Kr)?  [1]:2
Enter Conic Constant  [0.]:0                                               KcElt
Enter Radius  [-1.000000E+22]:-0.8                                         KrElt
Do you want an aperture on this element? [NO]:
Enter number of obscurations  [0]:
Enter Fresnel propagation distance  [0.800000]:1d22                        zElt
+-----------------------------------------+
| Propagation types:                      |
|   Geometric      GeomUpdate     FarField |
|   NFSpherical    NFS1surf                |
|   NFPlane        NFP1surf                |
|   NF1            NF2                     |
|   SpatialFilter  SF1surf                 |
+-----------------------------------------+
Enter propagation type  [Geometric]: Geometric                          PropType
Do you wish to use element-based coordinates?: [NO]:
Enter glass type [FixedIndex]: FixedIndex                               GlassElt
Enter element index of refraction  [1.00000]:1.5                          IndRef
Enter element extinction coefficient  [0.]:0                              Extinc

Enter surface reference point location (x,y,z)  [0.,0.,0.]:0,0,0            pMon
Enter surface x-axis unit vector (x,y,z)  [1.00000,0.,0.]:1,0,0             xMon
Enter surface y-axis unit vector (x,y,z)  [0.,1.00000,0.]:0,1,0             yMon
Enter surface z-axis unit vector (x,y,z)  [0.,0.,1.00000]:0,0,1             zMon
 Orthoganalized zMon=  0.  0.    1.0000000000000
Enter lenslet width  [1.00000]:0.1                                  LensArrayWidth
Enter lens array type (1=hex, 2=square)  [1]:1                       LensArrayType
```

This results in the following lines in the prescription file:

```
   iElt=    1
EltName= LensArray
Element= LensArray
Surface= Conic
   fElt=   8.000000000D-01
   eElt=   0.000000000D+00
  KrElt=  -8.000000000D-01
  KcElt=   0.000000000D+00
 psiElt=   0.000000000D+00  0.000000000D+00  1.000000000D+00
 VptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
 RptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
 IndRef=   1.500000000D+00
```

```
  Extinc=   0.000000000D+00
    pMon=   0.000000000D+00  0.000000000D+00  0.000000000D+00
    xMon=   1.000000000D+00  0.000000000D+00  0.000000000D+00
    yMon=   0.000000000D+00  1.000000000D+00  0.000000000D+00
    zMon=   0.000000000D+00  0.000000000D+00  1.000000000D+00
LensArrayType=  1
LensArrayWidth=   1.000000000D-01
    nObs=    0
  ApType=   None
    zElt=   1.000000000D+22
PropType=   Geometric
 nECoord=   -6
```

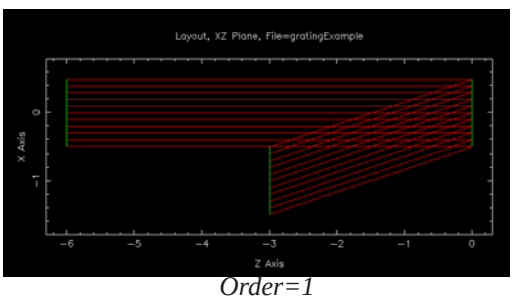
4.4.5 Reflective Gratings

Diffraction gratings are surfaces that have been ruled into a highly regular array of grooves. The rulings scatter light in all directions. The periodicity of the ruling causes there to be coherence in the light scattered in particular discrete directions. These directions correspond to various “diffraction orders,” which are functions of the ruling direction, the ruling width, and the wavelength of the incident beam (ref. Hecht). The amount of power scattered into each order is a more subtle matter, depending on the the detailed shape of the rulings. MACOS provides grating elements that model the beam diffracted into any specified order, but does not predict the intensity of the diffracted beams.

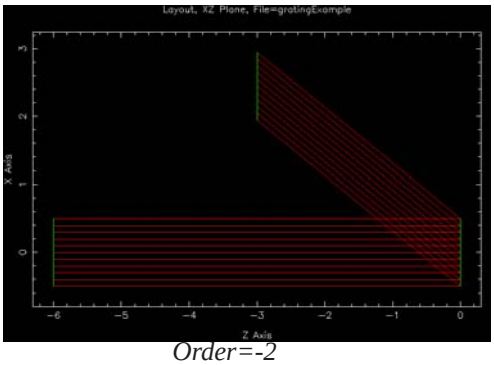
MACOS gratings can be defined on any surface shape. Enter surface parameters to define the underlying element, just as you would to define a regular mirror. Three additional element-specific parameters are required to define the grating geometry. These are:

- `OrderHOE`: The diffraction order you wish to trace off of the grating surface.
- `h1HOE`: A unit 3-vector perpendicular to the ruling direction and the `psiElt` vector. This vector is rotated and translated appropriately if the element is perturbed using the MACOS `Perturb` command.
- `RulingWidth`: The fixed distance between rules as projected to a flat plane underlying the surface. Distance between rules *along the curved surface* can vary if the surface shape is curved, which will be the case with a conic or aspheric surface type.

The following example shows the `new` dialog used to specify a grating on a flat surface. The complete prescription, with a collimated source illuminating the grating at normal incidence, is provided in Appendix A.2. Figure 19 shows the diffracted beam at a couple of orders.



Order=1



Order=-2

FIGURE 19 Diffraction grating showing diffraction orders 1 and -2.

```
Enter Element   1 Data:
 Enter element name (no blanks)  [Elt]: Grating
 +-----------------------------------------+
 | Element types:                          |
 |   Reflector      NSReflector    Segment |
 |   Refractor      NSRefractor    LensArray|
 |   FocalPlane     Reference      Return   |
 |   HOE            Grating        Obscuring|
 +-----------------------------------------+
 Enter element type :Grating
 +-----------------------------------------+
 | Surface types:                          |
 |   Flat           Conic          Aspheric |
 |   Anamorphic     Zernike        Monomial |
 |   Interpolated   UserDefined             |
 +-----------------------------------------+
 Enter surface type :Flat
 Enter element vertex location (x,y,z)  [no default]: 0,0,0
 Enter element rotation point (x,y,z)  [0.,0.,0.]:0,0,0
 Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,-1
 Do you want an aperture on this element? [NO]: no
 Enter number of obscurations  [0]:0
 Enter Fresnel propagation distance  [1.000000E+22]:1d22
+-----------------------------------------+
 | Propagation types:                      |
 |   Geometric      GeomUpdate     FarField |
 |   NFSpherical    NFS1surf                |
 |   NFPlane        NFP1surf                |
 |   NF1            NF2                     |
 |   SpatialFilter  SF1surf                 |
 +-----------------------------------------+
 Enter propagation type  [Geometric]: Geometric
 Do you wish to use element-based coordinates?: [NO]: no
 Enter element index of refraction  [1.00000]:1
 Enter element extinction coefficient  [1.000000E+22]:1d22
 Enter diffraction order  [1.00000]:1                                  OrderHOE
 Enter spacing of grating ruling  [1.00000]:18.97366596d-7             RuleWidth
 Enter vector perp to grating ruling (x,y,z)  [no default]: 0,1,0        h1HOE
```

This dialog produced the following prescription data:

```
   iElt=    1
 EltName= Grating
 Element= Grating
 Surface= Flat
   KrElt=  -1.000000000D+22
   KcElt=   0.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  RptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  IndRef=   1.000000000D+00
  Extinc=   1.000000000D+22
   h1HOE=   0.000000000D+00  1.000000000D+00  0.000000000D+00
OrderHOE=   1.000000000D+00
RuleWidth=   1.897366596D-06
    nObs=    0
  ApType=   None
    zElt=   1.000000000D+22
PropType=   Geometric
 nECoord=   -6
```

4.4.6 Transmissive Gratings

The transmissive grating elements, like the reflective gratings of the previous section, use surfaces that have been ruled into a highly regular array of grooves to diffract light. The periodicity of the ruling causes there to be coherence in the light scattered in particular discrete directions. These directions correspond to various "diffraction orders," which are functions of the ruling direction, the ruling width, and the wavelength of the incident beam (ref. Hecht).

MACOS grating elements model the beam diffracted into any specified order, but do not predict the intensity of the diffracted beams.

MACOS gratings can be defined on any surface shape. Enter surface parameters to define the underlying element, just as you would to define a regular mirror. Three additional element-specific parameters are required to define the grating geometry. These are:

- `OrderHOE`: The diffraction order you wish to trace off of the grating surface.
- `h1HOE`: A unit 3-vector perpendicular to the ruling direction and the `psiElt` vector. This vector is rotated and translated appropriately if the element is perturbed using the MACOS `Perturb` command.
- `RulingWidth`: The fixed distance between rules as projected to a flat plane underlying the surface. Distance between rules *along the curved surface* can vary if the surface shape is curved, which will be the case with a conic or aspheric surface type.

4.4.7 Reflective Holographic Optical Elements (HOEs)

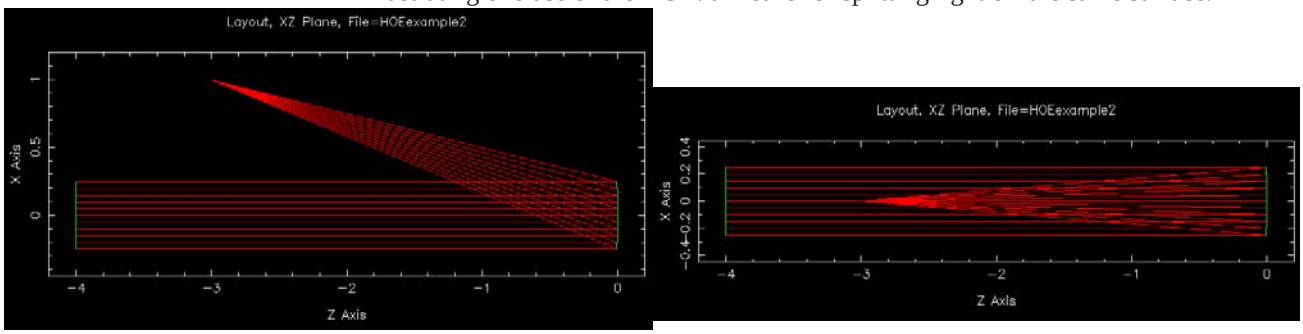
Holgraphic optical elements (HOEs) are diffractive elements that can be used to produce a more complex output beam than is possible with straight-ruled gratings. They are typically formed by interfering two coherent beams on the surface of the underlying optic. The surface is covered with photo-sensitive material, so the interference is recorded as areas of exposed and unexposed fringes. The piece is then etched to turn the interference photograph into permanent features on the surface of the optic. Subsequent illumination by one of the 2 reference beams produces a diffracted beam in the form and direction of the second beam. The result is a diffraction grating. MACOS currently supports reflective HOEs only.

MACOS HOEs can be defined on any surface shape. Enter surface parameters to define the underlying element, just as you would to define the shape of a regular mirror. The `HOE` grating is then defined using additional element-specific parameters. These parameters set up the original HOE interference geometry, so the grating on any point of the surface can be computed from the intended reference beams. The reference beams are rotated and translated appropriately if the element is perturbed using the MACOS `Perturb` command. The parameters are:

- `h1HOE`: The first is the input reference beam point source location in x,y,z. A collimated reference beam is located at infinity (1d22). Note that the reference beams have spherical (or flat) wavefronts only.
- `h2HOE`: The second is the ouput reference beam point source location in x,y,z.
- `OrderHOE`: The third is the diffraction order to be used to define the grating and to be ray-traced through the rest of the optical system. MACOS currently does not allow you to trace a different order than was used to define the grating.

- `WaveHOE`: The last is the construction or reference wavelength. The beam is traced using the source wavelength (`Wavelen`) which can be different.

The following example shows the `new` dialog used to specify a HOE on a parabolic surface. The complete prescription, with a collimated source illuminating the HOE at normal incidence, is provided in Appendix A.2. Figure 20 shows the output beam for both the diffracted beam and the reflected beam. Both will be present in an actual system, illustrating one use of the HOE: a means for splitting light off the same surface.



Diffracted beam *Reflected beam*

FIGURE 20 Holographic optical element (HOE), showing diffracted and reflected beams.

```
Enter Element   1 Data:
Enter element name (no blanks)  [Elt]: HOEonMirror
+-----------------------------------------+
| Element types:                          |
|   Reflector      NSReflector    Segment |
|   Refractor      NSRefractor    LensArray|
|   FocalPlane     Reference      Return   |
|   HOE            Grating        Obscuring|
+-----------------------------------------+
Enter element type :HOE
+-----------------------------------------+
| Surface types:                          |
|   Flat           Conic          Aspheric |
|   Anamorphic     Zernike        Monomial |
|   Interpolated   UserDefined             |
+-----------------------------------------+
Enter surface type :Conic
Enter element vertex location (x,y,z)  [no default]: 0,0,0
Enter element rotation point (x,y,z)  [0.,0.,0.]:
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:
Enter e&f or Kc&Kr (1=e&f,2=Kc&Kr)?  [1]:1
Enter element focal length  [1.000000E+22]:3
Enter element eccentricity  [0.]:1
Do you want an aperture on this element? [NO]:
Enter number of obscurations  [0]:
Enter Fresnel propagation distance  [3.00000]:1d22
+-----------------------------------------+
| Propagation types:                      |
|   Geometric      GeomUpdate     FarField |
|   NFSpherical    NFS1surf                |
|   NFPlane        NFP1surf                |
|   NF1            NF2                     |
|   SpatialFilter  SF1surf                 |
+-----------------------------------------+
Enter propagation type  [Geometric]: Geometric
Do you wish to use element-based coordinates?: [NO]: no
Enter element index of refraction  [1.00000]:1
Enter element extinction coefficient  [1.000000E+22]:1d22
```

```
Enter input reference point (x,y,z)  [no default]: 0,0,-1d22          h1HOE
Enter output reference point (x,y,z)  [no default]: 1,0,-3           h2HOE
Enter diffraction order  [1.00000]: 1                              OrderHOE
Enter reference wavelength  [1.000000E-07]:6e-7                    WaveHOE
```

This dialog produced the following prescription data:

```
   iElt=    1
 EltName= HOEonMirror
 Element= HOE
 Surface= Conic
   KrElt=  -6.000000000D+00
   KcElt=  -1.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  RptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  IndRef=   1.000000000D+00
  Extinc=   1.000000000D+22
   h1HOE=   0.000000000D+00  0.000000000D+00 -1.000000000D+22
   h2HOE=   1.000000000D+00  0.000000000D+00 -3.000000000D+00
OrderHOE=   1.000000000D+00
 WaveHOE=   6.000000000D-07
    nObs=    0
  ApType=   None
    zElt=   1.000000000D+22
PropType=   Geometric
 nECoord=   -6
```

4.4.8 Reference Surfaces or Dummy Elements

Reference surfaces are “dummy” elements, with no direct effect on the light. They are used in MACOS to define the start and stop points of diffraction propagations. They are also used to provide a screen on which to project the beam for viewing using ray-trace or diffraction commands. They have many other uses as well.

As discussed in Section , reference surfaces can be aligned with the nominally spherical (or flat) wavefront of a particular beam. When done correctly, the reference surface then defines the nominal reference wavefront, or the wavefront the beam would have if it were perfect. The OPD of the actual beam wavefront on the surface (Section 5.2.5) then represents the deviation from perfection of the beam wavefront. These differences drive the MACOS diffraction propagators. This approach greatly reduces the sampling density required to avoid undersampling a wavefront for diffraction (ref. Siegman and Szicklas).

MACOS provides functions that help assure that reference surfaces used for diffraction calculations are correctly set up. These are:

- ORS: Optimizes a reference surface to provide best sampling of a beam (Section 6.3.1).
- SRS: Slaves one reference surface to another to properly set up a near-field diffraction propagation (Section 6.3.2).

Figure 21 illustrates a typical scenario, with a reference surface inserted in the focused beam of a lens. The reference surface has its center located at the focus of the lens, and so is aligned with the ideal (spherical) wavefront. The Cassegrain example (see Appendix A.1) provides another example; indeed all of the examples in the Appendix use reference surfaces.

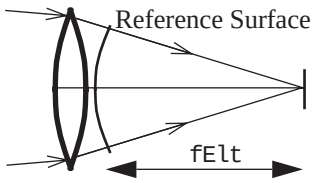


FIGURE 21 Reference surface correctly aligned with wavefront.

```
Enter element name (no blanks)  [Elt]: ref_surf                          EltName
+-----------------------------------------+
| Element types:                          |
|   Reflector      NSReflector    Segment |
|   Refractor      NSRefractor    LensArray|
|   FocalPlane     Reference      Return   |
|   HOE            Grating        Obscuring|
+-----------------------------------------+
Enter element type :Reference                                           Element
+-----------------------------------------+
| Surface types:                          |
|   Flat           Conic          Aspheric |
|   Anamorphic     Zernike        Monomial |
|   Interpolated   UserDefined             |
+-----------------------------------------+
Enter surface type :Conic                                               Surface
Enter element vertex location (x,y,z)  [no default]: 0,0,0               VptElt
Enter element rotation point (x,y,z)  [0.,0.,0.]:0,0,0                   RptElt
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,1             psiElt
Enter e&f or Kc&Kr (1=e&f,2=Kc&Kr)?  [1]:1
Enter element focal length  [1.000000E+22]:5.4                             fElt
Enter element eccentricity  [0.]:0                                         eElt
Do you want an aperture on this element? [NO]: no
Enter number of obscurations  [0]:0
Enter Fresnel propagation distance  [5.40000]:5.4                          zElt
+-----------------------------------------+
| Propagation types:                      |
|   Geometric      GeomUpdate     FarField |
|   NFSpherical    NFS1surf                |
|   NFPlane        NFP1surf                |
|   NF1            NF2                     |
|   SpatialFilter  SF1surf                 |
+-----------------------------------------+
Enter propagation type  [Geometric]: Geometric                          PropType
Do you wish to use element-based coordinates?: [NO]: no
```

This dialog produced the following prescription data:

```
   iElt=    3
 EltName= ref_surf
 Element= Reference
 Surface= Conic
   KrElt=  -5.4000000000D+00
   KcElt=   0.0000000000D+00
  psiElt=   0.0000000000D+00  0.0000000000D+00  1.0000000000D+00
  VptElt=   0.0000000000D+00  0.0000000000D+00  0.0000000000D+00
  RptElt=   0.0000000000D+00  0.0000000000D+00  0.0000000000D+00
  IndRef=   1.0000000000D+00
  Extinc=   0.0000000000D+00
    nObs=    0
  ApType=   None
    zElt=   5.4000000000D+00
PropType=   Geometric
 nECoord=   -6
```

4.4.9 Focal Planes

The focal plane is a reference element, but one that always has a flat surface. The only difference between MACOS `FocalPlane` and `Reference` elements is in the way that linear models are computed for them. Use focal planes to end a prescription that is to be used for linear models! As a a reference element, a focal plane element has no direct optical effect. Focal planes are commonly used to terminate far-field diffraction propagations.

```
Enter element name (no blanks)  [Elt]: focal_plane                    EltName
+-----------------------------------------+
| Element types:                          |
|   Reflector      NSReflector    Segment |
|   Refractor      NSRefractor    LensArray|
|   FocalPlane     Reference      Return   |
|   HOE            Grating        Obscuring|
+-----------------------------------------+
Enter element type :FocalPlane                                        Element
+-----------------------------------------+
| Surface types:                          |
|   Flat           Conic          Aspheric |
|   Anamorphic     Zernike        Monomial |
|   Interpolated   UserDefined            |
+-----------------------------------------+
Enter surface type :Flat                                              Surface
Enter element vertex location (x,y,z)  [no default]: 0,0,2             VptElt
Enter element rotation point (x,y,z)  [0.,0.,2.00000]:0,0,2            RptElt
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,1           psiElt
Do you want an aperture on this element? [NO]: no
Enter number of obscurations  [0]:0
Enter Fresnel propagation distance  [1.000000E+22]:1d22                  zElt
+-----------------------------------------+
| Propagation types:                      |
|   Geometric      GeomUpdate     FarField |
|   NFSpherical    NFS1surf                |
|   NFPlane        NFP1surf                |
|   NF1            NF2                     |
|   SpatialFilter  SF1surf                 |
+-----------------------------------------+
Enter propagation type  [Geometric]: Geometric                        PropType
```

This dialog produced the following prescription data:

```
   iElt=    5
 EltName= focal_plane
 Element= FocalPlane
 Surface= Flat
   KrElt=  -1.000000000D+22
   KcElt=   0.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00  1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00  2.000000000D+00
  RptElt=   0.000000000D+00  0.000000000D+00  2.000000000D+00
  IndRef=   1.000000000D+00
  Extinc=   0.000000000D+00
    nObs=    0
  ApType=   None
    zElt=   1.000000000D+22
PropType=   Geometric
 nECoord=   -6
```

4.4.10 Return Surfaces

The return surface is another element which is used for calculations. It has no effect on wavefront propagation. Specifically, two return surfaces in sequence are used to find the position of the exit pupil of the system in the FEX command. This discussed more thoroughly in Section 5.3.

```
Enter element name (no blanks)  [Elt]: return                          EltName
+-----------------------------------------+
| Element types:                          |
|   Reflector      NSReflector    Segment |
|   Refractor      NSRefractor    LensArray|
|   FocalPlane     Reference      Return   |
|   HOE            Grating        Obscuring|
+-----------------------------------------+
Enter element type :Return                                             Element
+-----------------------------------------+
| Surface types:                          |
|   Flat           Conic          Aspheric |
|   Anamorphic     Zernike        Monomial |
|   Interpolated   UserDefined             |
+-----------------------------------------+
Enter surface type :Conic                                              Surface
Enter element vertex location (x,y,z)  [no default]: 0,0,1              VptElt
Enter element rotation point (x,y,z)  [0.,0.,1.00000]:0,0,1             RptElt
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,1            psiElt
Enter e&f or Kc&Kr (1=e&f,2=Kc&Kr)?  [1]:1
Enter element focal length  [1.000000E+22]:5                              fElt
Enter element eccentricity  [0.]:0                                        eElt
Do you want an aperture on this element? [NO]: no
Enter number of obscurations  [0]:0
Enter Fresnel propagation distance  [5.00000]:5                           zElt
+-----------------------------------------+
| Propagation types:                      |
|   Geometric      GeomUpdate     FarField |
|   NFSpherical    NFS1surf                |
|   NFPlane        NFP1surf                |
|   NF1            NF2                     |
|   SpatialFilter  SF1surf                 |
+-----------------------------------------+
Enter propagation type  [Geometric]: Geometric                         PropType
Do you wish to use element-based coordinates?: [NO]: no
```

This dialog produced the following prescription data:

```
   iElt=    1
 EltName= return
 Element= Return
 Surface= Conic
   KrElt=  -5.000000000D+00
   KcElt=   0.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00  1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00  1.000000000D+00
  RptElt=   0.000000000D+00  0.000000000D+00  1.000000000D+00
  IndRef=   1.000000000D+00
  Extinc=   0.000000000D+00
    nObs=    0
  ApType=   None
    zElt=   5.000000000D+00
PropType=   Geometric
 nECoord=   -6
```

4.4.11 Hex and Pie Segmented Mirrors

Segmented systems (e.g. telescopes with segmented primary mirrors) are defined at the light source using `GridType=hex` or `GridType=pie`. This section describes entering the segmented surfaces. Figures 22 and 23 show the geometry for a segmented pie-shaped aperture and a large array of hexagonal segments, respectively. In both cases, the pannel coordinates are given by three variables (X,L,R). The segments must be entered in as elements in the same order they are listed in `SegCoord`. Segmented surfaces also require

- `nSeg`: number of segments
- `gap`: spacing between segments
- `width`: width of segments (flat to flat)

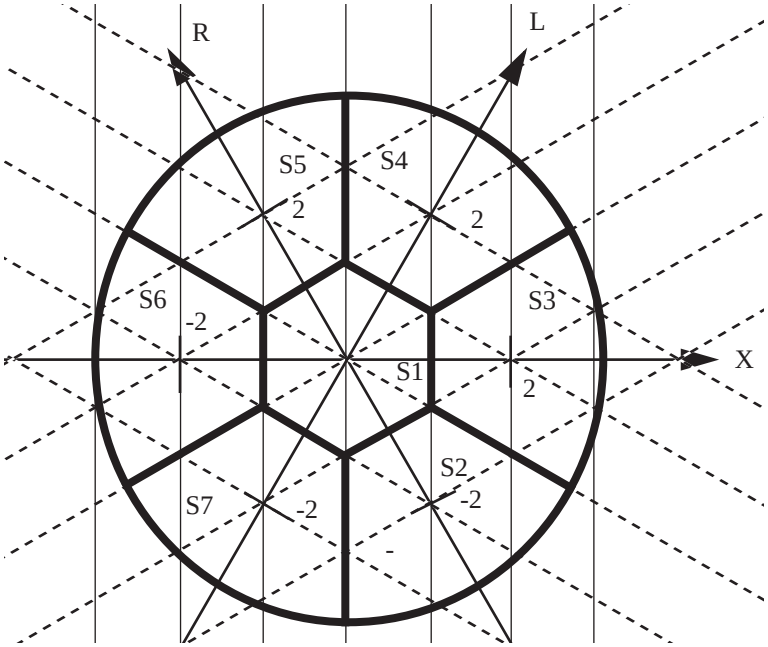


FIGURE 22 Hexagonal coordinates X,L,R for a pie-shaped segmented mirror.

More than 1 segmented mirror can be used in a single prescription. In the case of 2 segmented mirrors, the segments are required to be lined up, so that the segmentation defined at the source is valid for both beams.

An example of a segment mirror .in-file (`seg.in`) is in Appendix A.5. The panels are numbered in the same order as they are in the Figure 23.

Input data:

```
+-----------------------------------------+
| Source ray grid types:                  |
|   Circular       Square         Hex     |
|   Pie            Flower                 |
+-----------------------------------------+
Enter source grid type  [Circular]: Pie                    GridType
Enter aperture diameter or angle  [1.00000]:3.65           Aperture
Enter number of grid points  [128]:31                      nGridPts
```

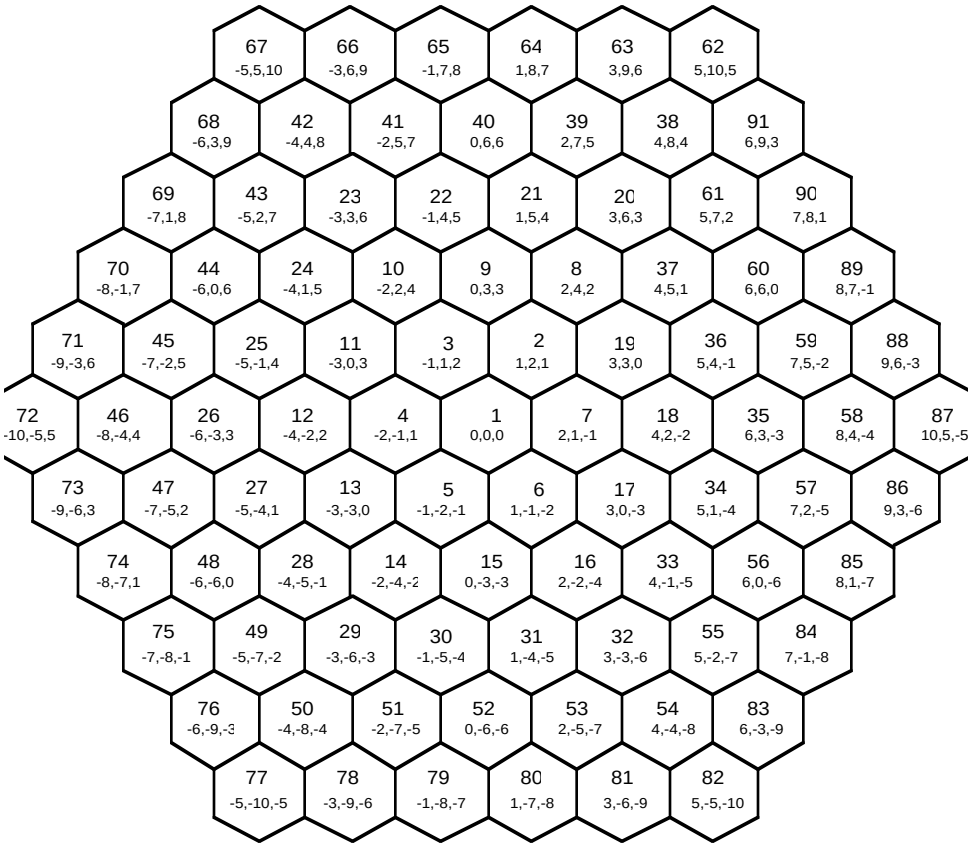


FIGURE 23 Hexagonal coordinates X,L,R for a large array

```
Enter input plane x-axis vector (x,y,z)  [1.00000,0.,0.]:1,0,0          xGrid
Enter input plane y-axis vector (x,y,z)  [0.,1.00000,0.]:0,1,0          yGrid
Enter number of segments  [no default]: 7                               nSeg
Enter segment width: [3.65000]:1.2                                      width
Enter gap between segments: [0.]:0                                      gap
Enter 3 segment 1 hex coords  [no default]: 0,0,0                       SegCoord
Enter 3 segment 2 hex coords  [no default]: 1,-1,-2
Enter 3 segment 3 hex coords  [no default]: 2,1,-1
Enter 3 segment 4 hex coords  [no default]: 1,2,1
Enter 3 segment 5 hex coords  [no default]: -1,1,2
Enter 3 segment 6 hex coords  [no default]: -2,-1,1
Enter 3 segment 7 hex coords  [no default]: -1,-2,-1
Enter segment plane x-axis vector (x,y,z)  [1.00000,0.,0.]:1,0,0        SegXgrid
```

Element data:

```
Enter Element 1 Data:
Enter element name (no blanks): (Elt): s1                               EltName
+-----------------------------------------+
| Element types:                          |
|   Reflector      NSReflector    Segment |
|   Refractor      NSRefractor    LensArray|
|   FocalPlane     Reference      Return   |
|   HOE            Grating        Obscuring|
+-----------------------------------------+
Enter element type :Segment                                             Element
+-----------------------------------------+
| Surface types:                          |
```

```
|   Flat           Conic          Aspheric   |
|   Anamorphic     Zernike        Monomial   |
|   Interpolated   UserDefined                |
+---------------------------------------------+
Enter surface type :Conic                                          Surface
Enter element vertex location (x,y,z)  [no default]: 0,0,0          VptElt
Enter element rotation point (x,y,z)  [0.,0.,0.]:0,0,0              RptElt
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,-1       psiElt
Enter e&f or Kc&Kr (1=e&f,2=Kc&Kr)?  [1]:1
Enter element focal length  [1.000000E+22]:1.46                      fElt
Enter element eccentricity  [0.]:1                                   eElt
Do you want an aperture on this element? [NO]:
Enter number of obscurations  [0]:
Enter Fresnel propagation distance  [1.46000]:1.46                   zElt
+---------------------------------------------+
| Propagation types:                          |
|   Geometric      GeomUpdate     FarField    |
|   NFSpherical    NFS1surf                   |
|   NFPlane        NFP1surf                   |
|   NF1            NF2                        |
|   SpatialFilter  SF1surf                    |
+---------------------------------------------+
Enter propagation type  [Geometric]: Geometric                     PropType
Do you wish to use element-based coordinates?: [NO]:
Enter element index of refraction  [1.00000]:
Enter element extinction coefficient  [1.000000E+22]:

    iElt=    1
 EltName= s1
 Element= Segment
 Surface= Conic
    fElt=   1.460000000D+00
    eElt=   1.000000000D+00
   KrElt=  -2.920000000D+00
   KcElt=  -1.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  RptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  IndRef=   1.000000000D+00
  Extinc=   1.000000000D+22
    nObs=    0
  ApType=   None
    zElt=   1.460000000D+00
PropType=   Geometric
 nECoord=   -6
 Is this correct? [YES]:
 Enter Element   2 Data:
 Enter element name (no blanks)  [Elt]: s2                         EltName
 +-----------------------------------------+
 | Element types:                          |
 |   Reflector      NSReflector    Segment |
 |   Refractor      NSRefractor    LensArray|
 |   FocalPlane     Reference      Return   |
 |   HOE            Grating        Obscuring|
 +-----------------------------------------+
 Enter element type :Segment                                       Element
 +-----------------------------------------+
 | Surface types:                          |
 |   Flat           Conic          Aspheric |
```

```
 |   Anamorphic     Zernike        Monomial   |
 |   Interpolated   UserDefined                |
 +---------------------------------------------+
 Enter surface type :Conic                                                    Surface
 Enter element rotation point (x,y,z)  [0.,0.,0.]:.6124997330,-1.060881274,
.2569563356
 Do you want an aperture on this element? [NO]:
 Enter number of obscurations  [0]:
 Enter Fresnel propagation distance  [1.46000]:1.46                            zElt
 +---------------------------------------------+
 | Propagation types:                          |
 |   Geometric      GeomUpdate     FarField    |
 |   NFSpherical    NFS1surf                   |
 |   NFPlane        NFP1surf                   |
 |   NF1            NF2                        |
 |   SpatialFilter  SF1surf                    |
 +---------------------------------------------+
 Enter propagation type  [Geometric]: Geometric                            PropType
 Do you wish to use element-based coordinates?: [NO]:
 Enter element index of refraction  [1.00000]:
 Enter element extinction coefficient  [1.000000E+22]:

    iElt=    2
 EltName= s2
 Element= Segment
 Surface= Conic
    fElt=   1.460000000D+00
    eElt=   1.000000000D+00
   KrElt=  -2.920000000D+00
   KcElt=  -1.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  RptElt=   6.124997330D-01 -1.060881274D+00  2.569563356D-01
  IndRef=   1.000000000D+00
  Extinc=   1.000000000D+22
    nObs=    0
  ApType=   None
    zElt=   1.460000000D+00
PropType=   Geometric
 nECoord=   -6
 Is this correct? [YES]:

...
Enter Element   7 Data:
 Enter element name (no blanks)  [Elt]: s7                                   EltName
 +---------------------------------------------+
 | Element types:                              |
 |   Reflector      NSReflector    Segment     |
 |   Refractor      NSRefractor    LensArray   |
 |   FocalPlane     Reference      Return      |
 |   HOE            Grating        Obscuring   |
 +---------------------------------------------+
 Enter element type :Segment                                                  Element
 +---------------------------------------------+
 | Surface types:                              |
 |   Flat           Conic          Aspheric    |
 |   Anamorphic     Zernike        Monomial    |
 |   Interpolated   UserDefined                |
 +---------------------------------------------+
 Enter surface type :Conic                                                    Surface
 Enter element rotation point (x,y,z)  [0.,0.,0.]:-.612500534,-1.060880811,
.2569563356
 Do you want an aperture on this element? [NO]:
 Enter number of obscurations  [0]:
 Enter Fresnel propagation distance  [1.46000]:1.46                            zElt
```

```
+-----------------------------------------+
| Propagation types:                      |
|   Geometric      GeomUpdate     FarField   |
|   NFSpherical    NFS1surf                 |
|   NFPlane        NFP1surf                 |
|   NF1            NF2                      |
|   SpatialFilter  SF1surf                  |
+-----------------------------------------+
Enter propagation type  [Geometric]: Geometric                    PropType
Do you wish to use element-based coordinates?: [NO]:
Enter element index of refraction  [1.00000]:
Enter element extinction coefficient  [1.000000E+22]:

   iElt=    7
 EltName= s7
 Element= Segment
 Surface= Conic
    fElt=   1.460000000D+00
    eElt=   1.000000000D+00
   KrElt=  -2.920000000D+00
   KcElt=  -1.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  RptElt=  -6.125005340D-01 -1.060880811D+00  2.569563356D-01
  IndRef=   1.000000000D+00
  Extinc=   1.000000000D+22
    nObs=    0
  ApType=   None
    zElt=   1.460000000D+00
PropType=   Geometric
 nECoord=   -6
 Is this correct? [YES]:
```

4.4.12 Flower Segmented Mirrors

Flower segmented mirrors consist of a central segment surrounded by “petal” segments. As such, they are like the “pie” segmented mirrors of the previous section, except flower segmentation is not limited to a hexagonal geometry. Flower segmentation is defined at the light source using `GridType=Flower`. Two other parameters are used to define the segmentation:

- `nPetals`: number of petals surrounding the central segment
- `radCtr`: radius of the largest inscribed circle defining the size of the center segment

Only 1 segment coordinate is needed per segment, to specify its location in sequence around the central segment, with the 0th segment being the center segment, the 1st segment having one edge along the SegXgrid axis and the other along a line rotated 360°/

nPetals towards the SegYgrid axis, and the 2nd and subsequent segments located in the same sequence around the center segment. See Fig. 24 for an illustration.

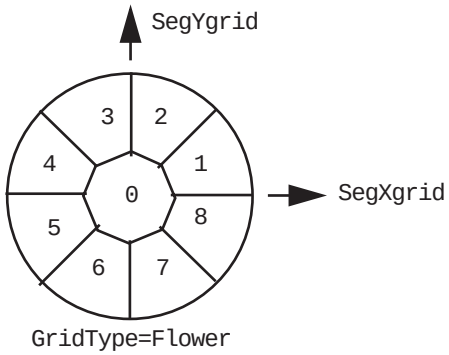


FIGURE 24 Flower segmentation, showing segment coordinates

4.4.13 Reflective Non-Sequential Surfaces

In most optical systems, all rays strike one element before passing on to the next element. There are exceptions, such as corner cube reflectors, where some rays will strike an element before others. These are modeled in MACOS using non-sequential surfaces. There is no limit on the number of non-sequential elements and the order that the non-sequential elements are entered into MACOS does not matter. However, there must be sequential surfaces before and after the group of non-sequential surfaces. For the purpose of the following discussion, we call them the first and second surfaces, respectively. After the ray passes the first sequential surface, MACOS solves for the next surface that the ray strikes which can be any of the group of non-sequential surfaces or the second sequential surface. If the ray hits a non-sequential surface, the procedure continues until the second sequential surface is encountered. After the second sequential surface is encountered the ray trace proceeds normally. Figure 25 shows the corner cube reflector example. The .in-file is in Appendix A.3

NOTE: Geometric propagation only is supported through non-squential elements.

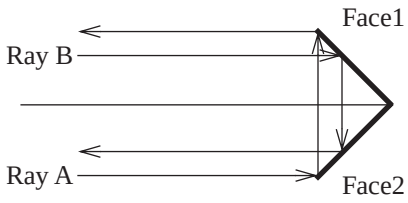


FIGURE 25 Non-sequential surface: Ray A has Face2 as first element, Ray B's first element is Face1.

Corner cube example

```
Enter element name (no blanks)  [Elt]: Face1                    EltName
+-----------------------------------------+
| Element types:                          |
|   Reflector      NSReflector    Segment  |
|   Refractor      NSRefractor    LensArray|
|   FocalPlane     Reference      Return   |
```

```
|   HOE           Grating        Obscuring  |
+-------------------------------------------+
Enter element type :NSReflector                                    Element
+-------------------------------------------+
| Surface types:                            |
|   Flat          Conic          Aspheric   |
|   Anamorphic    Zernike        Monomial   |
|   Interpolated  UserDefined               |
+-------------------------------------------+
Enter surface type :Flat                                           Surface
Enter element vertex location (x,y,z) [no default]: 0,0,0           VptElt
Enter element rotation point (x,y,z) [0.,0.,0.]:0,0,0               RptElt
Enter element principal axis (x,y,z) [0.,0.,-1.00000]:-.7071,0,-.7071   psiElt
Do you want an aperture on this element? [NO]:
Enter number of obscurations  [0]:
Enter Fresnel propagation distance  [1.000000E+22]:1d22              zElt
+-------------------------------------------+
| Propagation types:                        |
|   Geometric     GeomUpdate     FarField   |
|   NFSpherical   NFS1surf                  |
|   NFPlane       NFP1surf                  |
|   NF1           NF2                       |
|   SpatialFilter SF1surf                   |
+-------------------------------------------+
Enter propagation type  [Geometric]: Geometric                     PropType
Do you wish to use element-based coordinates?: [NO]:
Enter element index of refraction  [1.00000]:1d0
Enter element extinction coefficient  [1.000000E+22]:



Enter element name (no blanks)  [Elt]: Face2                       EltName
+-------------------------------------------+
| Element types:                            |
|   Reflector     NSReflector    Segment    |
|   Refractor     NSRefractor    LensArray  |
|   FocalPlane    Reference      Return     |
|   HOE           Grating        Obscuring  |
+-------------------------------------------+
Enter element type :NSReflector                                    Element
+-------------------------------------------+
| Surface types:                            |
|   Flat          Conic          Aspheric   |
|   Anamorphic    Zernike        Monomial   |
|   Interpolated  UserDefined               |
+-------------------------------------------+
Enter surface type :Flat                                           Surface
Enter element vertex location (x,y,z) [no default]: 0,0,0           VptElt
Enter element rotation point (x,y,z) [0.,0.,0.]:0,0,0               RptElt
Enter element principal axis (x,y,z) [0.,0.,-1.00000]:.7071,0,-.7071   psiElt
Do you want an aperture on this element? [NO]:
Enter number of obscurations  [0]:
Enter Fresnel propagation distance  [1.000000E+22]:1d22              zElt
+-------------------------------------------+
| Propagation types:                        |
|   Geometric     GeomUpdate     FarField   |
|   NFSpherical   NFS1surf                  |
|   NFPlane       NFP1surf                  |
```

```
|   NF1            NF2                      |
|   SpatialFilter  SF1surf                  |
+-------------------------------------------+
Enter propagation type  [Geometric]: Geometric                    PropType
Do you wish to use element-based coordinates?: [NO]:
Enter element index of refraction  [1.00000]:1d0
Enter element extinction coefficient  [1.000000E+22]:
```

4.4.14 Refractive Non-Sequential Surfaces

Refractive non-sequential surfaces are very similiar to reflective non-sequential surface (see Section 4.4.13). An example illustrating their application is a Luneberg Lens Antenna is provided in Appendix A.3. The Luneberg Lens is a stepped-gradient index lens consisting of concentric spherical shells, which grow denser as they decrease in radius. A point source on the surface produces rays that follow nearly elliptical trajectories, emerging on the other side of the lens in a collimated beam. Using multiple point sources distributed on the surface, the Luneberg Lens can provide instantaneous, solid-state beam coverage over nearly 4π steradians.

4.5 Element Surface Types

4.5.1 Flat Surfaces

Flat surfaces are completely defined by a point (the "vertex" `VptElt`) on the surface, and the unit vector normal to the surface at that point (`psiElt`). Technically, the *vertex* of a plane or sphere is undefined; the user is free to pick any point on the surface as the vertex. In MACOS, a flat surface is a spherical surface with an infinite radius of curvature, `KrElt=1d22`. See Section 4.4.1 for an example of entering flat surface data.

4.5.2 Conic Surfaces

Conic surfaces – really conicoids of revolution – include the most common optical shapes: spheres, ellipsoids, paraboloids, hyperboloids. Flat surfaces are spheroids with an infinite radius of curvature. MACOS specifies the shape of a conic surface using two surface-specific parameters: the conic constant `KcElt` and the "radius," `KrElt`. Table 7 lists the common names associated with special values of the conic constant.

TABLE 7. Conic coefficients

Conic coefficient	Conic section
`KcElt<-1`	hyperboloid
`KcElt=-1`	paraboloid
`-1<KcElt<0`	ellipsoid with major axis on the principal axis (also known as prolate spheroid), `psiElt`
`KcElt=0`	sphere
`KcElt>0`	oblate ellipsoid

MACOS also accepts input of conic parameters in terms of the geometric focal length (which is *not* the same as optical focal length of a spherical mirror) and eccentricity which are denoted `fElt` and `eElt`, respectively. Eccentricity and focal length are related to the conic constant and radius by:

$$eElt = \sqrt{-KcElt} \qquad (4.1)$$

$$fElt = \frac{-KrElt}{(1 + \sqrt{-KcElt})} \quad \textbf{(4.2)}$$

The “sag equation” for a conic surface, which is to say, the equation which defines all points on the surface, is:

$$Z(h) = \frac{h^2}{[KrElt + \sqrt{(KrElt)^2 - (1 + KcElt)h^2})]} \quad \textbf{(4.3)}$$

Here h is the radial distance to the projection of the point of incidence onto a plane perpendicular to the principal axis and including the surface vertex; and Z is the height of the surface above that plane (Figure 26).

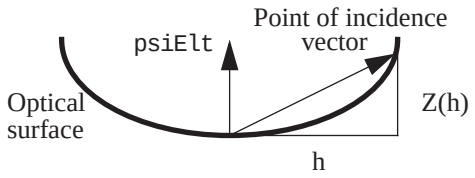


FIGURE 26 Conic mirror parameters

Note that the *vertex* of a conic refers to the geometric vertex except in the case of an oblate ellipsoid. For an oblate ellipsoid (`KcElt>0`), the “vertex” is taken as the point where the minor axis intersects the surface.

See Section 4.4.2 for an example of entering conic surface data.

4.5.3 Generalized (10th-Order) Aspheric Surfaces

Generalized aspheric surfaces use a conic surface as a base, and then impose a deviation on it, defined by additional, higher order, symmetric terms in radius h. In MACOS, generalized aspheric surfaces take the form

$$Z(h) = \frac{h^2}{[KrElt + \sqrt{(KrElt)^2 - (1 + KcElt)h^2})]} + Ah^4 + Bh^6 + Ch^8 + Dh^{10} \quad \textbf{(4.1)}$$

where h is the radial pupil coordinate (the magnitude projection of the of the point of incidence vector onto the plane perpendicular to `psiElt` as shown in Figure 29), `KrElt` is the radius of curvature of the element, `KcElt` is the conic constant, and A, B, C, D are the aspheric coefficients, `AsphCoef(i)`, with `AsphCoef(1)` = A, `AsphCoef(2)` = B, `AsphCoef(3)` = C, and `AsphCoef(4)` = D.

Note: The A,B,C,D parameters are used by optical design codes as well, except that the signs of the coefficients may differ in some cases. The sign of the coefficients in some codes is a function of the direction of the incident ray. MACOS references surfaces to a global coordinate system and does not assume a particular direction of the beam in defining the optical train. In MACOS the coefficients are positive in the direction of the element psiElt vector.

This example shows how to enter a higher order aspheric reflector.

```
Enter element name (no blanks)  [Elt]: gen_asphere                    EltType
```

```
+-----------------------------------------+
| Element types:                          |
|   Reflector      NSReflector    Segment |
|   Refractor      NSRefractor    LensArray|
|   FocalPlane     Reference      Return   |
|   HOE            Grating        Obscuring|
+-----------------------------------------+
Enter element type :Reflector                                          Element
+-----------------------------------------+
| Surface types:                          |
|   Flat           Conic          Aspheric |
|   Anamorphic     Zernike        Monomial |
|   Interpolated   UserDefined             |
+-----------------------------------------+
Enter surface type :Aspheric                                           Surface
Enter element vertex location (x,y,z)  [no default]: 0,0,0              VptElt
Enter element rotation point (x,y,z)  [0.,0.,0.]:0,0,0                  RptElt
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,1            psiElt
Enter e&f or Kc&Kr (1=e&f,2=Kc&Kr)?  [1]:2
Enter Conic Constant  [0.]:-1                                           KcElt
Enter Radius  [-1.000000E+22]:5                                         KrElt
Do you want an aperture on this element? [NO]:
Enter number of obscurations  [0]:
Enter Fresnel propagation distance  [2.50000]:5                         zElt
+-----------------------------------------+
| Propagation types:                      |
|   Geometric      GeomUpdate     FarField |
|   NFSpherical    NFS1surf                |
|   NFPlane        NFP1surf                |
|   NF1            NF2                     |
|   SpatialFilter  SF1surf                 |
+-----------------------------------------+
Enter propagation type  [Geometric]: Geometric                          PropType
Do you wish to use element-based coordinates?: [NO]:
Enter element index of refraction  [1.00000]:
Enter element extinction coefficient  [1.000000E+22]:
Enter aspheric coefficients A,B,C,D [0,0,0,0]:1e-9,2e-19,3e-17,4e-20    AsphCoef
```

This dialog produced the following prescription data:

```
    iElt=    1
 EltName= gen_asphere
 Element= Reflector
 Surface= Aspheric
   KrElt=  -5.000000000D+00
   KcElt=  -1.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00  1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  RptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  IndRef=   1.000000000D+00
  Extinc=   1.000000000D+22
AsphCoef=   1.000000000D-09  2.000000000D-19  3.000000000D-17  4.00000000D-20
    nObs=    0
  ApType=   None
    zElt=   5.000000000D+00
PropType=   Geometric
 nECoord=   -6
```

4.5.4 Zernike Surfaces

Zernike surfaces add a deformation—described by a Zernike polynomial—to the surface of a conicoid. The deformation is entered as the “excess sag” of the surface (*not* the wavefront error). MACOS allows up to the first 45 Zernike terms to be added to a conicoid base surface shape. The Zernike polynomial set that is used is that defined by Mal-

acara in his book, “Optical Shop Testing.” The ordering of the polynomials used in MACOS, and well as two other popular orderings, are shown in Table 8. In the next release of MACOS, the user will be able to select from these three orderings, as well as only enter the first 36 terms if so desired (for backwards compatibility with previous versions of MACOS).

TABLE 8. Zernike polynomials

MACOS (Malacara)	Code V (Noll)	FRINGE (Fringe)	Polynomial
1	1	1	1
2	3	3	R sin(A)
3	2	2	R cos(A)
4	6	6	R^2 sin(2A)
5	5	4	$2R^2$-1
6	4	5	R^2 cos(2A)
7	10	11	R^3 sin(3A)
8	9	8	($3R^3$-2R) sin(A)
9	8	7	($3R^3$-2R) cos(A)
10	7	10	R^3 cos(3A)
11	15	18	R^4sin(4A)
12	14	13	($4R^4$-$3R^2$) sin(2A)
13	13	9	$6R^4$-$6R^2$+1
14	12	12	($4R^4$-$3R^2$) cos(2A)
15	11	17	R^4 cos(4A)
16	21	27	R^5 sin(5A)
17	20	20	($5R^5$-$4R^3$) sin(3A)
18	19	15	($10R^5$-$12R^3$+3R) sin(A)
19	18	14	($10R^5$-$12R^3$+3R) cos(A)
20	17	19	($5R^5$-$4R^3$) cos(3A)
21	16	26	R^5 cos(5A)
22	28		R^6 sin(6A)
23	27	29	($6R^6$-$5R^4$) sin(4A)
24	26	22	($15R^6$-$20R^4$+$6R^2$) sin(2A)
25	25	16	$20R^6$-$30R^4$+$12R^2$+1
26	24	21	($15R^6$-$20R^4$+$6R^2$) cos(2A)
27	23	28	($6R^6$-$5R^4$) cos(4A)
28	22		R^6 cos(6A)
29	36		R^7 sin(7A)
30	35		($7R^7$-$6R^5$) sin(5A)
31	34	31	($21R^7$-$30R^5$+$10R^3$) sin(3A)
32	33	24	($35R^7$-$60R^5$+$30R^3$-4R) sin(A)
33	32	23	($35R^7$-$60R^5$+$30R^3$-4R) cos(A)

TABLE 8. Zernike polynomials

MACOS (Malacara)	Code V (Noll)	FRINGE (Fringe)	Polynomial
34	31	30	$(21R^7-30R^5+10R^3)\cos(3A)$
35	30		$(7R^7-6R^5)\cos(5A)$
36	29		$R^7\cos(7A)$
37	45		$R^8\sin(8A)$
38	44		$(8R^8-7R^6)\sin(6A)$
39	43		$(28R^8-42R^6+15R^4)\sin(4A)$
40	42	33	$(56R^8-105R^6+60R^4-10R^2)\sin(2A)$
41	41	25	$70R^8-140R^6+90R^4-20R^2+1$
42	40	32	$(56R^8-105R^6+60R^4-10R^2)\cos(2A)$
43	39		$(28R^8-42R^6+15R^4)\cos(4A)$
44	38		$(8R^8-7R^6)\cos(6A)$
45	37		$R^8\cos(8A)$

`KcElt` and `KrElt` (or `eElt` and `fElt`) are the parameters that describe the conicoid. The deviations occur only in the z-direction of a special coordinate frame. The variable `pMon` is the center of the coordinate frame. The variables `xMon`, `yMon`, and `zMon` define the special coordinate frame as shown in Figure 27.The variable `lMon` is the semi-diameter of the circle to which the Zernike deviations are scaled (remember that Zernikes are defined on a unit circle).

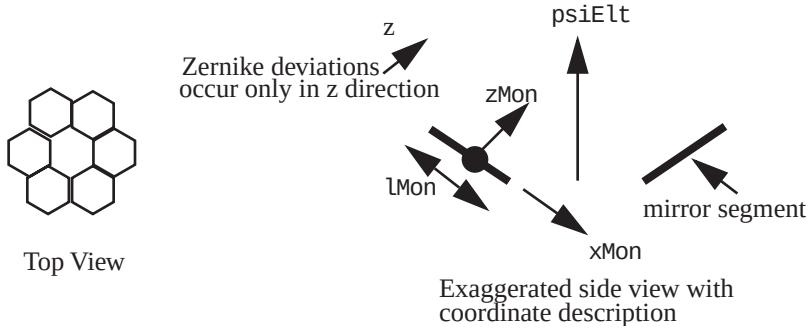


FIGURE 27 Specialized coordinate frame

Zernike polynomials, listed in Table 8, are orthogonal over a unit circle. The polar coordinates (R, A) are related to the coordinates of the projection of the point of incidence vector onto the plane perpendicular defined by `xMon` and `yMon` (X, Y) by X=R `lMon` cos(A) and Y=R `lMon` sin(A). The more common Seidel aberrations can be calculated from the Zernike coefficients (ref. Malacara).An example using a Zernike surface to simulate deformations on the primary mirror of a small telescope is provided in Appendix A.7. The following dialog illustrates the responses the `NEW` command expects in entering a Zernike reflector:

```
Enter element name (no blanks)  [Elt]: zern_surface                    EltName
+-----------------------------------------+
| Element types:                          |
|   Reflector      NSReflector    Segment    |
|   Refractor      NSRefractor    LensArray  |
|   FocalPlane     Reference      Return     |
```

```
|   HOE           Grating        Obscuring  |
+-------------------------------------------+
Enter element type :Reflector                                    Element
+-------------------------------------------+
| Surface types:                            |
|   Flat          Conic          Aspheric   |
|   Anamorphic    Zernike        Monomial   |
|   Interpolated  UserDefined               |
+-------------------------------------------+
Enter surface type :Zernike                                      Surface
Enter element vertex location (x,y,z)  [no default]: 0,0,0        VptElt
Enter element rotation point (x,y,z)  [0.,0.,0.]:0,0,0            RptElt
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,-1     psiElt
Enter e&f or Kc&Kr (1=e&f,2=Kc&Kr)?  [1]:2
Enter Conic Constant  [0.]:0                                      KcElt
Enter Radius  [-1.000000E+22]:1d22                                KrElt
Do you want an aperture on this element? [NO]:
Enter number of obscurations  [0]:
Enter Fresnel propagation distance  [1.000000E+22]:1d22           zElt
+-------------------------------------------+
| Propagation types:                        |
|   Geometric     GeomUpdate     FarField   |
|   NFSpherical   NFS1surf                  |
|   NFPlane       NFP1surf                  |
|   NF1           NF2                       |
|   SpatialFilter SF1surf                   |
+-------------------------------------------+
Enter propagation type  [Geometric]: Geometric                   PropType
Do you wish to use element-based coordinates?: [NO]:
Enter element index of refraction  [1.00000]:1
Enter element extinction coefficient  [1.000000E+22]:
Enter Zernike coefficients 1-6  [0.,0.,0.,0.,0.,0.]:.0001,0,.00002,0,0, .00001
ZernCoef
 Enter Zernike coefficients 7-12  [0.,0.,0.,0.,0.,0.]:0,0,.000001,.00002,0,0  Zern-
Coef
 Enter Zernike coefficients 13-18 [0.,0.,0.,0.,0.,0.]:0,.000003,.000001,0,0, .000002
ZernCoef
Enter Zernike coefficients 19-24 [0.,0.,0.,0.,0.,0.]:.0000005,.0000001,0,0,.000002,.00004
ZernCoef
 Enter Zernike coefficients 25-30  [0.,0.,0.,0.,0.,0.]:.0000001,0,0,0, .000001,0
ZernCoef
 Enter Zernike coefficients 31-36  [0.,0.,0.,0.,0.,0.]:0,0,0,0,0,0        ZernCoef
 Enter Zernike coefficients 37-42  [0.,0.,0.,0.,0.,0.]:0,0,0,0,0,0        ZernCoef
 Enter Zernike coefficients 43-45  [0.,0.,0.]:0,0,0                       ZernCoef
 Enter surface reference point location (x,y,z)  [0.,0.,0.]:0,0,0           pMon
 Enter surface x-axis unit vector (x,y,z)  [-1.00000,0.,0.]:1,0,0          xMon
 Enter surface y-axis unit vector (x,y,z)  [0.,-1.00000,0.]:0,1,0          yMon
 Enter surface z-axis unit vector (x,y,z)  [0.,0.,-1.00000]:0,0,1          zMon
  Orthoganalized zMon=  0.  0.    1.0000000000000
 Enter scale length  [1.00000]:1                                          lMon
```

This dialog produced the following prescription data:

```
   iElt=    1
 EltName= zern_surface
 Element= Reflector
 Surface= Zernike
   KrElt=  -1.000000000D+22
   KcElt=   0.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
```

```
  RptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  IndRef=   1.000000000D+00
  Extinc=   1.000000000D+22
ZernCoef=   1.000000000D-04  0.000000000D+00  2.000000000D-05 0.000000000D+00
0.000000000D+00  1.000000000D-05
            0.000000000D+00  0.000000000D+00  1.000000000D-06  2.000000000D-05
0.000000000D+00  0.000000000D+00
            0.000000000D+00  3.000000000D-06  1.000000000D-06  0.000000000D+00
0.000000000D+00  2.000000000D-06
            5.000000000D-07  1.000000000D-07  0.000000000D+00  0.000000000D+00
2.000000000D-06  4.000000000D-05
            1.000000000D-07  0.000000000D+00  0.000000000D+00  0.000000000D+00
1.000000000D-06  0.000000000D+00
            0.000000000D+00  0.000000000D+00  0.000000000D+00  0.000000000D+00
0.000000000D+00  0.000000000D+00
            0.000000000D+00  0.000000000D+00  0.000000000D+00  0.000000000D+00
0.000000000D+00  0.000000000D+00
            0.000000000D+00  0.000000000D+00  0.000000000D+00
    pMon=   0.000000000D+00  0.000000000D+00  0.000000000D+00
    xMon=   1.000000000D+00  0.000000000D+00  0.000000000D+00
    yMon=   0.000000000D+00  1.000000000D+00  0.000000000D+00
    zMon=   0.000000000D+00  0.000000000D+00  1.000000000D+00
    lMon=   1.000000000D+00
    nObs=    0
  ApType=   None
    zElt=   1.000000000D+22
PropType=   Geometric
 nECoord=   -6
```

4.5.5 Monomial Surfaces

MACOS has a second polynomial-based deformed-surface type, evaluated as a sum of Cartesian monomials. The generating function is:

(4.1)

The user must specify 120 coefficients in the prescription file.

4.5.6 Toric Surfaces

Toric surfaces are surfaces that have different shapes in some specified orthogonal x- and y-axes.Cylindrical and toroidal surfaces are two examples. Toric surfaces are distinguished from the anamorphic surfaces by there being a non-zero conic constant in ony 1 of the axes. MACOS toric surfaces are entered using the standard KrElt and KcElt format, except that two sets of these conic parameters are entered: one set for each of the x- and y-axis directions. The variable AnaCoef holds the values in the following order: Krx, Kry, Kcx. In addition, a specialized coordinate frame must be used to define the local x- and y-axes for these variables. This is the same specialized coordinate frame used in Section 4.5.4 and is defined by pMon, xMon, yMon, and zMon as shown in Figure 27. An example of an (cylindrical) anamorphic surface as defined in a .in-file is shown below

4.5.7 Anamorphic Surfaces

Anamorphic surfaces are surfaces that have different (conic) shapes in some specified orthogonal x- and y-axes. Cylindrical optics and astigmatic plates are two examples. MACOS anamorphic surfaces are entered using the standard KrElt and KcElt format, except that two sets of these conic parameters are entered: one set for each of the x- and

y-axis directions. The variable AnaCoef holds the values in the following order: Krx, Kry, Kcx, Kcy. In addition, a specialized coordinate frame must be used to define the local x- and y-axes for these variables. This is the same specialized coordinate frame used in Section 4.5.4 and is defined by pMon, xMon, yMon, and zMon as shown in Figure 27. An example of an (cylindrical) anamorphic surface as defined in a .in-file is shown below.

```
   iElt=    1
 EltName= ana_surf
 Element= Refractor
 Surface= Anamorph
  psiElt=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  RptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  IndRef=   1.000000000D+00
  Extinc=   1.000000000D+22
 AnaCoef=  -1.700000000D+02 -1.000000000D+00 -1.000000000D+22 0.000000000D+00
    pMon=   0.000000000D+00  0.000000000D+00  0.000000000D+00
    xMon=  -1.000000000D+00  0.000000000D+00  0.000000000D+00
    yMon=   0.000000000D+00  1.000000000D+00  0.000000000D+00
    zMon=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
    nObs=    0
  ApType=   None
    zElt=   1.000000000D+22
PropType=   Geometric
 nECoord=   -6
```

4.5.8 User-Defined Surfaces: Deformable Mirrors

MACOS comes with 5 built-in user-defined surfaces. These surfaces use influence functions to model standard deformable mirror configurations for adaptive optics applications. They assume a grid of push-pull actuators spread out in regular patterns across the surface of the element. Coefficients that define the extension of each actuator are read from a file (or are directly modified by a calling program, if you are using SMACOS). The mirror coefficient file is specified in the prescription (or through the `MODIFY` command) as:

`UDSrfFile='`filename`'`

The supplied deformable mirror types are listed in order of `UDSrfType`:

1. 349-actuator deformable mirror. Actuators are located on a 7-mm spacing square grid. Units are inches.
2. 349-actuator deformable mirror. Actuators are located on a 7-mm spacing square grid. Units are cm.
3. 397-actuator deformable mirror. Actuators are located on a 7-mm spacing hex grid. Units are cm.
4. 349-actuator deformable mirror with squared-out influence function. Actuators are located on a 7-mm spacing square grid. Units are cm.
5. 349-actuator deformable mirror with ideal actuator influence functions (bilinear approximation). Actuators are located on a 7-mm spacing square grid. Units are inches.

6. 349-actuator deformable mirror with squared-out influence function. Actuators are located on a 7-mm spacing square grid. Units are cm.

An example of the use of deformable mirrors, as well as lenslet arrays and the ATMOS atmospheric phase disturbance command, is provided in Appendix A.4.

4.5.9 Interpolated Surfaces Using Regularly Gridded Data

MACOS provides interpolated surfaces of 2 types. The first type uses regular square gridded data, in the form of a matrix, whose rows and columns are filled with values giving the surface height deviation from a base conicoid of revolution. This form of interpolated surface, described in this section, is compatible with the high-density information typically generated by optical test interferometers. It uses linear interpolation to determine the intercept point of each ray at the optical surface, rather than the more compute-intensive 3rd-order interpolation of the more general X-Y-Z interpolated surface discussed in the next section.

Grid-data surfaces are defined using 3 additional parameters. These are:

- mGridMat: the dimension of the gridded data matrix (nGridMat by nGridMat)
- GridSrfdx: spacing of the data grids in base units
- GridFile: name of the text file containing the nGridMat by nGridMat surface matrix

The maximum number of grid-data surfaces, and the maximum number of points for grid-data surfaces, are limited at compile time, by setting the mGridSrf and mGridMat parameters in the param.cmn file.

The surface data file GridFile should be in ASCII text, with nGridMat rows and nGridMat columns. The data is read in using the following Fortran READ sequence:

```
((GridMat(i,j,jGridSrf),i=1,nGridMat),j=1,nGridMat)
```

The i and j indices correspond to the xMon and yMon coordinates, respectively:. The matrix is assumed centered at the pMon point. Increasing index values represent steps of size GridSrfdx along that axis. The maximum extent of the surface is ±(nGridMat/2)*GridSrfdx in either direction. Rays that fall outside the covered area are passed, but using only the base conic to determine direction and pathlength.

The data is read in when the prescription is read. No setup computations are required (no SINT command). Interpolation data can also be provided to SMACOS directly from a calling program that includes the appropriate common statements (see Section 9.2.3).

4.5.10 Interpolated Surfaces Using X-Y-Z Data

Interpolated surfaces using X-Y-Z data triplets allow arbitrary perturbations to be applied to a base conic-of-revolution surface, using irregularly sampled data. Compared to the gridded-data surface of the previous section, the X-Y-Z interpolated surfaces are slower to compute, using a 3rd-order, triangular, continuous surface interpolation algorithm. They also require large amounts of RAM.

The maximum number of interpolated data surfaces, and the maximum number of points for interpolated data surfaces, are limited at compile time, by setting the mIntSrf and mDP parameters in the param.cmn file. These settings are shown in the command window as MACOS is started up.

The interpolation data is entered from an input file, which is not read until the user executes a SINT (Surface INTerpolator) command. The SINT command reads the data files defining the surface(s) and precomputes interpolation coefficients used in the ray trace.

If rays are traced prior to executing the SINT command, the interpolated surfaces are converted to conic surfaces and the interpolation data is not used.

Interpolation data can also be provided to SMACOS directly from a calling program that includes the appropriate common statements (see Section 9.2.3).

The input data file can be in either binary or text data format. A binary surface data file is expected to have the name "`filnam.srfN.bin`," where `filnam` is the .in-file root name and *N* is the element number. A text surface data file is expected to have the name "`filnam.srfN.txt`." If no file with the expected name is found, the user is prompted for a different file name. If none is provided, the surface is not initialized and will be converted to a conic when rays are traced.

The first line of the file contains two integer numbers: `nDP`, the number of nodes for the optical element; and `iComputeDZ`, which indicates whether slope data is provided or not. If `iComputeDZ=1`, slopes are to be estimated from the height data; `iComputeDZ=0` indicates that slopes are provided in the data file.

The `nDP` subsequent lines of the file contain the data for the interpolation. The data for one node is in one row in the file. Each row has 3 or 5 real numbers, giving the x and y coordinates of the data point, the surface height at that point, and then, if `iComputeDZ=0`, the x and y slopes at that point. The grid defining the data does not need to be a regular grid. The grid points need not be in any particular order in the data file.

4.6 Obscurations and Apertures

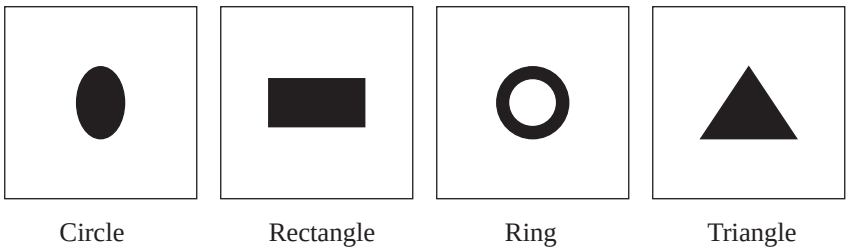
Apertures are the areas through which light can pass. MACOS supports circular and rectangular apertures as shown in Figure 28. When a wavefront is propagated, the intensity that does not go through the hole is set to zero. Likewise, obscurations block light that falls within their interior. Figure 29 shows the types of obscurations implemented in MACOS: circular, rectangular, annular, and triangular. Geometrical rays are unaffected by the obscurations; the rays are traced through the optical system regardless. However, the `OBScuration` command will allow one to view only the unobscured rays in the spot diagram.

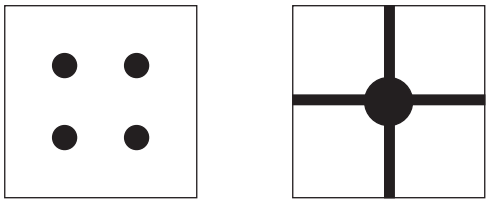
When entering data for obscurations and apertures, `fElt, eElt`, and, `psiElt` are required inputs. This may seem unnatural, but including these properties allows curvature to be given to apertures and obscurations. If an aperture or obscuration is planar, then `fElt=1d22` and `eElt=0`. In this case, `psiElt` is normal to the plane. However, if the obscuration or aperture is not planar, such as an obscuration caused by mirror support pads, `fElt, eElt`, and, `psiElt` are the same as they would be for that mirror. All obscuring apertures are defined on the plane defined by `psiElt` and `VptElt` and projected onto the surface.

This section has three different examples. The first example is of an element representing a square aperture in front of the primary mirror.

```
Enter Element   1 Data:
Enter element name (no blanks)  [Elt]: Sq_Ap                    EltName
+-----------------------------------------+
| Element types:                          |
```

FIGURE 28 Examples of apertures types.





Composite obscurations

FIGURE 29 Examples of obscuration types and some composite obscurations.

```
|   Reflector      NSReflector    Segment    |
|   Refractor      NSRefractor    LensArray  |
|   FocalPlane     Reference      Return     |
|   HOE            Grating        Obscuring  |
+-------------------------------------------+
Enter element type :Obscuring                                    Element
+-------------------------------------------+
| Surface types:                            |
|   Flat           Conic          Aspheric   |
|   Anamorphic     Zernike        Monomial   |
|   Interpolated   UserDefined               |
+-------------------------------------------+
Enter surface type :Flat                                         Surface
Enter element vertex location (x,y,z)  [no default]: 0,0,-5.4     VptElt
Enter element rotation point (x,y,z)  [0.,0.,-5.40000]:0,0,-5.4   RptElt
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,-1     psiElt
Do you want an aperture on this element? [NO]: yes
+-------------------------------------------+
| Aperture types:                           |
|   Circular       Rectangular              |
+-------------------------------------------+
Enter aperture type  [Circular]: Rectangular                     ApType
Enter aperture xmin,xmax,ymin,ymax  [-0.500000,0.500000,-0.500000,0.500000]: -
1.5,1.5,-1.5,1.5                                                 ApVec
Enter number of obscurations  [0]:0                               nObs
Enter element x-axis vector (x,y,z)  [-1.00000,0.,0.]:1,0,0       xObs
Enter Fresnel propagation distance  [1.000000E+22]:1d22           zElt
+-------------------------------------------+
| Propagation types:                        |
```

```
|   Geometric      GeomUpdate     FarField   |
|   NFSpherical    NFS1surf                  |
|   NFPlane        NFP1surf                  |
|   NF1            NF2                       |
|   SpatialFilter  SF1surf                   |
+--------------------------------------------+
Enter propagation type  [Geometric]: Geometric                          PropType
Do you wish to use element-based coordinates?: [NO]:
```

This dialog produced the following prescription data:

```
   iElt=    1
 EltName= Sq_Ap
 Element= Obscuring
 Surface= Flat
   KrElt=  -1.000000000D+22
   KcElt=   0.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00 -5.400000000D+00
  RptElt=   0.000000000D+00  0.000000000D+00 -5.400000000D+00
  IndRef=   1.000000000D+00
  Extinc=   0.000000000D+00
    nObs=    0
  ApType=   Rectangular
   ApVec=  -1.500000000D+00  1.500000000D+00 -1.500000000D+00  1.50000000D+00
    zElt=   1.000000000D+22
PropType=   Geometric
 nECoord=   -6
```

The second example shows an obscuration consisting of four circles (see Figure 29). This is similar to what was done to model the Hubble Space Telescope's primary mirror which was drilled in three places for rubber retaining rings. This example is a Cassegrain telescope example with four support pads on the primary mirror. `fElt` and `eElt` are the same as the primary mirror. `NGridpts` was also increased so the obscurations could be seen in the focal plane. Note that the support pads are located just in front of the primary mirror.

```
Enter Element   1 Data:
Enter element name (no blanks)  [Elt]: 4CircObs                         EltName
+--------------------------------------------+
| Element types:                             |
|   Reflector      NSReflector    Segment    |
|   Refractor      NSRefractor    LensArray  |
|   FocalPlane     Reference      Return     |
|   HOE            Grating        Obscuring  |
+--------------------------------------------+
Enter element type :Obscuring                                           Element
+--------------------------------------------+
| Surface types:                             |
|   Flat           Conic          Aspheric   |
|   Anamorphic     Zernike        Monomial   |
|   Interpolated   UserDefined                |
+--------------------------------------------+
Enter surface type :Conic                                               Surface
Enter element vertex location (x,y,z)  [no default]: 0,0,-.01            VptElt
Enter element rotation point (x,y,z)  [0.,0.,-1.000000E-02]:0,0,-.01     RptElt
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,-1            psiElt
Enter e&f or Kc&Kr (1=e&f,2=Kc&Kr)?  [1]:1
Enter element focal length  [1.000000E+22]:5.4                             fElt
Enter element eccentricity  [0.]:1                                         eElt
```

```
Do you want an aperture on this element? [NO]: yes
+-----------------------------------------+
| Aperture types:                         |
|   Circular       Rectangular            |
+-----------------------------------------+
Enter aperture type  [Circular]: Circular                              ApType
Enter aperture radius, x, y  [0.500000,0.,0.]:2,0,0                    ApVec
Enter number of obscurations  [0]:4                                    nObs
+-----------------------------------------+
| Obscuration types:                      |
|   Circle         NegCircle              |
|   Rectangle      NegRectangle           |
|   Annulus        NegAnnulus             |
|   Triangle       NegTriangle            |
|   Ellipse        NegEllipse             |
+-----------------------------------------+
Enter obscuration type  [Circle]: Circle                               ObsType
Enter obscuration radius, x, y  [no default]: 0.2,1,1                  ObsVec
Enter obscuration type  [Circle]: Circle                               ObsType
Enter obscuration radius, x, y  [no default]: 0.2,1,-1                 ObsVec
Enter obscuration type  [Circle]: Circle                               ObsType
Enter obscuration radius, x, y  [no default]: 0.2,-1,1                 ObsVec
Enter obscuration type  [Circle]: Circle                               ObsType
Enter obscuration radius, x, y  [no default]: 0.2,-1,-1                ObsVec
Enter element x-axis vector (x,y,z)  [-1.00000,0.,0.]:1,0,0            xObs
Enter Fresnel propagation distance  [5.40000]:5.4                      zElt
+-----------------------------------------+
| Propagation types:                      |
|   Geometric      GeomUpdate     FarField |
|   NFSpherical    NFS1surf                |
|   NFPlane        NFP1surf                |
|   NF1            NF2                     |
|   SpatialFilter  SF1surf                 |
+-----------------------------------------+
Enter propagation type  [Geometric]: Geometric                         PropType
Do you wish to use element-based coordinates?: [NO]:
```

This dialog produced the following prescription data:

```
   iElt=    1
 EltName= 4CircObs
 Element= Obscuring
 Surface= Conic
   KrElt=  -1.080000000D+01
   KcElt=  -1.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00 -1.000000000D-02
  RptElt=   0.000000000D+00  0.000000000D+00 -1.000000000D-02
  IndRef=   1.000000000D+00
  Extinc=   0.000000000D+00
    nObs=    4
 ObsType=   Circle
  ObsVec=   2.000000000D-01  1.000000000D+00  1.000000000D+00
 ObsType=   Circle
  ObsVec=   2.000000000D-01  1.000000000D+00 -1.000000000D+00
 ObsType=   Circle
  ObsVec=   2.000000000D-01 -1.000000000D+00  1.000000000D+00
 ObsType=   Circle
  ObsVec=   2.000000000D-01 -1.000000000D+00 -1.000000000D+00
    xObs=   1.000000000D+00  0.000000000D+00  0.000000000D+00
  ApType=   Circular
   ApVec=   2.000000000D+00  0.000000000D+00  0.000000000D+00
    zElt=   5.400000000D+00
PropType=   Geometric
 nECoord=   -6
```

The third example shows an obscuration produced by a secondary mirror and it's support spider: two rectangles and a circle (see Figure 29). These planar obscurations obscur the primary mirror.

```
Enter Element   1 Data:
Enter element name (no blanks)  [Elt]: SpiderObs                          EltName
+-----------------------------------------+
| Element types:                          |
|   Reflector      NSReflector    Segment |
|   Refractor      NSRefractor    LensArray|
|   FocalPlane     Reference      Return   |
|   HOE            Grating        Obscuring|
+-----------------------------------------+
Enter element type :Obscuring                                             Element
+-----------------------------------------+
| Surface types:                          |
|   Flat           Conic          Aspheric |
|   Anamorphic     Zernike        Monomial |
|   Interpolated   UserDefined             |
+-----------------------------------------+
Enter surface type :Flat                                                  Surface
Enter element vertex location (x,y,z)  [no default]: 0,0,-5.4              VptElt
Enter element rotation point (x,y,z)  [0.,0.,-5.40000]:0,0,-5.4            RptElt
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,-1              psiElt
Do you want an aperture on this element? [NO]: yes
+-----------------------------------------+
| Aperture types:                         |
|   Circular       Rectangular            |
+-----------------------------------------+
Enter aperture type  [Circular]: Circular                                 ApType
Enter aperture radius, x, y  [0.500000,0.,0.]:2,0,0                       ApVec
Enter number of obscurations  [0]:3                                       nObs
+-----------------------------------------+
| Obscuration types:                      |
|   Circle         NegCircle              |
|   Rectangle      NegRectangle           |
|   Annulus        NegAnnulus             |
|   Triangle       NegTriangle            |
|   Ellipse        NegEllipse             |
+-----------------------------------------+
Enter obscuration type  [Circle]: Circle                                  ObsType
Enter obscuration radius, x, y  [no default]: 0.5,0,0                     ObsVec
Enter obscuration type  [Circle]: Rectangle                               ObsType
Enter obscuration xmin, xmax, ymin, ymax  [no default]: -.25,.25,-2,2     ObsVec
Enter obscuration type  [Circle]: Rectangle                               ObsType
Enter obscuration xmin, xmax, ymin, ymax  [no default]: -2,2,-.25,.25     ObsVec
Enter element x-axis vector (x,y,z)  [-1.00000,0.,0.]:1,0,0               xObs
Enter Fresnel propagation distance  [1.000000E+22]:1d22                   zElt
+-----------------------------------------+
| Propagation types:                      |
|   Geometric      GeomUpdate     FarField|
|   NFSpherical    NFS1surf               |
|   NFPlane        NFP1surf               |
|   NF1            NF2                    |
|   SpatialFilter  SF1surf                |
+-----------------------------------------+
Enter propagation type  [Geometric]: Geometric                            PropType
Do you wish to use element-based coordinates?: [NO]:
```

This dialog produced the following prescription data:

```
   iElt=    1
 EltName= SpiderObs
 Element= Obscuring
 Surface= Flat
   KrElt=  -1.000000000D+22
   KcElt=   0.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00 -5.400000000D+00
  RptElt=   0.000000000D+00  0.000000000D+00 -5.400000000D+00
  IndRef=   1.000000000D+00
  Extinc=   0.000000000D+00
    nObs=    3
 ObsType=   Circle
  ObsVec=   5.000000000D-01  0.000000000D+00  0.000000000D+00
 ObsType=   Rectangle
  ObsVec=  -2.500000000D-01  2.500000000D-01 -2.000000000D+00  2.00000000D+00
 ObsType=   Rectangle
  ObsVec=  -2.000000000D+00  2.000000000D+00 -2.500000000D-01  2.50000000D-01
    xObs=   1.000000000D+00  0.000000000D+00  0.000000000D+00
  ApType=   Circular
   ApVec=   2.000000000D+00  0.000000000D+00  0.000000000D+00
    zElt=   1.000000000D+22
PropType=   Geometric
 nECoord=   -6
```

4.7 Specifying Coatings

Multi-layer coatings are not supported in this release of MACOS.

4.8 Example Telescope

In Section 4.2.2 the source data for the Cassegrain telescope example was entered. This section finishes the example adding the necessary optical elements: the primary mirror, secondary mirror, reference surface, and focal plane. Figure 15 shows the telescope. Table 9 lists the optical design.

TABLE 9. Cassegraing telescope design

Element	Focal length	Eccentricity	Diameter	Distance from primary
Primary Mirror	5.4	1.0	4.0	
Secondary Mirror	1.338854098	1.634183595	1.0	4.061145901
Focal Plane	1d22	0.0		1.5

Continuation of MACOS input from Section 4.2.2.

```
Enter Element   1 Data:
Enter element name (no blanks)  [Elt]: SecMirObs
+-----------------------------------------+
| Element types:                          |
|   Reflector      NSReflector    Segment    |
|   Refractor      NSRefractor    LensArray  |
|   FocalPlane     Reference      Return     |
|   HOE            Grating        Obscuring  |
+-----------------------------------------+
Enter element type :Obscuring
+-----------------------------------------+
| Surface types:                          |
|   Flat           Conic          Aspheric   |
|   Anamorphic     Zernike        Monomial   |
```

```
|   Interpolated   UserDefined              |
+-------------------------------------------+
Enter surface type :Flat
Enter element vertex location (x,y,z)  [no default]: 0,0,-5.4
Enter element rotation point (x,y,z)  [0.,0.,-5.40000]:0,0,-5.4
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,-1
Do you want an aperture on this element? [NO]: yes
+-------------------------------------------+
| Aperture types:                           |
|   Circular       Rectangular              |
+-------------------------------------------+
Enter aperture type  [Circular]: Circular
Enter aperture radius, x, y  [0.500000,0.,0.]:2,0,0
Enter number of obscurations  [0]:1
+-------------------------------------------+
| Obscuration types:                        |
|   Circle         NegCircle                |
|   Rectangle      NegRectangle             |
|   Annulus        NegAnnulus               |
|   Triangle       NegTriangle              |
|   Ellipse        NegEllipse               |
+-------------------------------------------+
Enter obscuration type  [Circle]: Circle
Enter obscuration radius, x, y  [no default]: 0.5,0,0
Enter element x-axis vector (x,y,z)  [-1.00000,0.,0.]:1,0,0
Enter Fresnel propagation distance  [1.000000E+22]:1d22
+-------------------------------------------+
| Propagation types:                        |
|   Geometric      GeomUpdate     FarField  |
|   NFSpherical    NFS1surf                 |
|   NFPlane        NFP1surf                 |
|   NF1            NF2                      |
|   SpatialFilter  SF1surf                  |
+-------------------------------------------+
Enter propagation type  [Geometric]: Geometric
Do you wish to use element-based coordinates?: [NO]:

   iElt=    1
 EltName= SecMirObs
 Element= Obscuring
 Surface= Flat
   fElt=   1.000000000D+22
   eElt=   0.000000000D+00
  KrElt=  -1.000000000D+22
  KcElt=   0.000000000D+00
 psiElt=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
 VptElt=   0.000000000D+00  0.000000000D+00 -5.400000000D+00
 RptElt=   0.000000000D+00  0.000000000D+00 -5.400000000D+00
 IndRef=   1.000000000D+00
 Extinc=   0.000000000D+00
   nObs=    1
ObsType=   Circle
 ObsVec=   5.000000000D-01  0.000000000D+00  0.000000000D+00
   xObs=   1.000000000D+00  0.000000000D+00  0.000000000D+00
 ApType=   Circular
  ApVec=   2.000000000D+00  0.000000000D+00  0.000000000D+00
   zElt=   1.000000000D+22
PropType=  Geometric
 nECoord=  -6
```

```
 Is this correct? [YES]:

 Enter Element   2 Data:
 Enter element name (no blanks)  [Elt]: Primary
 +-----------------------------------------+
 | Element types:                          |
 |   Reflector      NSReflector    Segment    |
 |   Refractor      NSRefractor    LensArray  |
 |   FocalPlane     Reference      Return     |
 |   HOE            Grating        Obscuring  |
 +-----------------------------------------+
 Enter element type :Reflector
 +-----------------------------------------+
 | Surface types:                          |
 |   Flat           Conic          Aspheric   |
 |   Anamorphic     Zernike        Monomial   |
 |   Interpolated   UserDefined               |
 +-----------------------------------------+
 Enter surface type :Conic
 Enter element vertex location (x,y,z)  [no default]: 0,0,0
 Enter element rotation point (x,y,z)  [0.,0.,0.]:0,0,0
 Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,-1
 Enter e&f or Kc&Kr (1=e&f,2=Kc&Kr)?  [1]:1
 Enter element focal length  [1.000000E+22]:5.4
 Enter element eccentricity  [0.]:1
 Do you want an aperture on this element? [NO]:
 Enter number of obscurations  [0]:
 Enter Fresnel propagation distance  [5.40000]:5.4
 +-----------------------------------------+
 | Propagation types:                      |
 |   Geometric      GeomUpdate     FarField   |
 |   NFSpherical    NFS1surf                  |
 |   NFPlane        NFP1surf                  |
 |   NF1            NF2                       |
 |   SpatialFilter  SF1surf                   |
 +-----------------------------------------+
 Enter propagation type  [Geometric]: Geometric
 Do you wish to use element-based coordinates?: [NO]:
 Enter element index of refraction  [1.00000]:
 Enter element extinction coefficient  [1.000000E+22]:

    iElt=    2
 EltName= Primary
 Element= Reflector
 Surface= Conic
    fElt=   5.400000000D+00
    eElt=   1.000000000D+00
   KrElt=  -1.080000000D+01
   KcElt=  -1.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  RptElt=   0.000000000D+00  0.000000000D+00  0.000000000D+00
  IndRef=   1.000000000D+00
  Extinc=   1.000000000D+22
    nObs=    0
  ApType=   None
    zElt=   5.400000000D+00
PropType=   Geometric
 nECoord=   -6
 Is this correct? [YES]:

 Enter Element   3 Data:
 Enter element name (no blanks)  [Elt]: Secondary
 +-----------------------------------------+
```

```
| Element types:                              |
|   Reflector      NSReflector    Segment    |
|   Refractor      NSRefractor    LensArray  |
|   FocalPlane     Reference      Return     |
|   HOE            Grating        Obscuring  |
+---------------------------------------------+
Enter element type :Reflector
+---------------------------------------------+
| Surface types:                              |
|   Flat           Conic          Aspheric    |
|   Anamorphic     Zernike        Monomial    |
|   Interpolated   UserDefined                |
+---------------------------------------------+
Enter surface type :Conic
Enter element vertex location (x,y,z)  [no default]: 0,0,-4.0611459018
Enter element rotation point (x,y,z)  [0.,0.,-4.06115]:0,0,-5.4
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,-1
Enter e&f or Kc&Kr (1=e&f,2=Kc&Kr)?  [1]:1
Enter element focal length  [1.000000E+22]:1.3388540982
Enter element eccentricity  [0.]:1.6341835953
Do you want an aperture on this element? [NO]:
Enter number of obscurations  [0]:
Enter Fresnel propagation distance  [1.33885]:1.3388540982
+---------------------------------------------+
| Propagation types:                          |
|   Geometric      GeomUpdate     FarField    |
|   NFSpherical    NFS1surf                   |
|   NFPlane        NFP1surf                   |
|   NF1            NF2                        |
|   SpatialFilter  SF1surf                    |
+---------------------------------------------+
Enter propagation type  [Geometric]: Geometric
Do you wish to use element-based coordinates?: [NO]:
Enter element index of refraction  [1.00000]:
Enter element extinction coefficient  [1.000000E+22]:

   iElt=    3
 EltName= Secondary
 Element= Reflector
 Surface= Conic
   fElt=   1.338854098D+00
   eElt=   1.634183595D+00
  KrElt=  -3.526787502D+00
  KcElt=  -2.670556023D+00
 psiElt=   0.000000000D+00  0.000000000D+00 -1.000000000D+00
 VptElt=   0.000000000D+00  0.000000000D+00 -4.061145902D+00
 RptElt=   0.000000000D+00  0.000000000D+00 -5.400000000D+00
 IndRef=   1.000000000D+00
 Extinc=   1.000000000D+22
   nObs=    0
 ApType=   None
   zElt=   1.338854098D+00
PropType=  Geometric
 nECoord=   -6
 Is this correct? [YES]:

 Enter Element   4 Data:
 Enter element name (no blanks)  [Elt]: Ref_surf
 +---------------------------------------------+
```

```
| Element types:                              |
|   Reflector      NSReflector    Segment    |
|   Refractor      NSRefractor    LensArray  |
|   FocalPlane     Reference      Return     |
|   HOE            Grating        Obscuring  |
+---------------------------------------------+
Enter element type :Reference
+---------------------------------------------+
| Surface types:                              |
|   Flat           Conic          Aspheric    |
|   Anamorphic     Zernike        Monomial    |
|   Interpolated   UserDefined                |
+---------------------------------------------+
Enter surface type :Conic
Enter element vertex location (x,y,z)  [no default]: 0,0,-4.0601459018
Enter element rotation point (x,y,z)  [0.,0.,-4.06015]:0,0,-4.0601459018
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,1
Enter e&f or Kc&Kr (1=e&f,2=Kc&Kr)?  [1]:1
Enter element focal length  [1.000000E+22]:5.5601459018
Enter element eccentricity  [0.]:0
Do you want an aperture on this element? [NO]:
Enter number of obscurations  [0]:
Enter Fresnel propagation distance  [5.56015]:5.5601459018
+---------------------------------------------+
| Propagation types:                          |
|   Geometric      GeomUpdate     FarField    |
|   NFSpherical    NFS1surf                   |
|   NFPlane        NFP1surf                   |
|   NF1            NF2                        |
|   SpatialFilter  SF1surf                    |
+---------------------------------------------+
Enter propagation type  [Geometric]: FarField
Do you wish to use element-based coordinates?: [NO]:

   iElt=    4
 EltName= Ref_surf
 Element= Reference
 Surface= Conic
   fElt=   5.560145902D+00
   eElt=   0.000000000D+00
  KrElt=  -5.560145902D+00
  KcElt=   0.000000000D+00
 psiElt=   0.000000000D+00  0.000000000D+00  1.000000000D+00
 VptElt=   0.000000000D+00  0.000000000D+00 -4.060145902D+00
 RptElt=   0.000000000D+00  0.000000000D+00 -4.060145902D+00
 IndRef=   1.000000000D+00
 Extinc=   0.000000000D+00
   nObs=    0
 ApType=   None
   zElt=   5.560145902D+00
PropType=   FarField
 nECoord=   -6
 Is this correct? [YES]:

 Enter Element   5 Data:
 Enter element name (no blanks)  [Elt]: foc_pln
 +---------------------------------------------+
 | Element types:                              |
 |   Reflector      NSReflector    Segment    |
 |   Refractor      NSRefractor    LensArray  |
 |   FocalPlane     Reference      Return     |
 |   HOE            Grating        Obscuring  |
 +---------------------------------------------+
 Enter element type :FocalPlane
```

```
+-----------------------------------------+
| Surface types:                          |
|   Flat          Conic         Aspheric  |
|   Anamorphic    Zernike       Monomial  |
|   Interpolated  UserDefined             |
+-----------------------------------------+
Enter surface type :Flat
Enter element vertex location (x,y,z)  [no default]: 0,0,1.5
Enter element rotation point (x,y,z)  [0.,0.,1.50000]:0,0,1.5
Enter element principal axis (x,y,z)  [0.,0.,-1.00000]:0,0,1
Do you want an aperture on this element? [NO]:
Enter number of obscurations  [0]:
Enter Fresnel propagation distance  [1.000000E+22]:0
+-----------------------------------------+
| Propagation types:                      |
|   Geometric     GeomUpdate    FarField  |
|   NFSpherical   NFS1surf                |
|   NFPlane       NFP1surf                |
|   NF1           NF2                     |
|   SpatialFilter SF1surf                 |
+-----------------------------------------+
Enter propagation type  [Geometric]: Geometric
Do you wish to use element-based coordinates?: [NO]:

   iElt=    5
 EltName= foc_pln
 Element= FocalPlane
 Surface= Flat
   fElt=   1.000000000D+22
   eElt=   0.000000000D+00
  KrElt=  -1.000000000D+22
  KcElt=   0.000000000D+00
 psiElt=   0.000000000D+00  0.000000000D+00  1.000000000D+00
 VptElt=   0.000000000D+00  0.000000000D+00  1.500000000D+00
 RptElt=   0.000000000D+00  0.000000000D+00  1.500000000D+00
 IndRef=   1.000000000D+00
 Extinc=   0.000000000D+00
   nObs=    0
 ApType=   None
   zElt=   0.000000000D+00
PropType=  Geometric
 nECoord=  -6

nOutCord=    0
   Tout=
 Is this correct? [YES]:

 Enter xOut (x,y,z)  [-1.00000,0.,0.]:1,0,0
 Enter yOut (x,y,z)  [0.,1.00000,0.]:0,1,0
```

This completes the input example. The file is now saved as `Cassegrain.in` within the directory in which MACOS is running. The input file is included in Appendix A.1.

4.9 Editing .In-Files

Once a prescription has been entered and stored as a .in-file, the user can edit and reuse it for further analyses. Editing an .in-file is a matter of opening the file in the text editor

of your choice. On Unix machines this can be vi, Emacs, TextEdit, or another text editor. On the Macintosh, Qued-M, MPW Editor, BBEdit, Alpha, or other text or word processors can be used.

Files can be opened in directories other than the one MACOS is running in, by using appropriate path descriptors in specifying the file name. On Unix systems, the full name takes the form: "`subdirectory/subdirectory/filename`." On Macintosh systems, the full name takes the form: "`:subdirectory:subdirectory:filename`."

There are some simple rules for .in-files that must be followed to avoid ambiguities in the data. These are:

- The first data block must be for the source.
- Each element's data must be in a block starting with '`iElt=`' followed by the number of that element. All data within that data block refers to that element. The end of the data block is indicated by the next '`iElt=`' line or the '`nOutCord=`' line.
- Ordering within data blocks is generally not important, except that `nObs` must preceed the description of the obscurations.
- The last element data block must be followed by the '`nOutCord=`' specifying the number of output coordinates and the output coordinate matrix.
- Blank lines can be used to separate data for clarity.
- Comments must be preceeded by a '%' sign
- Data strings can be entered in free format. Up to 16 decimals of precision will be recognized.

Data for a single element must be grouped together, but the order of the data is not important as long as the '`iElt=`' statement is first. When a new '`iElt=`' statement is encountered, MACOS checks to make sure all the necessary data has been provided for the previous element. If data for an element has not been provided, MACOS will do one of two things. For some elements, MACOS will provide "default" data for that element. Other data is "necessary" for an .in-file and MACOS will print a warning that the .in-file was not loaded and the name of the omitted variable. A listing of the defaults provided for variables is included in Section 4.11.

4.10 Commands

MACOS provides the user several commands to help setup the optical prescription, to modify or perturb the optical prescription, and to list parameters of the optical system. Commands to help Some of these are described here; others are mentioned here but described elsewhere.

4.10.1 OLD and LOAD

Previously created .in-files are loaded using the `OLD` command. This command has one argument, the name of the input file (minus the ".in" suffix). For instance, to load the `Cassegrain.in` prescription:

```
MACOS> old Cassegrain
```

LOAd also work identically:

```
MACOS> loa Cassegrain
```

4.10.2 MODIFY

The MODify command allows the user to change the current optical prescription without editing the .in-file. These changes are not saved in a .in-file unless the SAVe command is used. This allows the user to perturb elements, modify the prescription, and save the results if desired. Any of the .in-file variables listed in Section 4.4.1 can be changed.

```
MACOS>mod
MOD: (q): h

MACOS MODify Function
MODify Syntax Examples:
   Flux=3
   ChfRayPos(1)=2-MOD 1st element of vector
   ChfRayPos(:)=2,0,0-MOD all elements of vector
   RptElt(3,2)=3-MOD 3rd element of vector for iElt=2
   RptElt(:,2)=1.5,3,0-MOD all elements of vector for iElt=2
Quit: quit Modify function.
```

To use the MODify command, just type MOD. The user will then be prompted for the modification. If a MODification is successful, it will be echoed at the terminal. For instance to change the eccentricity of element 2 to 1.03.

```
MACOS>mod
MOD: (q): eElt(2)=1.03
eElt( 2) = 1.030000000D+00
KcElt( 2) = -1.060900000D+00
KrElt( 2) = -2.842000000D+00
```

If a MOD is not successful, it will not be echoed.

```
MOD: (q): eeElt(2)=1.04
MOD: (q):
```

There are 2 ways to enter use MODify for vectors and arrays. The first is changing one element of the vector or array at a time:

```
MOD: (q): VptElt(3,2)=4.2
```

The second is to re-enter the entire vector:

```
MOD: (q): VptElt(:,2)=1.0,0.0,3.0
```

The SHOw command can be used to display the element's data.

4.10.3 SAVE

The SAVe command writes a new .in-file containing the current optical prescription data to disk. Parameters changed as the result of a PERturb or MODify command will be written as modified. This command takes one argument: the new .in-file name (without the ".in" suffix). If the specified file already exists, the user is asked if it should be overwritten.

4.10.4 SUMMARIZE

The SUMmarize command displays a brief summary of the loaded optical system, listing each element and showing its type, conic constant KcElt, and radius of curvature KrElt.

4.10.5 STATUS

The STAtus command displays information about some of the current states.

```
MACOS>status
Status for file cass_ap3:
Current element for OPD calculations= 4
Current element for WF/Spot calculations= 5
Obscuration option for ray-trace= 0
Current WF plot type=SLICE
REGRID option= F
Composed image= T
Pixel location set by chief ray option= F
```

4.10.6 RESET

The RESet command reinitializes some of the current states (see below) to their default values.

```
MACOS>reset
MACOS>status
Status for file cass_ap3:
Current element for OPD calculations= 0
Current element for WF/Spot calculations= 0
Obscuration option for ray-trace= 0
Current WF plot type=GRAY
REGRID option= F
Composed image= F
Pixel location set by chief ray option= F
```

4.10.7 SHOW

The SHOw command displays all the prescription data for a particular element. It has one argument, the element number.

4.11 Summary of Prescription Variables

Tables 10 through 12 provide a summary of the MACOS prescription variables and their default values. When MACOS loads an .in-file, these defaults are used variables, if the user has not supplied values. When a default is used, the user is notified, e.g.:

```
Default used for zSource
```

Not all variables have defaults, and these variables must be specified by the user. If one of these variables is not given a value in the .in-file, then the .in-file is not loaded and the user is notified; for instance:

```
VptElt(1) must be specified by user
Input file not properly loaded
```

The variables are shown with their index types in parentheses. These indices are:

- iObs: which obscuration in an element (can be up to 6 obscurations per element).
- iElt: number of the element in the .in-file.
- iAxis: ranges over the x,y,z coordinates in the global coordinate system.
- iPert: ranges over x,y,z rotations, x,y,z translations in the u-vector.
- iLin: ranges over the x,y,z components of the ray direction perturbation, the x,y,z components of the ray transverse aberration and the pathlength perturbation in the linear state vector x.
- iAsCoef: A, B, C, and D coefficients described in Section 4.5.3.

- iAp: either radius,x,y, or xmin,xmax,ymin,ymax depending on ApType.
- iZern: the 36 Zernike coefficients.
- iLayer: ranges over the coating layers.

All variables can be changed with the MODify command.

TABLE 10. Light source variables

Source Variables	Defaults	Description
Aperture	1.0D0	input aperture
BeamType	1 (uniform)	beam intensity
ChfRayDir(iAxis)	NO DEFAULT	chief ray direction
ChfRayPos(iAxis)	(0.0D0,0.0D0,0.0D0)	chief ray location
CosPower		
ExtInc(0)	0.0D0	extinction coefficient
Ex0		electric field strength of source along x-axis
Ey0		electric field strength of source along x-axis
Flux	1.0D0	total intensity received across the unobscured aperture
gap	0.0D0	spacing between segments in segmented apertures
GridType	1 (circular)	type of input aperture
IndRef(0)	1.0D0	real part of index of refraction of input medium
nElt	NO DEFAULT	number of elements
nGridPts	IFIX(mpts/2)	number of ray grid points across the aperture
nSeg	0	number of segments
Obscratn	0.0D0	input obscuration
rxBeam		beam diameter in x-direction
ryBeam		beam diameter in y-direction
SegCoord(xyz,iElt)	NO DEFAULT	segment location in the entrance pupil
Wavelen	6.0D-7	wavelength of light
width	Aperture	width of segments
xGrid(iAxis)	NO DEFAULT	orientation of input x-axis vector
yGrid(iAxis)	cross product of xGrid and ChRayDir	orientation of input y-axis vector
zGrid(iAxis)		orientation of input z-axis vector
Zelt(0)		
zSource	1.0D22	distance of light source from point of origin

TABLE 11. Element variables

Element variables	Default values	Description
AnaCoef(iAna,iElt)		anamorphic coefficients
ApType	1 (circle)	aperture type
ApVec(iAp,iElt)	NO DEFAULT	aperture definition information
AsphCoef(iAsCoef,iElt)	(0.0D0,0.0D0,0.0D0, 0.0D0)	asphere coefficients (4)
CoatIndxElt(iLayer, iElt)		index of coating layers
CoatThkElt(iLayer,iElt)		thickness of coating layers
eElt(iElt)	0.0D0	element eccentricity
EltName(iElt)	' ' (blanks)	element name
Element(iElt)	NO DEFAULT	element type
ExtInc(iElt)	0.0D0	extinction coefficient
fElt(iElt)	1.0D22	element focal length
h1HOE(iAxis,iElt)		
h2HOE(iAxis,iElt		
IndRef(iElt)	1.0D0	real part of refactive index for element
KcElt(iElt)	0.0D0	element conic constant
KrElt(iElt)	-1.0D22	element conic radius
LensArrayType(iElt)	1	type of lens in array (hex, square)
LensArrayWidth(iElt)	1	width of a single lens in the array
lMon(iElt)	Aperture	scale length for monomial (Zernike) element calculations
MonCoef(iMon,iElt)		monomial coefficients
nCoatElt(iElt)		number of coatings
nECoord(iElt)	-6	number of element coordinates
nLayer		number of layers in coating, maximum 20
nObs(iElt)	0	number of obscurations
ObsType(iObs,iElt)	1 (circle)	obscuration type
ObsVec(iAp,iObs,iElt)	(0.0D0,0.0D0,0.0D0, 0.0D0,0.0D0,0.0D0)	obscuration definition information
OrderHOE(iElt)		diffraction order of HOE to trace
PinHole()		pin hole
pMon(iAxis,iElt)	VptElt	reference for monomial (Zernike) surface calculations

TABLE 11. Element variables

Element variables	Default values	Description
`PropType(iElt)`	1 (Rays Only)	type of propagation from element `iElt` to element `iElt+1`
`psiElt(iAxis,iElt)`	NO DEFAULT	element principal axis vector
`RptElt(iAxis,iElt)`	`VptElt`	element rotation point
`RuleWidth(iElt)`		grating ruling width
`Surface(iElt)`		surface type
`TElt(iPert,iElt-Cord,iElt)`	(1.0D0,0.0D0,0.0D0, 0.0D0,0.0D0,0.0D0) (0.0D0,1.0D0,0.0D0, 0.0D0,0.0D0,0.0D0) (0.0D0,0.0D0,0.0D0, 1.0D0,0.0D0,0.0D0) (0.0D0,0.0D0,0.0D0, 1.0D0,0.0D0,0.0D0) (0.0D0,0.0D0,0.0D0, 0.0D0,1.0D0,0.0D0) (0.0D0,0.0D0,0.0D0, 0.0D0,0.0D0,1.0D0)	element local coordinates transformation matrix
`UDSrfCoef(iCoef,iElt)`		user defined surface coefficients
`UDSrfType(iElt)`		user defined surface type
`VptElt(iAxis,iElt)`	NO DEFAULT	element vertex point
`xMon(iAxis,iElt)`	`xGrid`	x-axis for monomial (Zernike) surface calculations
`xObs(iAxis,iElt)`	NO DEFAULT	orientation of x-axis for each element
`yMon(iAxis,iElt)`	cross product of `psiElt` and `xMon`	y-axis for monomial (Zernike) surface calculations
`WaveHOE(iElt)`		recording wavelength for HOE
`zElt(iElt)`	`fElt`	Fresnel propagation distance

TABLE 11. Element variables

Element variables	Default values	Description
`ZernCoef(iZern,iElt)`	(0.0D0,0.0D0,0.0D0, 0.0D0,0.0D0,0.0D0) (0.0D0,0.0D0,0.0D0, 0.0D0,0.0D0,0.0D0) (0.0D0,0.0D0,0.0D0, 0.0D0,0.0D0,0.0D0) (0.0D0,0.0D0,0.0D0, 0.0D0,0.0D0,0.0D0) (0.0D0,0.0D0,0.0D0, 0.0D0,0.0D0,0.0D0) (0.0D0,0.0D0,0.0D0, 0.0D0,0.0D0,0.0D0) (0.0D0,0.0D0,0.0D0)	Zernike coefficients
`zMon(iAxis,iElt)`	`psiElt`	z-axis for monomial (Zernike) surface calculations

TABLE 12. Output variables

Output Variables	Defaults	Description
`Tout(iLin,iPert)`	NO DEFAULT	the output coordinate transformation matrix
`nOutCord`	NO DEFAULT	number of output coordinates

SECTION 5 Ray-Trace Analysis

MACOS ray-trace commands define a bundle of rays at the light source, and then advance each ray element-by-element through the beam train. Individual rays can be followed through the system, or the entire bundle can be traced to display OPD maps or spot diagrams at particular elements. Placement of reference surfaces using ORS and SRS is introduced. The system exit pupil can be placed using the FEX command. Using the PERturb and MODify commands, individual elements can be altered and the analyses repeated. The commands defined in this section include:

```
DRAW                    MAP
RAY                     SPOt
OPD                     OBS
PERturb                 PREad
CENter                  STOP
FEX                     COORDinates
FFP                     PFP
POLarization            NOPolarizaton
SINT
```

5.1 Background

Having loaded the .in-file prescription that describes the optical system to MACOS, the user can trace rays to analyze the system. MACOS provides ray-trace commands that produce useful graphical (or file) outputs, such as SPOT and OPD. As discussed earlier, spot diagrams are graphs of the pierce points of the beam rays on a particular surface. They provide a measure of the geometric beam or image shape and are extremely useful in gauging performance. OPD maps show the wavefront profile at a particular surface. When used with an appropriate reference or return surface, OPD generates the wavefront error map of the system.

MACOS also provides commands to help setup appropriate reference surfaces. These include FEX, ORS and SRS (ORS and SRS are discussed later, in Section 6.3). Both SPOT and OPD provide means to quantify system performance, whether for design, tolerancing or simulation, for instrument-level or system-level analyses.

Detailed summaries of the paths taken by single rays can be generated using the RAY command. The optical train can be visualized using the DRAW command. These commands are useful in verifying design layout and correctness, as well as for single-ray performance measurements.

In addition to these commands, which generate specific data products, MACOS provides commands that can be used to change the system or the beam path in specific ways. The PERturb command implements a kinematically correct single-axis rotation and translation perturbation of an element or the beam source, and can be used to alter the system according to effects of misalignments or disturbances. The CENTER and STOP commands are used to set up the correct beam path through the optical train, by enforcing the system stop. The FEX command sets up return surfaces at exit pupils for OPD evaluation and for diffraction calculations. Commands such as FFP and PFP are

used to find the field angles that place the beam at particular points on an element, and are very useful in simulating imaging systems.

Before we describe the MACOS ray-trace commands, we will summarize how MACOS computes a ray-trace and introduce some of the internal MACOS variables that store ray-trace information. These variables are directly accessible to user programs that use the SMACOS include files.

A full beam ray-trace is initiated when the user issues an `OPD` or `SPOT` command (other commands, such as `BUILD`, `INTensity` or several of the other diffraction commands also initiate a full-beam ray-trace). Along with the command, the user specifies the element where the requested data is to be generated.

MACOS keeps track of the current position of the beam in the beam train. If the starting element for the trace is the source, then MACOS generates the starting ray states at the source point, using the source prescription data. Otherwise the trace starts where it left off previously. From the starting element, the beam is traced to the specified ending element. MACOS works in 2 loops to trace the beam from the starting element to the end element. The inner loop traces a single ray past each element until it reaches the specified ending element. The outer loop traces successive rays, following the order illustrated by the `MAP` command, until all rays have been traced.

In MACOS, "tracing rays" means updating specific ray-state variables so that they hold values appropriate for the specified end element. These variables include:

- `RayPos(1:3,iRay)` stores a 3-vector giving the position of the current segment of ray `iRay` in global coordinates. The current segment is the one that ends at the last element specified in an `OPD`, `SPOt`, `BUild` command. The last element after a diffraction propagation command (such as `INTensity`) depends on the type of propagation used just prior to the current element.
- `RayDir(1:3,iRay)` stores a unit magnitude 3-vector giving the direction of the *next* segment of ray `iRay`. If the current element is a mirror, `RayDir` gives the direction of the reflected ray. If it is a refracting surface, `RayDir` gives the direction of the refracted ray.
- `RayL(iRay)` stores a scalar giving the *geometric* length (not the optical path length which includes the effect of the index of refraction) of the *current segment* of ray `iRay`.
- `CumRayL(iRay)` stores a scalar giving the *cumulative optical* length (includes effects of non-unity index of refraction) of the ray `iRay` up to and including the current segment. It gives the optical pathlength from the source to the current element.
- `RayIndex(iRay)`stores the current index of refraction of the ray.

These definitions are illustrated in Figure 30.

5.2 Ray-Trace Commands

5.2.1 DRAW

The `DRAW` command produces a drawing of the beam train, showing the optical element surfaces and the beam propagating through them. The beam is represented by eleven rays. The elements are drawn in cross-section, using the ray-intercept points from 100 rays across the beam to define the surfaces. The drawing is in a specified (xz, yz or xy) plane, and the surfaces and rays are drawn as the projection of the onto that plane. Tilted or folded systems can sometimes produce strange results! Different colors are used to distinguish between rays and several different element types.

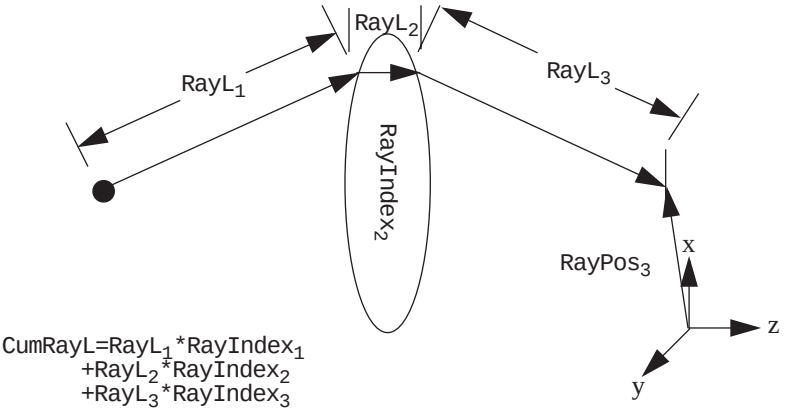


FIGURE 30 Ray state definitions

The user is asked to specify the starting element, the stopping element, and the viewing plane. The following example draws the ImageDemo.in prescription (see Appendix) beam train from the source (element 0) to element 11:

```
MACOS>draw
 Enter first element to include: [0]: 0
 Enter last element to include: [13]: 11
 Enter drawing plane (XZ, YZ or XY in source coords):  [XZ]:  xz
```

The result is shown in Figure 31. The system is somewhat aberrated, so the beam does not come to a clean focus! A close-up of refractive elements 6-11 is specified by changing the starting element:

```
MACOS>draw
 Enter first element to include: [0]: 6
 Enter last element to include: [13]: 11
 Enter drawing plane (XZ, YZ or XY in source coords):  [XZ]:  xz
```

FIGURE 31 DRAW views of ImageDemo.in

5.2.2 MAP

Single rays are specified by their "ray ID" number, iRay. This number must be specified in tracing individual rays, printing individual ray partials, or in accessing ray state data directly using the SMACOS include files.

The relative location of each ray in the pupil can be displayed using the MAP command. When there are more than 15 rays across the aperture, only a subset of the ray numbers are shown. The chief ray is always ray number 1 (iRay=1) and is not displayed by MAP. For Cassegrain.in, the MAP command produces the following:

```
MACOS>map
Map of rays at input aperture:
    0   0   0   0   0   0   0 150   0   0   0   0   0   0   0
    0   0   0   0 143 144 145 146 147 148 149   0   0   0   0
    0   0   0 134 135 136 137 138 139 140 141 142   0   0   0
    0   0 123 124 125 126 127 128 129 130 131 132 133   0   0
    0 110 111 112 113 114 115 116 117 118 119 120 121 122   0
    0  97  98  99 100 101 102 103 104 105 106 107 108 109   0
```

```
  0  84  85  86  87  88  89  90  91  92  93  94  95  96   0
 69  70  71  72  73  74  75  76  77  78  79  80  81  82  83
  0  56  57  58  59  60  61  62  63  64  65  66  67  68   0
  0  43  44  45  46  47  48  49  50  51  52  53  54  55   0
  0  30  31  32  33  34  35  36  37  38  39  40  41  42   0
  0   0  19  20  21  22  23  24  25  26  27  28  29   0   0
  0   0   0  10  11  12  13  14  15  16  17  18   0   0   0
  0   0   0   0   3   4   5   6   7   8   9   0   0   0   0
  0   0   0   0   0   0   0   2   0   0   0   0   0   0   0
MACOS>
```

The lower left corner is the (x=0, y=0) point for the aperture coordinates defined by the `ChfRayDir (=-zGrid)` and `xGrid` vectors. The view is looking into the optical system from the source.

When using `MAP` with a segmented system, a map of the segments is also printed, showing which segment each ray hits. For `SegmentDemo.in`, the following map is produced.

```
MACOS>map
Map of rays at input aperture:
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0 684 686 688 690 692 694 696 698   0   0   0   0
    0   0   0 645 647 649 651 653 655 657 659 661 663   0   0   0
    0   0 597 599 601 603 605 607 609 611 613 615 617 619   0   0
    0 544 546 548 550 552 554 556 558 560 562 564 566 568 570   0
    0 489 491 493 495 497 499 501 503 505 507 509 511 513 515   0
    0 431 433 435 437 439 441 443 445 447 449 451 453 455 457   0
    0 373 375 377 379 381 383 385 387 389 391 393 395 397 399   0
    0 313 315 317 319 321 323 325 327 329 331 333 335 337 339   0
    0 255 257 259 261 263 265 267 269 271 273 275 277 279 281   0
    0 197 199 201 203 205 207 209 211 213 215 217 219 221 223   0
    0 142 144 146 148 150 152 154 156 158 160 162 164 166 168   0
    0   0  93  95  97  99 101 103 105 107 109 111 113 115   0   0
    0   0   0  49  51  53  55  57  59  61  63  65  67   0   0   0
    0   0   0   0  14  16  18  20  22  24  26  28   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
Hit return to continue
Map of segments at input aperture:
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   5   5   5   5   4   4   4   4   0   0   0   0
    0   0   0   5   5   5   5   5   4   4   4   4   4   0   0   0
    0   0   5   5   5   5   5   5   4   4   4   4   4   4   0   0
    0   6   5   5   5   5   5   5   4   4   4   4   4   4   3   0
    0   6   6   6   5   5   5   1   1   4   4   4   3   3   3   0
    0   6   6   6   6   5   1   1   1   1   4   3   3   3   3   0
    0   6   6   6   6   6   1   1   1   1   3   3   3   3   3   0
    0   6   6   6   6   6   1   1   1   1   3   3   3   3   3   0
    0   6   6   6   6   7   1   1   1   1   2   3   3   3   3   0
    0   6   6   6   7   7   7   1   1   2   2   2   3   3   3   0
    0   6   7   7   7   7   7   7   2   2   2   2   2   2   3   0
    0   0   7   7   7   7   7   7   2   2   2   2   2   2   0   0
    0   0   0   7   7   7   7   7   2   2   2   2   2   0   0   0
    0   0   0   0   7   7   7   7   2   2   2   2   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
MACOS>
```

5.2.3 Undefined Rays

If a `MAP` command is used after a ray-trace or diffraction command such as `OPD`, `SPOt` or `INTensity`, some of the rays may disappear. This indicates that those rays became *undefined* because the rays:

- missed an optical element

- hit an optical element out of order
- hit two non-sequential surfaces simultaneously

as shown in Figure 32. Undefined rays are called out during the trace command. The surface where a ray became undefined can be pinpointed using the RAY command.

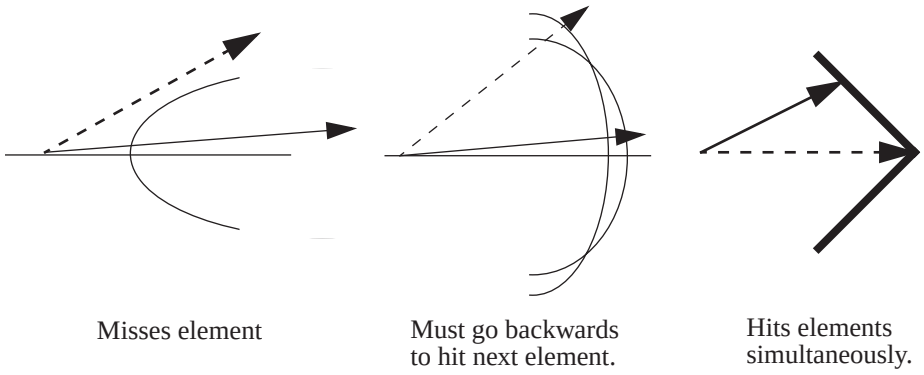


FIGURE 32 Example with defined rays (solid) and undefined rays (dashed)

5.2.4 RAY

The RAY command traces specific rays and prints the results. RAY has one argument, the number of the ray to be printed (iRay). The chief ray for Cassegrain.in is printed as:

```
MACOS>ray
Enter number of ray (1=chief ray, 0=quit):1
Ray     1 is the chief ray.
Ray    1 segment from Element    0 (InputRay) to Element    1 (SecMirOb):
  Starting point=  0.000000000D+00  0.000000000D+00 -6.000000000D+00
       End point=  0.000000000D+00  0.000000000D+00 -5.400000000D+00
       Direction=  0.000000000D+00  0.000000000D+00  1.000000000D+00
  Segment Length=  6.000000000D-01
    Total Length=  6.000000000D-01
      Index(i-1)=  1.000000000D+00  0.000000000D+00
        Index(i)=  1.000000000D+00  0.000000000D+00

Ray     1 is obscured by element  1
Ray    1 segment from Element    1 (SecMirOb) to Element    2 (Primary ):
  Starting point=  0.000000000D+00  0.000000000D+00 -5.400000000D+00
       End point=  0.000000000D+00  0.000000000D+00 -8.881784197D-16
       Direction=  0.000000000D+00  0.000000000D+00  1.000000000D+00
  Segment Length=  5.400000000D+00
    Total Length=  6.000000000D+00
      Index(i-1)=  1.000000000D+00  0.000000000D+00
        Index(i)=  1.000000000D+00  0.000000000D+00
Ray    1 segment from Element    2 (Primary ) to Element    3 (Secondar):
  Starting point=  0.000000000D+00  0.000000000D+00 -8.881784197D-16
       End point=  0.000000000D+00  0.000000000D+00 -4.061145902D+00
       Direction=  0.000000000D+00  0.000000000D+00 -1.000000000D+00
  Segment Length=  4.061145902D+00
    Total Length=  1.006114590D+01
      Index(i-1)=  1.000000000D+00  0.000000000D+00
        Index(i)=  1.000000000D+00  0.000000000D+00
Ray    1 segment from Element    3 (Secondar) to Element    4 (Ref_surf):
```

```
  Starting point=  0.000000000D+00  0.000000000D+00 -4.061145902D+00
       End point=  0.000000000D+00  0.000000000D+00 -4.060145902D+00
       Direction=  0.000000000D+00  0.000000000D+00  1.000000000D+00
  Segment Length=  1.000000000D-03
    Total Length=  1.006214590D+01
      Index(i-1)=  1.000000000D+00  0.000000000D+00
        Index(i)=  1.000000000D+00  0.000000000D+00
Ray    1 segment from Element    4 (Ref_surf) to Element    5 (foc_pln ):
  Starting point=  0.000000000D+00  0.000000000D+00 -4.060145902D+00
       End point=  0.000000000D+00  0.000000000D+00  1.500000000D+00
       Direction=  0.000000000D+00  0.000000000D+00  1.000000000D+00
  Segment Length=  5.560145902D+00
    Total Length=  1.562229180D+01
      Index(i-1)=  1.000000000D+00  0.000000000D+00
        Index(i)=  1.000000000D+00  0.000000000D+00
```

This printout gives the start and end point of each segment of the ray in global coordinates. The direction is a unit vector in global coordinates. The segment length refers to the geometric distance between the ray start and end points; it is not scaled by index of refraction. The total length gives a running sum of the length of the ray that is scaled by the index of refraction. The index give the index of refraction for the previous and current element.

To print the lower marginal ray, we see from the `MAP` command that the `iRay` number is 2, and respond accordingly:

```
MACOS>ray
Enter number of ray (1=chief ray, 0=quit):2
Ray     2 has aperture coordinates    8 and    1.
WF coordinates are  129 and  122.
Adjacent rays are    0,    0,    0,    6.
Ray    2 segment from Element    0 (InputRay) to Element    1 (SecMirOb):
  Starting point= -4.440892099D-16 -2.000000000D+00 -6.000000000D+00
       End point= -4.440892099D-16 -2.000000000D+00 -5.400000000D+00
       Direction=  0.000000000D+00  0.000000000D+00  1.000000000D+00
  Segment Length=  6.000000000D-01
    Total Length=  6.000000000D-01
      Index(i-1)=  1.000000000D+00  0.000000000D+00
        Index(i)=  1.000000000D+00  0.000000000D+00
Ray    2 segment from Element    1 (SecMirOb) to Element    2 (Primary ):
  Starting point= -4.440892099D-16 -2.000000000D+00 -5.400000000D+00
       End point= -4.440892099D-16 -2.000000000D+00 -1.851851852D-01
       Direction=  0.000000000D+00  0.000000000D+00  1.000000000D+00
  Segment Length=  5.214814815D+00
    Total Length=  5.814814815D+00
      Index(i-1)=  1.000000000D+00  0.000000000D+00
        Index(i)=  1.000000000D+00  0.000000000D+00
Ray    2 segment from Element    2 (Primary ) to Element    3 (Secondar):
  Starting point= -4.440892099D-16 -2.000000000D+00 -1.851851852D-01
       End point= -1.110223024D-16 -4.999999999D-01 -4.096296297D+00
       Direction=  7.951199381D-17  3.580901857D-01 -9.336870027D-01
  Segment Length=  4.188888889D+00
    Total Length=  1.000370370D+01
      Index(i-1)=  1.000000000D+00  0.000000000D+00
        Index(i)=  1.000000000D+00  0.000000000D+00
Ray    2 segment from Element    3 (Secondar) to Element    4 (Ref_surf):
  Starting point= -1.110223024D-16 -4.999999999D-01 -4.096296297D+00
       End point= -1.098674950D-16 -4.947992095D-01 -4.038085968D+00
       Direction=  1.975982230D-17  8.899032834D-02  9.960324902D-01
  Segment Length=  5.844219807D-02
    Total Length=  1.006214590D+01
      Index(i-1)=  1.000000000D+00  0.000000000D+00
        Index(i)=  1.000000000D+00  0.000000000D+00
Ray    2 segment from Element    4 (Ref_surf) to Element    5 (foc_pln ):
```

```
Starting point= -1.098674950D-16 -4.947992095D-01 -4.038085968D+00
     End point= -1.877761358D-26 -8.456685352D-11  1.500000000D+00
     Direction=  1.975982230D-17  8.899032834D-02  9.960324902D-01
Segment Length=  5.560145902D+00
  Total Length=  1.562229180D+01
    Index(i-1)=  1.000000000D+00  0.000000000D+00
      Index(i)=  1.000000000D+00  0.000000000D+00
```

This printout is similar to the chief ray printout. However, the first three lines are new. They specify the ray grid coordinates, the coordinates of the diffraction amplitude matrix grid point with which the ray is associated, and the `iRay` numbers of the adjacent rays. The adjacent rays are listed in (-x, +x, -y, +y) order.

5.2.5 OPD

The commands that perform full-beam ray-trace analyses are `SPOt` and `OPD`. The `OPD` (for Optical Path Difference) command traces the full bundle of rays to a specified element. At that element it computes an OPD for each ray and plots the beam OPD map. It also computes and prints the RMS OPD for the full beam.

The optical path difference of a ray is the difference between the optical pathlength of that ray and the chief ray. A positive `OPD` indicates that the ray travelled farther than the chief ray.

The `OPD` map as plotted is always a square grid corresponding to the aperture ray grid printed out in the `MAP` command. To see the true locations of the rays the `SPOt` command must be used. Rays which are obscured or vignetted by obscurations or apertures on the current or previous elements are not plotted by default. The `OBScuration` command changes which rays are plotted and is discussed in Section 5.2.7.

OPD plots will be representative of the system wavefront error *if evaluated at an exit pupil or other appropriate reference surface*. Such a surface will generally be located in the near-field region of the beam, at a point where the ray grid is regular.

NOTE: OPD plots made in a caustic or other region where the beam sampling geometry is disrupted will not auccurately capture system wavefront error. Better results may usually be obtained by using a near- or far-field diffraction propagation to the point of interest, and then plotting the phase of the complex amplitude (see Section 6).

The `OPD` command has a single argument `iElt` which is the number of the element where the `OPD` map or spot diagram is to be computed. The element numbering is as specified when creating the .in-file.

OPD plots are not affected by `STRETCH` setting. OPD plots are always on a linear scale.

The default plot type is `GRAy` scale. The plot type can be changed with the following commands: `WIRe`, `SLIce`, `GRAy`, `COLumn`, `ROW`, `CONtour`, `TEXt`, `MATlab`,`FITs`, or `BINary`. These plotting commands are discussed in more detail in Section 7.3.

The following examples show output of the `OPD` command with the `GRAY`, `WIRe`, and `BINary` plot types for `Cassegrain.in`.

```
MACOS>opd
Enter element where OPD is to be evaluated [0]:3
Tracing    150 rays...
Compute time was    0.1445 sec
RMS OPD error is  1.609556E-02
Graphics device/type (? to see list, default /NULL): /xw
```

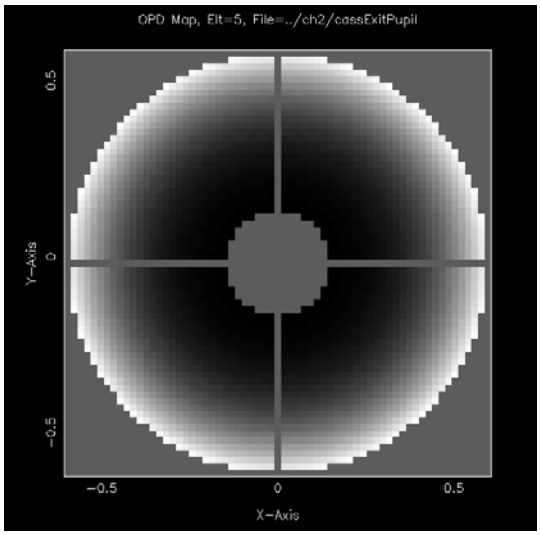


FIGURE 33 OPD GRAy scale plot

```
MACOS>wire
 Plot type set to WIRE
MACOS>opd
Enter element where OPD is to be evaluated [3]:3
RMS OPD error is  1.609556E-02
Type <RETURN> for next page:
```

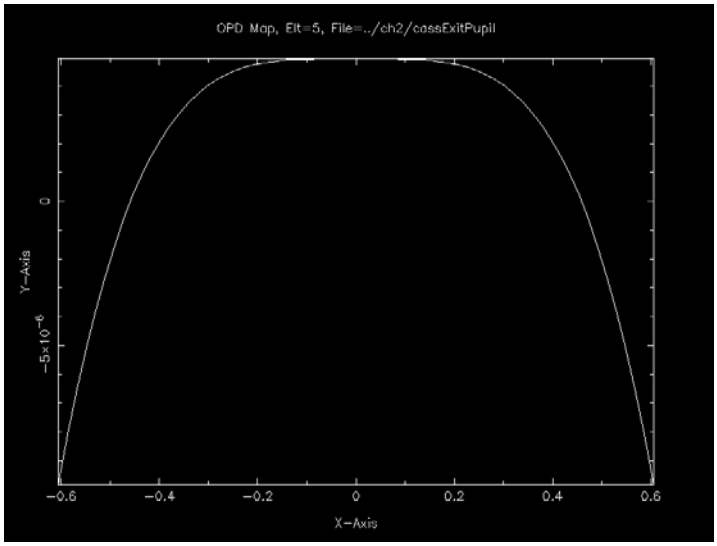


FIGURE 34 OPD ROW plot

```
MACOS>bin
 Plot type set to BINARY
MACOS>opd
Enter element where OPD is to be evaluated [3]:3
RMS OPD error is  1.609556E-02
Writing OPD Map, Elt=3
DIRECT ACCESS File=Cassegrain.OPD3
```

```
Binary array is of dimension 128  128  0. Magnitude of central pixel is
0.00000000000D+00
```

5.2.6 SPOT

Like the OPD command, the SPOt command traces the full bundle of rays to a specified element. At that element it computes and plots a spot diagram.

A spot diagram is an x-y plot across the beam. It puts a dot at the point of incidence of each ray as projected onto a plane across the specified element (see Figure 35).

Rays which are obscured or vignetted by obscurations or apertures on the current or previous elements are not plotted by default. The OBScuration command changes which rays are plotted and is discussed in Section 5.2.7.

The SPOt command has the argument iElt: the number of the element where the spot diagram is to be computed. In addition, the user is asked to select which output coordinates to use to define the x and y axes. The choices are Tout, Enter or Beam (see Section 4). SPOt plots an x-y scatter format unless plot types TEXt, BINary, or MATlab are selected, in which case a ".spot" file is written to disk. Figures 4 and 36 show spot diagrams for Cassegrain.in.

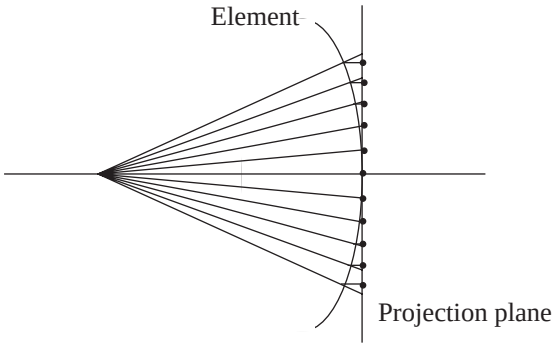


FIGURE 35 Spot on the projection plane

```
MACOS>spot
Enter element where spot diagram is to be computed [0]:4
Enter output coordinate option (Tout, Enter or Beam): [Tout]:
Tracing 150 rays...
Compute time was 0.3086 sec
Graphics device/type (? to see list): /xw
      Chief ray location: x= 0.0000000D+00 y= 0.0000000D+00
Centroid of spot diagram: x=-5.2080625E-18 y= 0.0000000E+00
MACOS>
```

5.2.7 OBSCURATION

By default, the SPOt and OPD commands do not plot rays which are obscured or vignetted by obscurations or apertures on the current or previous elements. The OBScuration command is used to change this default.

OBS has three settings. OBS=ALL plots *every* ray -- even rays that are obscured. OBS=POSITIVE, the default, plots only rays that are *not* obscured on the current or on any previous element. OBS=NEGATIVE plots only those rays that *are* obscured some-

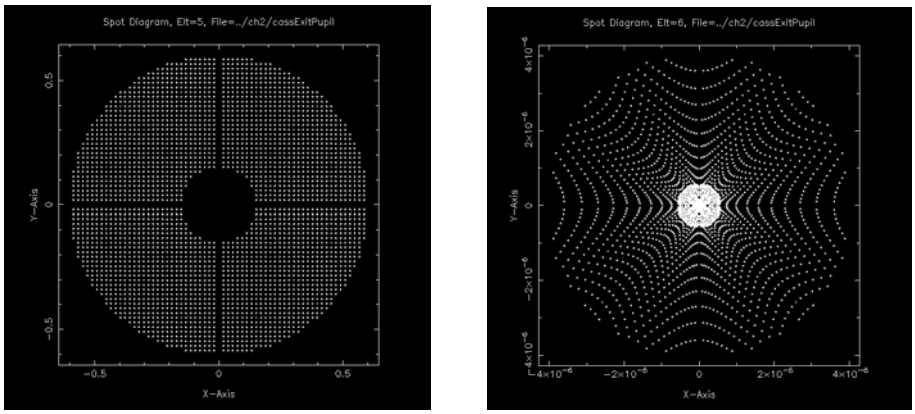


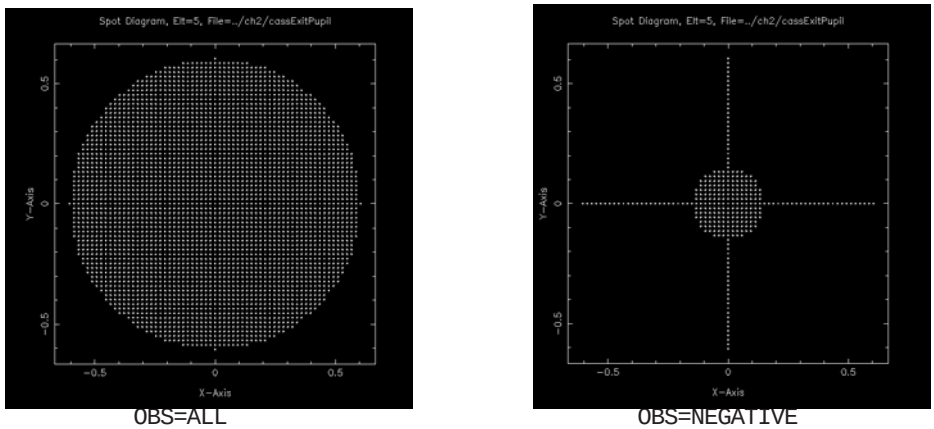
FIGURE 36 `Exit pupil and detector SPOt` diagram for `Cassegrain.in`.

where upstream of the specified element. Of course, rays that become undefined upstream are not plotted under any of these options.

Figure 37 shows spot diagrams of the Cassegrain telescope, for comparison with Figure 36. The spot diagram on the left is plotted with `OBS=ALL`, showing all rays. The spot diagram on the right is plotted with `OBS=NEGATIVE`, showing only obscured rays.

MACOS>**spot**

```
Enter element where spot diagram is to be computed [4]:5
Enter output coordinate option (Tout, Enter or Beam): [Tout]:
Tracing 150 rays...
Compute time was 0.1211 sec
      Chief ray location: x= 0.0000000D+00 y= 0.0000000D+00
Centroid of spot diagram: x=-1.1055660E-16 y=-1.1055660E-16
Type <RETURN> for next page:
MACOS>obs
 Enter ray-trace obscuration option (ALL, POSITIVE, NEGATIVE) [POSITIVE]: ALL
 All rays plotted on spot diagrams
MACOS>spot
Enter element where spot diagram is to be computed [5]:5
Enter output coordinate option (Tout, Enter or Beam): [Tout]:
      Chief ray location: x= 0.0000000D+00 y= 0.0000000D+00
Centroid of spot diagram: x= 7.1761210E-17 y= 7.1761210E-17
Type <RETURN> for next page:
MACOS>
```



OBS=ALL OBS=NEGATIVE

FIGURE 37 Exit pupil Spot diagrams with alternate `OBS` settings

5.3 Beam Set-Up Commands

The next several commands are designed primarily to help the user correctly set up particular computations, such as computation of beam wavefront error, or diffraction computations. These commands help the user implement perturbations of the optical train to evaluate performance impacts of misalignments, different field angles, or "zoom" settings. The STOP command specifies the system stop, either at an element or at a point in global coordinates, to help in defining the beam. It is required by other commands, such as PERTURB, FFP, PFP and FEX, that might move the source. The CENTER command finds source positions that cause the beam to be centered at a specified element. The FFP and PFP commands find source field angles that place the beam at a particular point on a particular surface, such as on a detector. The FEX command finds the system exit pupil and places a return surface at that point. These commands, together with the ORS and SRS commands discussed in the next section, provide a full capability for correctly setting up most ray-trace and diffraction analyses.

5.3.1 STOP

The lateral position of a beam propagating through a system is constrained by the presence of a system stop. The system stop is an aperture somewhere in the system through which the beam passes. It defines the beam position and (usually its shape), as per Figure 38. Certain commands (PERturb, FFP, PFP and FEX) require knowledge of the stop position to enforce the correct beam geometry.

The system stop is specified using the STOp command. STOp does not center the beam in the stop (CENTEr can be used to do this). It merely defines the system stop position. This information is used to constrain the beam when setting exit pupils or perturbing the source.

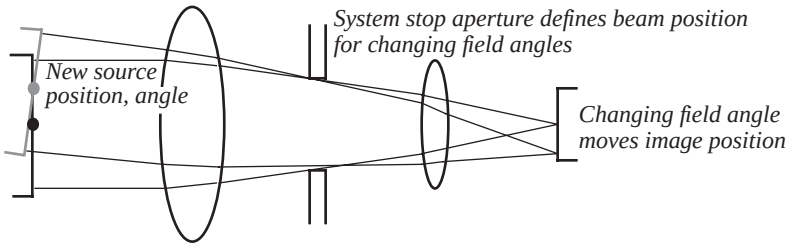


FIGURE 38 System stop

STOp has 2 or 3 arguments, depending on whether the stop is to be specified at an element or at a point in object space. The first argument is a character string specifying whether the stop is to be tied to a particular element (ELT) or to a particular point in object space (OBJ).

If the former, the next parameter requested is the element number. This is followed by a request for an offset vector: a 2-vector in local beam coordinates giving the center of the stop with respect to the vertex of the element (VptElt).

If the stop is to be specified in object space, the next request is for the 3-vector coordinates of the stop position.

The following examples illustrate the setting of stops for the Cassegrain.in example prescription. The first example sets the stop at the nominal location in 3-space of the primary mirror vertex:

```
MACOS>stop
Stop at ELT or OBJect point?  [OBJ]: object
Enter stop position in object space (x,y,z): [0.,0.,0.]:0,0,0
MACOS>
```

The second example puts the stop at the secondary mirror. The STOp command computes the equivalent object-space stop position:

```
MACOS>stop
Stop at ELT or OBJect point?  [OBJ]: elt
Enter system stop element number: [1]:3
Enter offset vector (dx,dy): [0.,0.]:0,0
Computed StopPos=  0.000000000D+00  0.000000000D+00  1.637981902D+01
MACOS>
```

5.3.2 CENTER

The CENTEr command solves for a new source position that places the beam at a specified point on a specified element. This command is useful for enforcing the system stop; it has other applications as well. This command has two arguments: the element number of the element where the beam is to be placed; and the offset of the center of the beam from the vertex of the element. The CENTEr comand changes the source chief ray position ChfRayPos but not the field angle.

The following example uses the Cassegrain.in example to center a perturbed beam, with the result shown by tracing the chief ray to the specified element:

```
MACOS>center
Enter system stop element number: [1]:2
Enter offset vector (dx,dy): [0.,0.]:0,0
Beam centered in   2 iterations, error=  3.231486333D-15
MACOS>ray
Enter number of ray (1=chief ray, 0=quit): [0]: 1
Ray     1 is the chief ray.
Ray    1 segment from Element    0 (InputRay) to Element    1 (SecMirOb):
  Starting point=  6.055112155D-17  5.000007357D-03 -5.000005690D+00
       End point=  6.055112155D-17  4.200001400D-03 -4.200000000D+00
       Direction=  0.000000000D+00 -9.999998333D-04  9.999995000D-01
  Segment Length=  8.000060899D-01
    Total Length=  8.000060899D-01
      Index(i-1)=  1.000000000D+00  0.000000000D+00
        Index(i)=  1.000000000D+00  0.000000000D+00

Ray     1 is obscured by element  1
Ray    1 segment from Element    1 (SecMirOb) to Element    2 (Primary ):
  Starting point=  6.055112155D-17  4.200001400D-03 -4.200000000D+00
       End point=  6.055112155D-17 -3.230922474D-15  0.000000000D+00       Zeroed!
       Direction=  0.000000000D+00 -9.999998333D-04  9.999995000D-01
  Segment Length=  4.200002100D+00
    Total Length=  5.000008190D+00
      Index(i-1)=  1.000000000D+00  0.000000000D+00
        Index(i)=  1.000000000D+00  0.000000000D+00
(etc...)
```

5.3.3 COORD

The COOrdinates command computes beam coordinates at any specified element. Beam coordinates are the projection onto that element of the xGrid and yGrid axes defined at the source element (see previous section). The COORDinates command takes a single argument: the element number iElt. It prints out two orthogonal unit vectors in

response. These vectors are the projection onto the surface at the chief ray of the xGrid and yGrid axes.

These coordinates are computed using four differential chief rays. The first two differ slightly in x- and y-*angle* from the nominal chief ray. In the near-field of the source, such as at a pupil element, the incidence points of these rays on the element differ enough to indicate the projected axes directions. The second two differential chief rays differ in x- and y-*position* from the nominal chief ray. They are used to indicate axes direction in the far-field of the beam, such as at a focal plane.

Note: The beam coordinates may undergo a sign change in the vicinity of a focus. The unit vectors will be correct and consistent, however, their signs may abruptly flip. Use common sense in interpreting the results of the COORD command!

The following example uses the Cassegrain.in example, determining the coordinates projection onto the secondary obscurations (iElt=1), the exit pupil (iElt=5), and the detector (iElt=6):

```
MACOS>coord
Enter element number: [1]:1
Computed xLocal= -1.000000000D+00  0.000000000D+00  0.000000000D+00
Computed yLocal=  0.000000000D+00 -1.000000000D+00  0.000000000D+00
Computed zLocal=  0.000000000D+00  0.000000000D+00  1.000000000D+00
MACOS>coord 5
Enter element number: [1]: 5
Computed xLocal= -9.999999998D-01  0.000000000D+00  2.229180881D-05
Computed yLocal=  0.000000000D+00 -9.999999998D-01  2.229180881D-05
Computed zLocal=  2.229180880D-05  2.229180880D-05  9.999999995D-01
MACOS>coord 6
Enter element number: [1]: 6
Computed xLocal= -1.000000000D+00  0.000000000D+00  0.000000000D+00
Computed yLocal=  0.000000000D+00 -1.000000000D+00  0.000000000D+00
Computed zLocal=  0.000000000D+00  0.000000000D+00  1.000000000D+00
MACOS>pert 6
```

5.3.4 FFP

The find field point (FFP) command finds the field angles for which the chief ray intersects a specified element at a specified point. It resets both the source position and angle (ChfRayAng and ChfRayPos) to both satisfy the system stop condition and to place the beam at the given point.

The FFP command has two arguments: the element number of the surface at which the chief ray intercept is specified; and the offset of the chief ray from the vertex of the target element in global coordinates. The STOP command must be used prior to invoking FFP.

The following example uses the Cassegrain.in example, finding the beam that offsets an image by (1e-3, -2e-3):

```
MACOS>stop
Stop at ELT or OBJect point?  [Elt]: obj 0,0,0
Enter stop position in object space (x,y,z): [0.,0.,0.]: 0,0,0
MACOS>ffp 6
Enter element number: [6]:
Enter offset vector in global units (dx,dy): [0.,0.]:1e-3,-2e-3
Field angle finder results:
Did    1 iterations, error=  6.936774704D-11
```

```
 Old dx=  0.000000000D+00  New dx=  1.000000031D-03
          0.000000000D+00          -2.000000062D-03
Old crd=  0.000000000D+00 New crd=  4.458354140D-05
          0.000000000D+00          -8.916708280D-05
          1.000000000D+00           9.999999950D-01
Old crp=  0.000000000D+00 New crp= -2.229177070D-04
          0.000000000D+00           4.458354140D-04
         -5.000000000D+00          -4.999999975D+00
Accept the new chief ray?  [YES]: yes
MACOS>ray 1                                  Trace ray to show new incidence point...
Enter number of ray (1=chief ray, 0=quit): [0]: 1
Ray     1 is the chief ray.
Ray    1 segment from Element    0 (InputRay) to Element    1 (SecMirOb):
  Starting point= -2.229177070D-04  4.458354140D-04 -4.999999975D+00
       End point= -1.872508748D-04  3.745017496D-04 -4.200000000D+00
       Direction=  4.458354140D-05 -8.916708280D-05  9.999999950D-01
  Segment Length=  7.999999791D-01
    Total Length=  7.999999791D-01
      Index(i-1)=  1.000000000D+00  0.000000000D+00
        Index(i)=  1.000000000D+00  0.000000000D+00

(etc...)

Ray    1 segment from Element    5 (ExitPupi) to Element    6 (Detector):
  Starting point=  1.812075347D-04 -3.624150694D-04 -4.060145887D+00
       End point=  1.000000031D-03 -2.000000062D-03  1.500010000D+00
       Direction=  1.472607007D-04 -2.945214014D-04  9.999999458D-01
  Segment Length=  5.560156189D+00
    Total Length=  1.462230217D+01
      Index(i-1)=  1.000000000D+00  0.000000000D+00
        Index(i)=  1.000000000D+00  0.000000000D+00
MACOS>
```

5.3.5 PFP

The pixel field point (PFP) command is the same as the FFP command, in that it finds the field angles for which the chief ray intersects a specified element at a specified point. It differs in that the incidence point is specified in pixel coordinates. PFP resets both the source position and angle (ChfRayAng and ChfRayPos) to both satisfy the system stop condition and to place the beam at the target point.

The PFP command has two arguments: the element number of the surface at which the chief ray intercept is specified; and the offset of the chief ray from the vertex of the target element in global coordinates. The STOP command must be used prior to invoking PFP.

The following example uses the Cassegrain.in example, finding the beam that puts an image at pixel location (20, -100):

```
MACOS>PFP 6
Enter element number: [6]:6
Enter pixel size for placing image on pixel array: [0.]:15e-4
Enter image position in pixel units (dx,dy): [0.,0.]:20,-100
Field angle finder results:
Did    2 iterations, error=  1.581159622D-12
 Old dx=  1.000000031D-03  New dx=  3.000000000D-02
         -2.000000062D-03          -1.500000000D-01
Old crd=  4.458354140D-05 New crd=  1.337301749D-03
         -8.916708280D-05          -6.686508743D-03
          9.999999950D-01           9.999767508D-01
Old crp= -2.229177070D-04 New crp= -6.686508743D-03
          4.458354140D-04           3.343254372D-02
         -4.999999975D+00          -4.999883754D+00
Accept the new chief ray?  [YES]:
```

```
MACOS>ray 1                                    Trace ray to show new incidence point...
Enter number of ray (1=chief ray, 0=quit): [0]: 1
Ray     1 is the chief ray.
Ray    1 segment from Element    0 (InputRay) to Element    1 (SecMirOb):
  Starting point= -6.686508743D-03  3.343254372D-02 -4.999883754D+00
       End point= -5.616797930D-03  2.808398965D-02 -4.200000000D+00
       Direction=  1.337301749D-03 -6.686508743D-03  9.999767508D-01
  Segment Length=  7.999023513D-01
    Total Length=  7.999023513D-01
      Index(i-1)=  1.000000000D+00  0.000000000D+00
        Index(i)=  1.000000000D+00  0.000000000D+00

(etc...)

 Ray    1 segment from Element    5 (ExitPupi) to Element    6 (Detector):
  Starting point=  5.436452666D-03 -2.718226333D-02 -4.060076800D+00
       End point=  3.000000000D-02 -1.500000000D-01  1.500010000D+00
       Direction=  4.416714741D-03 -2.208357371D-02  9.997463720D-01
  Segment Length=  5.561497351D+00
    Total Length=  1.462402454D+01
      Index(i-1)=  1.000000000D+00  0.000000000D+00
        Index(i)=  1.000000000D+00  0.000000000D+00
MACOS>
```

5.3.6 FEX

The Find EXit pupil (FEX) command computes the location of the system exit pupil and places an optimized spherical reference surface at that point. By "optimized," we mean that the spherical reference surface is correctly set up to perform a far-field diffraction propagation to the next element, which is usually a detector.

To use FEX, the prescription must be set up with two return surfaces followed by a detector focal plane or reference surface. The first return surface is placed at the location of the focal plane detector, and the second is placed in the vicinity of the exit pupil. FEX locates the second precisely at the exit pupil location and computes its orientation and curvature based on the position of the detector element.

FEX works for systems where the exit pupil is located in front of the detector. It is not useful for telecentric systems (where the exit pupil is at infinity) or for cases where the exit pupil is located behind the detector.

FEX traces two rays: the nominal chief ray and a differential chief ray. The field angle of differential chief ray differs by 1 arcsecond from the nominal chief ray angle. Both rays are traced to the detector return surface and then backwards to find their intersection. The intersection of the nominal and differential chief rays defines the center of the exit pupil. The parameters of the exit pupil (the second return surface) are set as follows:

- vertex, VptElt, to the center of the exit pupil
- radius of curvature, KrElt, to the distance between the exit pupil and detector
- principal axis, PsiElt, to the chief ray direction, ChfRayDir

The result is a reference sphere aligned with the spherical wavefront converging to the point where the chief ray hits the detector. This configuration correctly models diffraction images at that point field point. If the optical system is perturbed, the FEX command must be rerun.

FEX requires two arguments: the number `iElt` of the exit pupil return surface and the location in global coordinates of the vertex of the aperture stop.

For example, we added two return surfaces to the `Cassegrain.in` prescription just before the detector using a text editor (see Section 4.9). The first is a return surface at the detector location. The second is close to the exit pupil. The entire .in-file is in Appendix A.1.4.

```
 MACOS>show 5
 Enter number of element (0=aperture): [0]: 5

    iElt=    5
 EltName= rtn1
 Element= Return
 Surface= Conic
    fElt=   1.000000000D+22
    eElt=   0.000000000D+00
   KrElt=  -1.000000000D+22
   KcElt=   0.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00  1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00  1.500000000D+00
  RptElt=   0.000000000D+00  0.000000000D+00  1.500000000D+00
  IndRef=   1.000000000D+00
  Extinc=   0.000000000D+00
    nObs=    0
  ApType=   None
    zElt=   0.000000000D+00
PropType=   Geometric
 nECoord=   -6
nECoord=   -6
 MACOS>show 6
 Enter number of element (0=aperture): [0]: 6

    iElt=    6
 EltName= rtn2
 Element= Return
 Surface= Conic
    fElt=   6.000000000D+00
    eElt=   0.000000000D+00
   KrElt=  -6.000000000D+00
   KcElt=   0.000000000D+00
  psiElt=   0.000000000D+00  0.000000000D+00  1.000000000D+00
  VptElt=   0.000000000D+00  0.000000000D+00 -4.500000000D+00
  RptElt=   0.000000000D+00  0.000000000D+00 -4.500000000D+00
  IndRef=   1.000000000D+00
  Extinc=   0.000000000D+00
    nObs=    0
  ApType=   None
    zElt=   6.000000000D+00
PropType=   Geometric
 nECoord=   -6
```

Then the exit pupil is optimized:

```
 MACOS>fex
 Enter number of exit pupil return surface: [5]:6
  Tracing   173 rays...
              Chief ray location: x= 0.0000000D+00 y= 0.0000000D+00 z=-
4.5000000D+00
               Centroid location: x= 7.9797280D-17 y= 8.6736174D-17 z=-
4.4875916D+00
  Centroid offset from chief ray: x=-7.9797280D-17 y=-8.6736174D-17 z=-
1.2408352D-02
 Exit pupil finder results:
   Old f=  6.000000000D+00   New f=  6.790667844D+00
```

```
  Old z=  6.000000000D+00   New z=  6.790667844D+00
Old psi=  0.000000000D+00 New psi=  9.936451891D-17
          0.000000000D+00           1.080049119D-16
          1.000000000D+00           1.000000000D+00
Old Vpt=  0.000000000D+00 New Vpt=  0.000000000D+00
          0.000000000D+00           0.000000000D+00
         -4.500000000D+00          -5.290667844D+00
Accept the new element?  [YES]:
MACOS>
```

5.3.7 PERTURB

The PERturb command allows the user to change the current optical prescription by rotating or translating the light source or any of the optical elements. PERturb accepts as arguments rotation and translation perturbation vectors, which it applies directly to the prescription data. Rotations of an element are taken about the prescription rotation point RptElt. Subsequent ray traces or propagations will use the new prescription.

PERturb differs from MODify in that it changes only the position and orientation of elements (MODIFY can be used to change any parameter in the prescription). PERturb is also different from MODify in that PERturb uses the rotation point and proper kinematics to apply rotational perturbations (PERturb rotations are relative to the old orientation) and MODify specifies a absolute location. PERturb is not limited to small motions.

To reset the prescription to its unperturbed condition, it is necessary to explicitly reverse the PERturb commands executed previously or to reload the prescription using the OLD command. Due to the noncommutativity of rotations, the latter method is to be preferred if more than one rotation has been applied.

The PERturb command takes arguments specifying the element to be perturbed and the perturbation vectors. The element is specified by number iElt which is in the range: 0 (the source) to nElt (the number of elements in the active prescription). Units for rotation perturbations are radians. The units of the translation perturbations are in the chosen base units (meters, inches, whatever).

Rotations are applied using Euler axis/angle parameters (Ref. Hughes). The axis of the rotation is given by the direction of the rotation perturbation vector. The angle is given by its magnitude. This approach is used in preference to Euler angles or quaternions as it provides a smooth transition to the usual small-angle linear approximations.

If iElt=0, indicating the source is to be perturbed, the user is prompted for the location of the aperture stop in global coordinates. For instance to look 0.5 deg (8.72664d-3 rad) off the x-axis, a rotation about the y-axis is commanded:

```
MACOS>perturb
Enter element to be perturbed: [0]:0
Enter rotational perturbation vector (x,y,z): [0.,0.,0.]:0,8.72664d-3,0
Enter translational perturbation vector (x,y,z): [0.,0.,0.]:0,0,0
Computing new perturbed system parameters
MACOS>
```

If iElt>0, indicating a regular element, the user is asked whether the perturbation vectors are in local element coordinates or in global coordinates. If in local coordinates, the user is prompted for a single 6-vector perturbation in those coordinates. If in global

coordinates, the user is prompted for a rotational perturbation 3-vector, then a translational 3-vector. For instance, to translate the secondary mirror in Cassegrain.in:

```
MACOS>pert
Enter element to be perturbed:3
Enter coordinate system for perturbation (ELEMENT or GLOBAL):  [GLOBAL]:
Enter rotational perturbation vector (x,y,z): [0.,0.,0.]:0,0,0
Enter translational perturbation vector (x,y,z): [0.,0.,0.]:0,0,1d-6
Computing new perturbed system parameters
MACOS>
```

5.3.8 PREAD

The PREad command reads element perturbations from a specified datafile. MACOS assumes that the name of the perturbation file is infilename.pert. If this file does not exist, MACOS prompts for ther perturbation file name.

The perturbation file must be in the following all numeric format. The perturbations must be in global coordinates. The order of the perturbation vectors in the .pert-file must be identical to the order of the elements (the first perturbation vector corresponds to the first element, the sixth perturbation vector corresponds with the sixth element). Each element must have a perturbation vector. If an element is unperturbed, then a string of zeroes should be entered as its perturbation vector. The perturbation vectors are rotational perturbations and translational perturbations. The 6 components of the perturbation vector must be separated by spaces.

```
MACOS>pre
File Cassegrain.pert read.
MACOS>
```

5.3.9 ATMOS

MACOS provides an atmospheric phase disturbance function with the ATMos command. ATMos imposes a ray pathlength and direction perturbation on the beam at a specified element. The ray perturbations are derived from a Kolmogorov phase disturbance model, parameterized by the atmospheric turbulence coherence length r_0, and driven by a random number generator. The seed for the random number generator can be set using the SEED command.

ATMos provides two options for specifying the atmospheric grid. The RAY option treats each ray as an independant but correlated random variable, determining the phase disturbance (and so the direction and OPD) for each ray directly from the Kolmogorov generating function. The ATM option implicitly defines a coarser grid of "atmospheric states" than the ray grid provides. Essentially, the phase disturbance is determined for cells whose size is specified by the user, but which will be in general larger than the ray grid cell size. Ray direction and OPD values are computed by interpolating from the surface defined by the atmospheric phase states to the ray grid cells.

ATMOS takes four or five arguments, depending on how the grid is to be specified (RAY or ATM). An example of the use of ATMOS in an adaptive optics system model is provided in Appendix A.4. A sample dialog showing use of ATMOS follows:

```
MACOS>atm
Enter number of atmosphere phase screen element: [0]:1
Enter atmosphere r0: [12.4492]: 24
Enter atmosphere wavelength: [2.854331E-05]: 2.854331E-05
Enter atmosphere tilt participation (0-1): [1.00000]:0
Use RAY grid or separate ATMospheric grid?  [RAY]: atm
Computed xLocal=  0.000000000D+00 -1.000000000D+00  0.000000000D+00
Computed yLocal= -1.000000000D+00  0.000000000D+00  0.000000000D+00
Computed zLocal=  0.000000000D+00  0.000000000D+00 -1.000000000D+00
```

```
Tracing  12113 rays...
Compute time was    0.2266 sec
Enter grid spacing: [1.56840]:20
MACOS>
```

Here the “tilt participation” regulates the percentage of the Kolmogorov that is to be included in the phase screen. Tilt is explicitly removed when this factor is set to zero, and increases proportionally as the factor goes to 1, which signifies full Kolmogorov tilt. An example, showing a typical atmospheric wavefront from AOexample.jou (Appendix A.4), is shown in Figure 39.

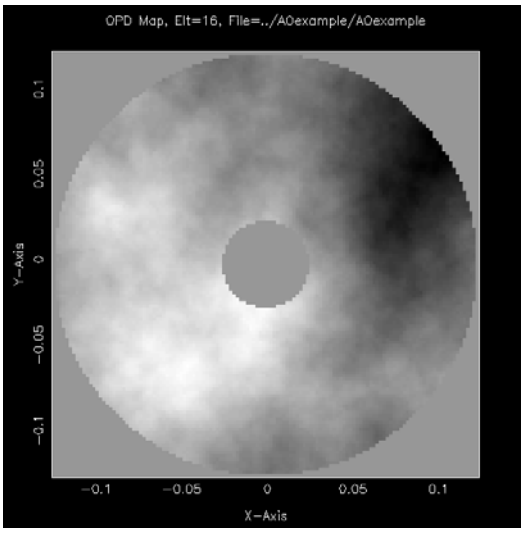


FIGURE 39 Atmospheric disturbances (AOexample).

5.3.10 POLARIZATION

The `POLarization` command turns on polarization ray tracing. The default is no polarization ray tracing.

5.3.11 NOPOLARIZATION

The `NOPOlarization` command turns off polarization ray tracing. This is the default setting.

5.3.12 SINT

The Surface INTerpolate (`SINT`) command activates the surface interpolation of elements. Prior to using the `SINT` command, the surface is assumed to be a conic-of-revolution. MACOS assumes that the interpolation input file has the name `in-file.srf#`, where # is the number of the element the data describes. If that file does not exist, MACOS prompts the user for the name of the file.

```
MACOS>SINT
Computing interpolated surface data...
Compute time was    5.7871 sec
MACOS>
```

SECTION 6 Diffraction Analysis

This section presents commands for physical optics modeling. Most optical systems are adequately modeled using a single-plane, exit-pupil diffraction model. Other systems are better modeled using multiple planes of diffraction. This section describes how to set up .in-files and other necessary data for both cases. The commands discussed in this section are:

ORS	SRS
BEAm	REGrid
SCAlar	VECtor

The diffraction modeling routines of MACOS can accurately compute both near-field (Fresnel diffraction) and far-field (Fraunhoffer diffraction) effects. Roughly speaking, near-field propagation takes the beam from pupil to pupil, regions where the beam is large compared to a wavelength. Far-field propagations take the beam from a pupil to a focal point, or from a focal point to a pupil. In the vicinity of a focus, the beam diameter approaches a wavelength.

Examples of systems where near-field diffraction is important include laser beam transfer systems, or spatial filters, where the beam propagates from pupil to pupil via a far-field plane where an aperture or obscruation is placed. Examples of far-field diffraction include imaging systems and high-gain directional antennas. In some cases both types of diffraction are important, and multiple planes of diffraction should be used.

6.1 Single- versus Multi-plane Diffraction

The first step in setting up a problem is to decide whether the important diffraction effects can be captured using a single-plane model, or whether multi-plane diffraction is neccessary. Most imaging systems are adequately modeled using the single-plane approach, with one far-field diffraction being used to create the image. Section 6.2.2 discusses the set up of the .in-file for a single far-field propagation.

Systems for which diffraction effects of beam truncation due to apertures or obscurations may be significant may be best modeled using multiple planes of diffraction. Examples are systems with spatial filters and coronographic systems. Examples of systems requiring near-field propagation, whether multi-plane or no, include telecentric systems, defocused systems, or any system for which intermediate-plane intensities are desired.

If a multi-plane model is necessary, the first step is to lay out the optical system and determine the sequence of propagations. Consider the Cassegrain telescope with relay optics (including a fold near the focus of the main telescope) shown in Figure 40. Figure 41 shows reference surfaces added for multiplane diffraction calculations. The beam is partially obscured at the main and relay telescope secondary mirror supports (“spiders”). There are 4 diffraction propagations which are clearly shown in the diffraction schematic, Figure 42. Note that points A and D are used in the model twice. The first time each is used to model the spider obscurations. The second time, the points are used as a reference surfaces for diffraction calculations.

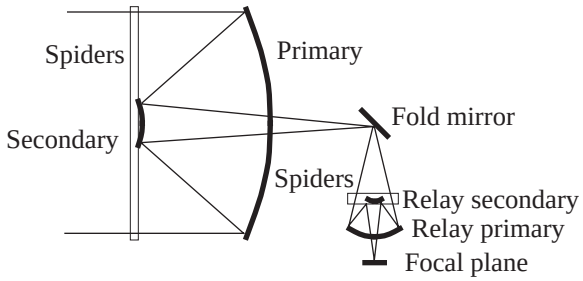


FIGURE 40 Optical schematic of Cassegrain telescope and relay optics example

The first near-field propagation, `PropType=5,` occurs between the main telescope pri-

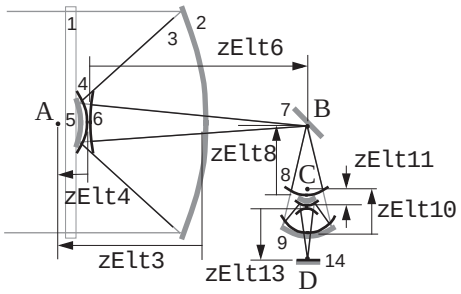


FIGURE 41 Reference surfaces for multi-plane diffraction in Cassegrain telescope and relay optics example

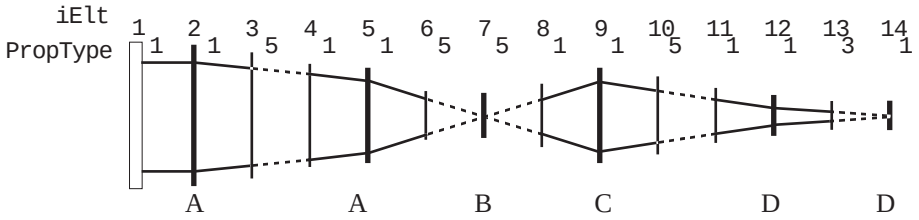


FIGURE 42 Diffraction schematic for Cassegrain telescope and relay optics example

mary and secondary mirror (between reference surfaces 3 and 4 which are located close to these optics). The reference surfaces share a common center of curvature at point A. Since this is a converging beam, the `zElt` parameters are positive for both reference surfaces (see Section 6.1.4).

The second near-field propagation occurs between the secondary mirror and the relay camera's secondary mirror support "spiders." It is a "through-focus" propagation, with `zElt=6` positive (converging beam) and `zElt=8` negative (diverging beam). The reference point for this propagation is point B. The fold mirror is included in the propagation.

The third propagation is at the converging beam between the relay primary and secondary. This near-field propagation is centered at point C.

The last propagation is a far-field propagation, PropType=3, to the focal plane. It is centered at the focal plane, point D.

6.2 Setting up Diffraction Propagations

To use MACOS diffraction features, the .in-file prescription must be set up correctly. A surface which does not change the wavefront (reference, obscuring, return, or focal plane elements) must ideally be placed at the start and end of each diffraction propagation. Each pair of reference surfaces must be defined together—*slaved* to each other—so as to avoid spurious phase errors. Slaving reference surfaces to each other means ensuring that they share a common axis and a common center of curvature. The reference surfaces should be *optimized* to minimize sampling errors. The beam must be adequately *padded* to avoid aliasing. In certain conditions, the rays that are used to determine geometric phase should be *regridded*.

MACOS provides several commands to help ensure that the prescription data is set up correctly. The FEX command (discussed earlier in Section 5.3.6) optimizes placement of a return surface at an exit pupil for far-field diffraction propagation. The ORS command optimizes a reference surface for minimum sampling error by matching it to the best-fit sphere and aligning it to the chief ray. The SRS command slaves one reference surface to another, solving for location, orientation, and radius of the starting (or ending) surface of a near- or far-field diffraction propagation as a function of the parameters of the ending (or starting) surface. Used together, FEX, ORS and SRS enable the correct setup of any MACOS diffraction propagation.

The FEX, ORS and SRS commands are run after a .in-file prescription is loaded. They *refine* the prescription for elements defined in the .in-file. The results can be saved in a new .in-file using the SAVe command. If a PERturb, MODify, FFP, PFP, CENTER, or other command changes the source position or angle, or the position, angle or figure of an element in the propagation path, FEX, ORS and SRS should be rerun.

The results of propagation commands (such as INtensity or PIXilate) are stored in particular internal MACOS arrays. These arrays are accessible through SMACOS common statements and the SMACOS call line. These include the ray-trace data variables defined in Section 5.1. Additional diffraction-specific arrays include:

- WFElt(mdttl,mdttl,mWF) holds complex amplitude matrices that store the results of propagation commands (such as INtensity). Here mdttl is the maximum dimension of the "wavefront" amplitude arrays, and is defined in the paramXXX.cmn include files.
- PixArray(mPix,mPix) stores the results of a pixilated image generation command (such as PIXilate or ADD). Here mPix is defined in the paramXXX.cmn include files.

MACOS implicitly enforces a padding requirement of at least 50% at compile time through internal data array declaration statements. This is accomplished by restricting the ray grid, which defines the maximum extent of the ray beam, to occupy the central half of the WFElt(mdttl,mdttl,mWF) matrices (see Section 2.3).

Once the surfaces for the diffraction calculations are properly set up, the beam can be propagated through the system by invoking one of the propagation commands described in this section (such as INtensity). These commands use geometric and Fresnel propa-

gators to propagate the complex amplitude matrix to the specified element. They then plot the appropriate data product.

We list here the available diffraction propagators (Table 13).

TABLE 13 MACOS diffraction propagators

PropType=...	Description
`Geometric`	Geometric propagator, updates phase only as $e^{i\phi}$
`FarField`	Fresnel/Fraunhaufer far-field propagator: sphere- or plane- to-plane
`NFSpherical`	Fresnel near-field propagator: sphere- to-sphere
`NFPlane`	Fresnel near-field propagator: plane- to-plane
`NF1`	Part 1 of the near-field propagator: sphere- to-plane. Use when an obscuration is to be imposed at an intermediate focus
`NF2`	Part 2 of the near-field propagator: plane-to-sphere
`GeomUpdate`	Geometric propagator, forces update at each surface (don't use)
`NFS1surf`	Near-field propagator: sphere- to-sphere, using end ref surf only
`NFP1surf`	Near-field propagator: plane- to-plane, using end ref surf only
`SpatialFilter`	Near-field propagator: sphere- to-sphere, with spatial filter
`SF1surf`	Near-field propagator with spatial filter, using end ref surf only

6.2.1 Choosing Propagator Type

For imaging systems, it is usually clear whether a particular diffraction propagation is best modeled using the near-field or far-field propagator. If the light is to be evaluated at a focus, the far-field propagator should be used. Otherwise, the near-field propagator should be used. For far-field propagations to focal planes, it is important to place the focal plane element at the physical detector location, which may not coincide with the center of curvature of the exit pupil, which is to say, the system might be defocused (see Figure 43).

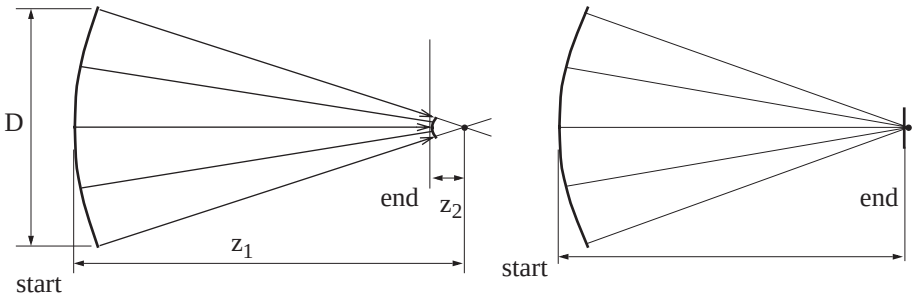


FIGURE 43 Imaging a defocused beam

Some systems require near-field propagation to model image formation. For example, spacecraft fine guidance sensors are defocused to spread a star image over multiple pixels to improve centroiding accuracies (see Figure 43). These images can be accurately generated using the near-field propagators followed by a geometric propagation to the focal plane. The intensity in each pixel can then computed using the `PIXilate` command described in Section 7.2.2.

Other conditions requiring near-field propagations include the presence of a spatial filter, occulting mask, or other intensity-altering object at an intermediate far-field point in the beam.

In some cases the choice of propagator will not be clear. Evaluating the Fresnel number F_n may help determine the choice. Consider the defocused beam of Figure 43. Here the first reference surface has diameter D and axial distance, i.e distance from the center of curvature, z_1. The second reference surface is located at an axial distance z_2. The Fresnel number F_n is

$$F_n = \frac{D^2}{\lambda(z_2 - z_1)} \quad \textbf{(EQ 1)}$$

Both far- and near-field propagators should be tested when F_n is in the range (0.1 – 1.0).

Obvious aliasing effects may also be mitigated by changing the sampling of the pupil, by increasing the diffractiongrid size, the ray grid size, or both. This may require using larger MACOS executable sizes (e.g., `macos512` if aliasing is seen with `macos256`), as well as larger values for `nGridPts`.

6.2.2 Far-Field

Consider the two cases of far-field diffraction shown in Figure 44. Both are typical of systems where a far-field diffraction to a focal plane is required to compute an image or point-spread function. This may be the only diffraction computation in the system (a single-plane model) or it may be the last of several propagations in a multi-plane model.

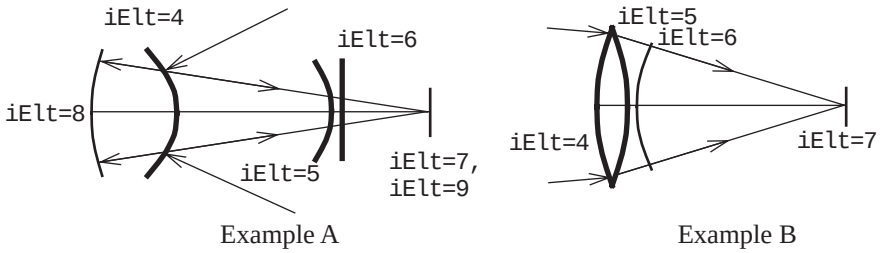


FIGURE 44 Far-field propagation examples

Example A in Figure 44 is set up for *exit pupil diffraction*. The light is incident on element 4 (the secondary mirror in some reimaging optics) and is reflected to two refractive surfaces (elements 5 and 6) which together form a lens. Finally, the light forms an image on element 7. For exit pupil diffraction, MACOS needs two additional surfaces in the model: elements 8 and 9. Elements 7 and 8 must be *return surfaces* with element 7 located at the focal plane and element 8 located at the exit pupil. From element 7, MACOS traces light backwards in image space (this is the way MACOS performs the calcualtion, light does not actually propagate in this fashion) until it reaches the exit pupil (element 8). The path length of this segment is negative for the purposes of geometric propagation. It then reverses direction until reaching the focal plane (element 9). Element 9 must be a *focal plane surface*.

The exit pupil surface must be a sphere with radius of curvature equal to the distance from the exit pupil to the image on the focal plane. The FEX command can be used to setup the exit pupil location, principal axis, and radius of curvature based on the chief ray. The Fresnel propagation distance for the exit pupil, zElt, must be set to the radius of curvature of the exit pupil element.

NOTE: The principal axis of the exit pupil should be set along the line connecting the center of the exit pupil and the center of the image. The FEX command sets it along the chief ray which is not the same for systems with a large amount of odd aberrations, such as coma. For some applications, there can be a significant difference.

We assign the geometric propagator (PropType=1) to all elements, except for the exit pupil (element 8) which is assigned the far-field propagator (PropType=3). This directs MACOS to use the far-field propagator to compute the amplitude matrix at the focal plane, based on the phases calculated at the exit pupil by the geometric propagator.

The main advantage of using the exit pupil for far-field diffraction is that the ray grid is usually well-formed there. If a system is significantly aberrated, sampling at non-pupil planes may give distorted ray grids which causes a poor match between the locations of the ray grid and the locations of the complex amplitude matrix grid used in the diffraction calculations. However if the system is well corrected and the ray grid is regular, a different plane may be used. A spot diagram at the plane provides a good qualitative check. Another check is provided by the propagation diffraction summary, which gives information on the “dx” or ray spacing across the reference surface.

Example B illustrates a system that uses a reference surface located away from the exit pupil to mark the start of a far-field diffraction calculation. Two fewer surfaces are required in this case, as the reference surface is placed in the beam between the final optical element (element 6) and the focal plane. All elements are assigned PropType=1, except the reference surface which is assigned PropType=3 to mark the start of the far-field propagation. Again, the reference surface is be a sphere with radius of curvature equal to the distance from the reference surface to the point of incidence of the chief ray with the focal plane. The SRS command can be used to find the orientation and radius for the reference surface.

6.2.3 Near-Field

There are basically four cases of near-field diffraction (see Figure 45). The algorithm uses planar reference surfaces. The first three cases deal with converging/diverging beams and use spherical reference surfaces. They employ the angular spectrum propagation algorithm for the paraxial equation and the Sziklas/Seigman algorithm for spherical wavefronts.The fourth case, nearly collimated light, uses the propagation algorithm based on the standard form angular spectrum propagation for the paraxial wave equation(see, e.g. Goodman [2])

The first three types are distinguished by the signs of the Fresnel propagation distance z to the *reference point*, the center of curvature of *both* of the reference spheres. z is positive when the light is moving towards the reference point, as in converging beam. In a diverging beam, the light is moving away from the reference point and z is negative. Both z’s are infinite for a collimated beam.

WARNING: In aberrated systems, the ray grid can lose its regularity. This is likely to be more pronounced near to focus, but can occur anywhere. The grid regularity can be restored using the REGrid command described in Section 6.3.3. In some cases this should be accompanied by an OPD interpolation step, which is not included in this release of MACOS.

Table 14 list the values for reference parameters for near-field propagations. The Fresnel propagation distances are entered both as the zElt parameters for each reference surface and as the focal length, fElt (or the radius of curvature, KrElt) of the element. The sphere-to-sphere propagations are all specified by setting PropType=5 for the first reference surface. The end reference surface has PropType=1, indicating geometric

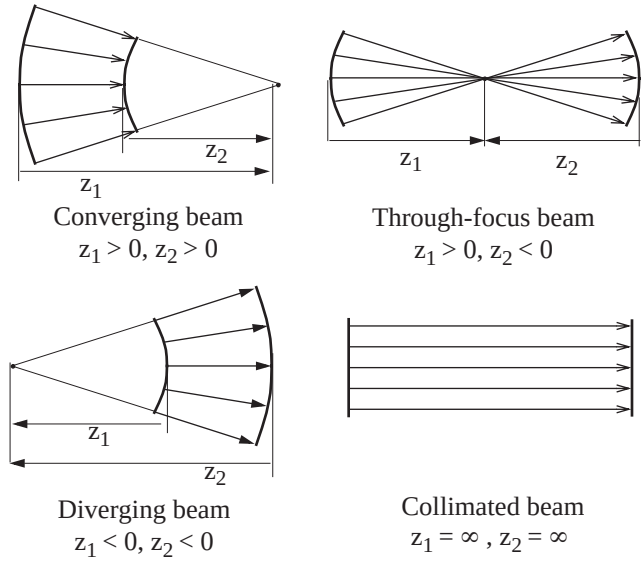


FIGURE 45 Near-field diffraction examples

propagation following the surface. The plane-to-plane propagation has `PropType=4`. (Unpowered elements such as fold mirrors can be included within the scope of a propagation by specifying `PropType=5` or 4.) The last element must have `PropType=1`.

NOTE: `PropType=2` assumes no phase change from the previous element and should be used with extreme caution.

TABLE 14. Reference surface parameters for near-field propagation

Parameter	Referece Surface 1	Reference Surface 2
`zElt`	z_1	z_2
`fElt`	$\|z_1\|$	$\|z_2\|$
`eElt`	0	0
`KrElt`	$-\|z_1\|$	$-\|z_2\|$
`KcElt`	0	0
`PropType`	4,5	1

The locations and orientations of the reference surfaces must be specified.

- Vertex location, `VptElt,` should be close to the optical element at the appropriate extreme of the propagation, but not so close that the surface intersects the optical element which would cause some rays to become undefined.
- The principal axis, `psiElt`, should be set so that the chief ray is normal to the reference surface.

Initial guesses can be refined later with the `ORS` and `SRS` commands.

6.3 Commands

6.3.1 ORS

The Optimize Reference Surface (ORS) command is used to refine the prescription for a reference surface used in near-field diffraction. To use ORS, the prescription must be set up with a reference surface at the desired location (VptElt) and with an approximate principal axis (psiElt). ORS requires one argument: the number iElt of the reference surface to be optimized.

ORS first relocates the reference surface so that the chief ray intersects the vertex point (VptElt). Then ORS tilts the specified reference surface so that psiElt is in alignment with the chief ray. Finally, ORS iteratively solves for the value of fElt that minimizes the OPD over a 15 by 15 grid of rays. The solution is the radius of the reference sphere that best fits the wavefront at that location. This process usually converges quite quickly. If it doesn't, a message may appear indicating a problem. The usual source of difficulty is that the surface has been put in backwards! Try changing the signs of psiElt using the MODify command and repeat.

NOTE: ORS may not find the optimal reference surface for systems with a large amont of odd aberrations, such as coma. In these systems, the minimum OPD is obtained when the reference sphere is tilted from the chief ray.

The ORS command prints the old and new values of the element parameters and asks the user if the new ones are acceptable. Enter yes to accept the new values are for subsequent computations. Enter no to retain the previous values. The new element data will not be kept unless a SAVe command is used to store the data. For example, using the seg.in example in Appendix A.5:

```
MACOS>ors
Enter number of element to be optimized:12
Reference surface optimization results:
 RMSmin=  1.673727619D-06  nCalls= 11
  Old f=  2.450000000D-01   New f= 2.486198802D-01
  Old z=  2.450000000D-01   New z= 2.486198802D-01
Old psi=  2.382657340D-01 New psi= 2.382657340D-01
          0.000000000D+00          0.000000000D+00
          9.712000000D-01          9.712000000D-01
Old Vpt=  7.630588700D-04 New Vpt= 7.630594517D-04
          0.000000000D+00          0.000000000D+00
         -1.495089150D+00         -1.495089150D+00
Accept the new element data? (YES):
MACOS>
```

6.3.2 SRS

The Slave Reference Surface (SRS) command is used to slave one of the two reference surfaces used in a diffraction propagation to the other reference surface. It applies these changes directly to the element prescription. To use SRS, the prescription must be set up with a reference surface at the start and end of the propagation. If the propagation is a far-field propagation, the second surface can be a focal plane. To use SRS, the slaved reference surface must be in the MACOS prescription at the desired location (VptElt) and with an approximate principal axis (psiElt).

SRS is to be used in conjunction with ORS. ORS is run first for the first of the two reference surfaces, then SRS is run for the second. ORS requires two arguments: the number iElt of the reference surface to be slaved and the number iElt to which the reference surface is to be slaved.

SRS first relocates the reference surface so that the chief ray intersects the vertex point (VptElt). Then SRS tilts the specified reference surface so that psiElt is normal to the

surface. Finally, SRS solves for the propagation distance zElt based on the zElt of the surface it is being slaved to and the optical path length between the surfaces along the chief ray. It computes the other element parameters accordingly.

The SRS command prints the old and new values of the element parameters and asks the user if the new ones are acceptable. Enter yes to accept the new values for subsequent computations. Enter no to retain the previous values. The new element data will not be kept unless a SAVe command is used to store the data. Continuing with the seg.in example gives

```
MACOS>srs
Enter number of element to be slaved:13
Enter number of element to slave to:12
Reference surface recalculation results:
  Old f=  1.237919100D-01   New f= 1.274117874D-01
  Old z=  1.237919100D-01   New z= 1.274117874D-01
Old psi=  2.382657340D-01 New psi= 2.382657340D-01
          0.000000000D+00          0.000000000D+00
          9.712000000D-01          9.712000000D-01
Old Vpt= 2.964279300D-02 New Vpt= 2.964279465D-02
         0.000000000D+00          0.000000000D+00
        -1.377371850D+00         -1.377371850D+00
Accept the new element? (YES):
MACOS>
```

6.3.3 REGRID

This feature is undergoing revision and not currently supported.

6.3.4 SCALAR

The SCAlar command sets a toggle to use a scalar formula when calculating single-plane diffraction (see Section 6.1). The SCAlar command has no arguments.

Scalar diffraction is the default and the only mode when polarization ray tracing is off (see Section 5.3.10). Vector diffraction is the default with polarization on.

6.3.5 VECTOR

The VECtor command sets a toggle to use vector formula when calculating single-plane diffraction (see Section 6.1). The VECtor command has no arguments.

Scalar diffraction is the default and the only mode when polarization ray tracing is off (see Section 5.3.11). Vector diffraction is the default with polarization on.

6.3.6 Propagation commands

Several commands, such as INT, LOG, ADD, etc. cause the beam to be propagated to the specified element using MACOS combined ray-trace and diffraction computations. These commands are described in the next section.

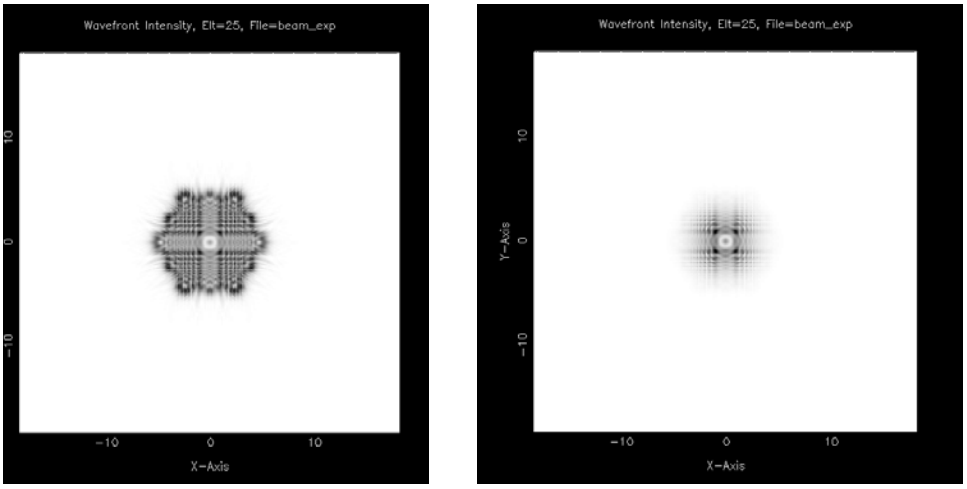
6.3.7 BEAM

The BEAm command allows the user to select a source beam profile. The default is a uniform beam. Alternatives are Gaussian, cosine-to-a-power, and dipole beam profiles.

If a Gaussian beam is choosen, the user is then prompted for the diameter of the beam in x and y, where the diameter is the standard deviation of the Gaussian profile.

Figure 46 shows the intensity for a beam expander using a uniform beam and the intensity for the same system using a Gaussian beam.

```
MACOS>in
Enter number of element where data is to be generated: [25]: 25
Tracing 4641 rays and propagating 65536 grid points...
 NF/PP Prop between Elt 24 and Elt 25 to WF 1:
 z1= 0.0000D+00 dx1= 1.4286D-01
 z2= 1.0000D+06 dx2= 1.4286D-01 min= 1.4286D-01 max= 1.4286D-01 dev= 6.8863D-
15 lin=100.00%
Compute time was 23.8438 sec
Wavefront Propagation Data Summary:
Wavelength= 8.0000000D-07; Transmission Distance= 0.0000000D+00
u-v plane diam= 3.6428574E+01 du= 1.4285715E-01
x-y plane diam= 3.6428574E+01 dx= 1.4285715E-01
Type <RETURN> for next page:
MACOS>beam
Enter beam type (UNIFORM, GAUSSIAN, COS**POWER, DIPOLE): [UNIFORM]: gau
Enter x beam waist radius: [1.00000]: 0.3
Enter y beam waist radius: [1.00000]: 0.3
MACOS>in
Enter number of element where data is to be generated: [25]: 25
Tracing 4641 rays and propagating 65536 grid points...
 NF/PP Prop between Elt 24 and Elt 25 to WF 1:
 z1= 0.0000D+00 dx1= 1.4286D-01
 z2= 1.0000D+06 dx2= 1.4286D-01 min= 1.4286D-01 max= 1.4286D-01 dev= 6.8863D-
15 lin=100.00%
Compute time was 20.4062 sec
Wavefront Propagation Data Summary:
Wavelength= 8.0000000D-07; Transmission Distance= 0.0000000D+00
u-v plane diam= 3.6428574E+01 du= 1.4285715E-01
x-y plane diam= 3.6428574E+01 dx= 1.4285715E-01
Type <RETURN> for next page:
MACOS>
```



Uniform beam Gaussian beam

FIGURE 46 Propagation examples with Gaussian and uniform beams

SECTION 7 Beam Propagation and Image Simulation

This section presents the MACOS image generation and plotting commands. The complex amplitude matrices can be plotted to show pupil functions, point images, or images of extended objects. Composed images can combine multiple objects or multiple colors. Pixel array detectors can be simulated. Noise effects and image blurring can be simulated. Plot types include wireframe, contour, grayscale, row, and columns. Text and binary outputs are also available. The commands discussed in this section are:

```
INtensity        GAIn
PIxilate         AMplitude
WINdow           COMpose
ADD              DADD
MULTISPEC        RFILT
NOISE            SEED
BLUR             LOG
REAl             IMAGinary
GRAy             WIRe
SLIce            CONtour
COLumn           ROW
TEXt             BINary
FITs             MATlab
GBLUR
```

7.1 Background

MACOS propagation commands, such as `INtensity`, propagate the beam through the optics to produce a complex amplitude matrix or matrices representing the light beam as projected onto a particular surface. The commands described in this section generate the complex amplitude matrices, perform basic analyses on them, and plot the results. Plots of intensity, magnitude and phase, and the real and imaginary parts of the amplitude are available. Images, or intensity plots, can be digitized or *pixilated*, simulating the images received by a pixel array detector such as a CCD. Multiple-source images can be composed on a single pixel array, allowing the accumulation of multiple-color or multiple object images. Simulated noise can be added.

Propagation works as follows. If no propagation command has been executed on the current system, it starts at the source. The rays are initialized according to the aperture and source geometry. The complex amplitude matrices are initialized to zero for those grid points that are not associated with rays (e.g. padding in the wavefront array). The grid points which are associated with rays are initialized to achieve uniform amplitude with total magnitude equal to the value set by the `Flux` .in-file parameter. If a propagation has been performed previously, to an element upstream from the current element, then the previous values of the complex amplitude matrix are the starting point for the current computations.

The light is then advanced surface-by-surface through the system. Rays are traced to determine the phases that drive the geometric and Fresnel propagators. The propagators

are invoked at the end of each specified diffraction to update the complex amplitude matrices. Depending on the compile-time parameter `mWF`, one or more of these are stored in the `WFElt` array. These are accessible to the image generation commands for plotting. The propagation terminates when it has computed the complex amplitude at the specified last element.

Propagation commands print a summary of the scaling calculations for Fresnel propagation. These provide information on the regularity of the ray grid (the “lin” or linearity is displayed as a percentage) at each reference surface. Warnings are issued if rays become undefined (usually indicative of a problem with the .in-file prescription).

7.2 Beam Propagation Commands

To introduce and motivate the various beam propagation and image simulation commands, we will use a couple of examples. The first example is a simple non-optimized coronagraph. As shown on Figure 47, this system consists of three telescopes in sequence. At the focus of the first telescope is an occulting mask (`iElt`=6), which is designed to suppress the light from the core of an on-axis star. The beam is then re-expanded. A second mask is placed at the system stop (`iElt`=11), which is located in the collimated beam between the second and third telescopes. This forms a Lyot stop, which suppresses high-frequency light scattered from the occulted axial star. The detected on-axis image (`iElt`=16) is a somewhat complex and chaotic PSF, with greatly reduced core intensity.

Off-axis images are passed with very little attenuation, yielding near-ideal PSFs. The net effect is to greatly increase the visibility of dim off-axis objects compared to bright on-axis objects. Coronographs can greatly decrease dynamic range between 2 very close objects, such as a star and a planet might provide, enabling easier observation of the dimmer object.

The prescription and macro for generating the results plotted in this section are `coroExample.in` and `coroExample.jou`, respectively. These files are listed in Appendix A.6.

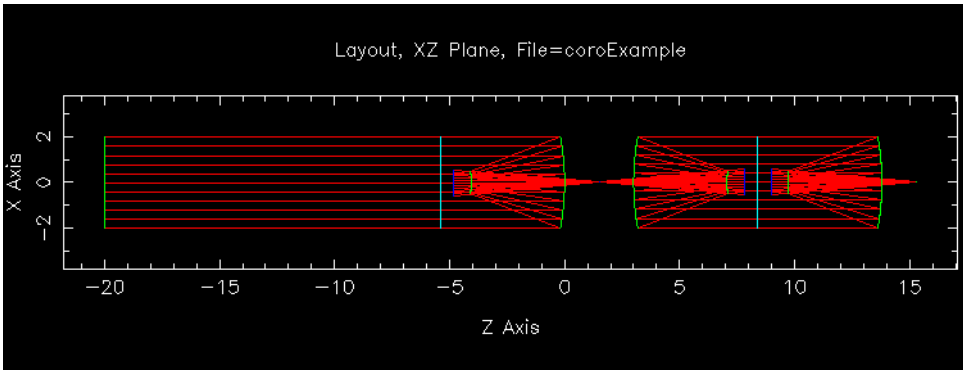


FIGURE 47 `DRAW` sketch of `coroExample.in` system.

7.2.1 INTENSITY

The `INtensity` command takes the complex amplitude matrix at the specified element, computes the modulus squared to determine the intensity, and plots or exports it according to the active plot option (see Section 7.3).

The INtensity command only accepts one argument–the number of the surface at which the intensity is viewed. This surface must be a stored wavefront. In the default gray scale plot, the intensity is displayed with black and white corresponding to the maixmum and minimum intensities, respectively.

We begin the example by propagating the on-axis beam through the first part of a near-field diffraction propagation (PropType=NF1), observing the image at the focus of the first telescope, both with and without the occulting mask:

```
MACOS> mod
MOD: [q]:  nObs(6)=0                                    Turns off the occulting mask
nObs(  6) =    0
MOD: [q]:  q
MACOS> int 6
Enter number of element where data is to be generated: [1]: 6
Tracing  12661 rays and propagating  262144 grid points...
 Prop to reference point Elt  5 and Elt  6 to WF  1:
 z1= 6.3293D+00 dx1= 8.8773D-03 min= 8.8703D-03 max= 8.8832D-03 dev=
3.7728D-06 lin= 99.96%
 z2= 1.0000D+30 dx2= 1.3925D-06
Compute time was    8.7598 sec
Wavefront Propagation Data Summary:
Wavelength= 1.0000000D-06;  Source Flux= 1.0000000D+00
u-v plane diam= 4.5362873D+00  du= 8.8772746D-03
x-y plane diam= 7.1158918D-04  dx= 1.3925425D-06
Peak intensity= 3.7152264D-02; Peak occurs at i=  257, j=  257
Sum of intensity= 8.7709310D-01
MACOS>
```

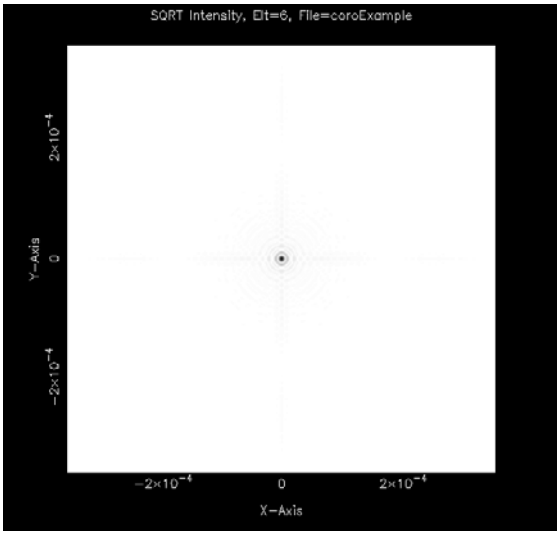


FIGURE 48 Unocculted image at first focal plane (coronagraph example)

Continuing with the example, the beam intensity can be observed at the other points of interest in the coronagraph, simply by specifying the appropriate element numbers. At the pupil, the Lyot stop, and then at the detector, the intensity plots are generated by:

```
MACOS> int 10
Enter number of element where data is to be generated: [6]: 10
Tracing  12661 rays and propagating  262144 grid points...
 Prop from reference point Elt  6 and Elt  7 to WF  1:
```

```
  z1= 1.0000D+30 dx1= 1.3925D-06
  z2=-6.3293D+00 dx2= 8.8773D-03 min= 8.8703D-03 max= 8.8832D-03 dev=
3.7728D-06 lin= 99.96%
  Geometric Prop between Elt  6 and Elt 10 to WF  1
 Compute time was    6.8867 sec
 Wavefront Propagation Data Summary:
 Wavelength= 1.0000000D-06;  Source Flux= 1.0000000D+00
 u-v plane diam= 0.0000000D+00  du= 0.0000000D+00
 x-y plane diam= 1.6256657D+01  dx= 3.1813417D-02
 Peak intensity= 7.6769075D-05; Peak occurs at i=  258, j=  297
 Sum of intensity= 1.3185163D-01
 MACOS> int 11
 Enter number of element where data is to be generated: [10]: 11
 Tracing  12661 rays and propagating  262144 grid points...
  Geometric Prop between Elt 10 and Elt 11 to WF  1
 Compute time was    0.4170 sec
 Wavefront Propagation Data Summary:
 Wavelength= 1.0000000D-06;  Source Flux= 1.0000000D+00
 u-v plane diam= 1.6256657D+01  du= 3.1813417D-02
 x-y plane diam= 1.6094488D+01  dx= 3.1496063D-02
 Peak intensity= 2.9045304D-05; Peak occurs at i=  267, j=  320
 Sum of intensity= 4.4725472D-02
 MACOS>
```

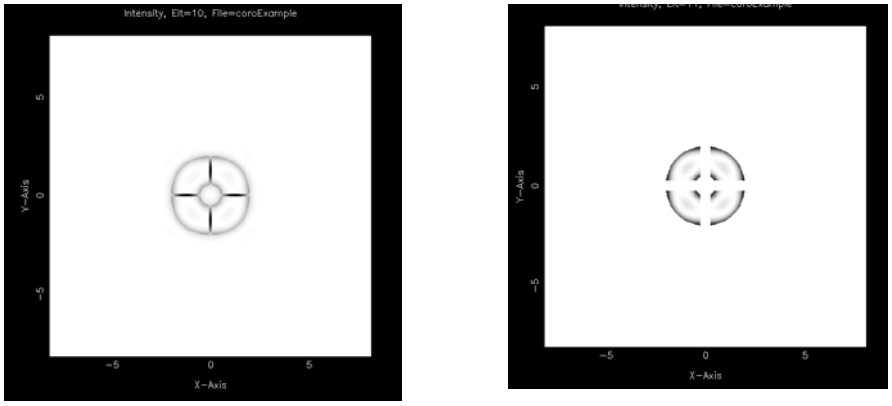


FIGURE 49 On-axis beam profile at a pupil; at the Lyot stop; and at the detector (coronagraph example)

7.2.2 PIXILATE

The `PIxilate` command is used to simulate discrete element detectors. MACOS calculates wavefront propagation on an array size which is independent of the detector size. The physical dimension of the diffracted image pixels is in general different from the detector pixel size being used to sample the signal. The `PIxilate` command imposes a square detector area over the intensity pattern. It averages the `INtensity` pattern within each detector pixel and sets that average equal to the value within the pixel. This is shown schematically in Figure 50. The `PIxilated` grid is centered on the chief ray (unless the `PLOcate` command has been executed) and the image location is determined by the Fourier transform of the exit pupil OPD map. Residual tilt in the OPD will shift the image from the chief ray intercept on the focal plane.

The `PIxilate` command has three arguments: the number of pixels per side (of the detector), the pixel width, and the element where the signal is to be `PIxilated`.

Continuing our example, we use the pixilate command to get a closer look at the occulted and unocculted images at the first focus:

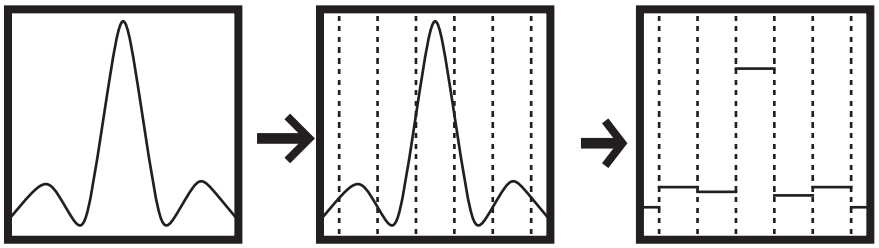


FIGURE 50 Pixilization of images

```
MACOS> pix 6
Enter number of element where data is to be generated: [6]: 6
Wavefront Propagation Data Summary:
Wavelength= 1.0000000D-06;  Source Flux= 1.0000000D+00
u-v plane diam= 2.2394683D+00  du= 1.7633609D-02
x-y plane diam= 3.5613167D-04  dx= 2.8041864D-06
Enter number of pixels per side: [128]: 64
Enter size of pixel: [2.804186E-06]:
Peak intensity= 1.6217206D-01; Peak occurs at i=   65, j=   65
Sum of intensity= 9.0994070D-01
MACOS> mod
MOD: [q]: nObs(6)=1                                   Reinstates the occulting mask
nObs(  6) =    1
MOD: [q]:  q
MACOS> pix 6 64 2.804186E-06
Enter number of element where data is to be generated: [16]: 6
Tracing   3210 rays and propagating   16384 grid points...
 Prop to reference point Elt  5 and Elt  6 to WF  1:
 z1= 6.3293D+00 dx1= 1.7634D-02 min= 1.7628D-02 max= 1.7637D-02 dev=
2.6054D-06 lin= 99.99%
 z2= 1.0000D+30 dx2= 2.8042D-06
Compute time was    0.4272 sec
Wavefront Propagation Data Summary:
Wavelength= 1.0000000D-06;  Source Flux= 1.0000000D+00
u-v plane diam= 2.2394683D+00  du= 1.7633609D-02
x-y plane diam= 3.5613167D-04  dx= 2.8041864D-06
Enter number of pixels per side: [128]: 64
Enter size of pixel: [2.804186E-06]:
Peak intensity= 8.2110545D-04; Peak occurs at i=   65, j=   72
Sum of intensity= 1.1480418D-01                    Note the reduced intensity
MACOS>
```

7.2.3 STRETCH

It is often important to be able to visualize very low-level details of an image or beam. This may require changing the stretch of the image: the scaling of the intensity. Plotting on a log10 or sqrt intensity scale greatly increases the dynamic range of an image. The MACOS STRetch command changes the intensity scaling, to log10, sqrt, or linear scale.

An example of setting the stretch is:

```
MACOS> stretch
```

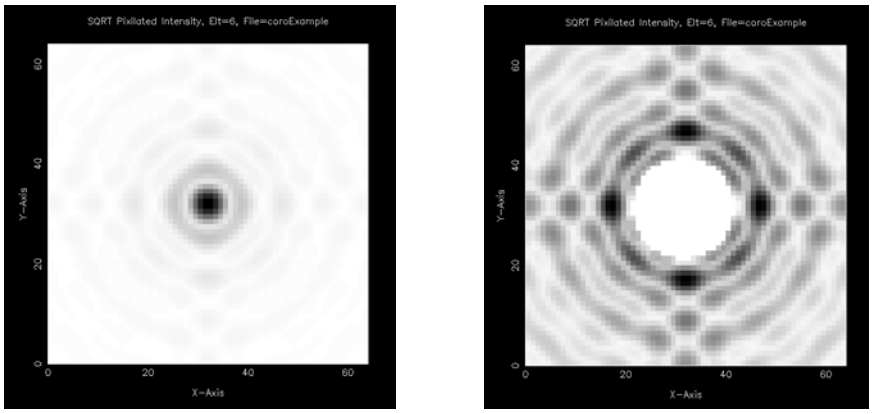


FIGURE 51 Pixilated images at first focal plane (coronagraph example)

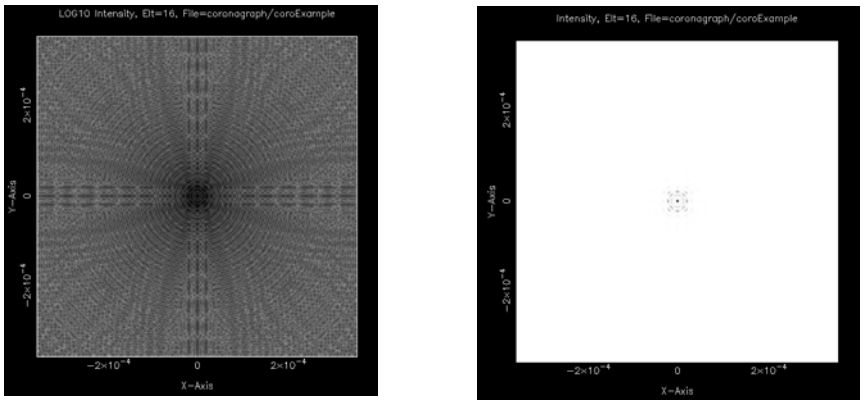


FIGURE 52 Pixilated images at log10 and sqrt stretch (coronagraph example)

```
Enter image stretch type (LINEAR, LOG10, SQRT): [LINEAR]:  log10
```

7.2.4 WINDOW

Using `PIXilate` to simulate detector images has a drawback: if the `FEX` command is used correctly, the image will always be centered in the PixArray window, even at differing field angles. The `WINdow` command provides a means of fixing the detector in space, so that changes in field angle result in appropriate motion of the image on the detector. `WINdow` sets `PixArray` equal to a specific region on the detector.

`WINdow` has four arguments. The first is a character string indicating the user's choice of output coordinate options. The second is the pixel size. The third is the location of the element vertex in pixel coords (this is used to define the origin of a pixel coordinate system, whose axis directions are defined by the previously selected coordinates. The fourth is the location in pixel coordinates of the center of the window:

```
MACOS> win
Enter output coordinate option (Tout, Enter or Beam): [Tout]:  Tout
Enter pixel size for placing window: [0.]: 1.392542d-6
Enter pixel coords of element vertex (x,y): [0.,0.]: 0,0
Enter window location in pixel coords (x,y): [0.,0.]: 0,0
```

7.2.5 COMPOSE

The `COMpose` command defines the pixel size and dimensions for a detector window. It also sets up an accumulate option, whereby images from a series of propagations, under

arbitrarily different conditions, can be incoherently added. It is useful for creating composite images from multiple-source or multiple-wavelength objects.

COMpose images are created by successive applications of the ADD command. ADD propagates the current beam, pixilates the intensity to the COMpose grid, and adds it to what is already present. COMpose is also used with the NOIse and BLUR commands, to add simulated noise to an image. An example using the COMpose and ADD commands is presented in Section 7.2.6.

COMpose has three arguments: the number of pixels per side (of the detector), the pixel width, and the element where the signal is to be COMposed.

Continuing the coronagraph example, a pixel array is defined at the detector element:

```
MACOS> compose
Enter element where image is to be composed: [16]: 16
Enter number of pixels per side: [512]: 64
Enter size of pixel: [1.000000E-03]: 1.392542E-06
```

7.2.6 ADD

The ADD command propagates the current beam, pixilates the intensity to the COMpose grid, and adds it to the composed pixel array image, which is stored in PixArray.

Continuing the coronagraph example, the on-axis image is added to the detector pixel array:

```
MACOS> add
 Wavefront Propagation Data Summary:
 Wavelength= 1.0000000D-06;  Source Flux= 1.0000000D+00
 u-v plane diam= 4.5362873D+00  du= 8.8772746D-03
 x-y plane diam= 7.1158918D-04  dx= 1.3925425D-06
 Peak intensity= 3.8883158D-04; Peak occurs at i=  257, j=  257
 Sum of intensity= 4.4725467D-02
 Image added. Plot composed image?  [YES]:  yes
```

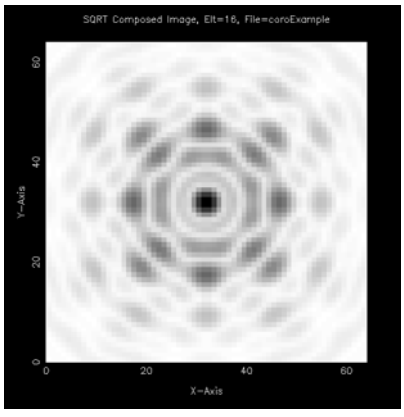


FIGURE 53 Axial image ADDed to begin composed image.

7.2.7 Propagating a Perturbed Beam

The next step in the coronograph example is to define an off-axis star and propagate it to the detector, adding it into the composed image. The result is an image with 2 PSFs on

it, allowing us to evaluate the effectiveness of the coronagraph in reducing dynamic range between neighboring objects.

```
MACOS> stop
 Stop at ELT or OBJect point?  [Elt]:  elt
 Enter system stop element number: [1]: 11
 Enter offset vector (dx,dy): [0.,0.]: 0,0
 Computed StopPos=  0.000000000D+00  0.000000000D+00 -1.427723273D+02
 MACOS> ffp^                                  Find field angles that put the off-axis
 Enter element number: [16]: 6                  spot right next to the occulting disk
 Enter offset vector in global units (dx,dy): [0.,0.]: 2d-5,2d-5
 Field angle finder results:
 Did    1 iterations, error=  1.424113342D-15
  Old dx=  0.000000000D+00  New dx=  2.000000000D-05
           0.000000000D+00           2.000000000D-05
 Old crd=  0.000000000D+00 New crd=  8.916721422D-07
           0.000000000D+00          -8.916721422D-07
           1.000000000D+00           1.000000000D+00
 Old crp=  0.000000000D+00 New crp=  1.094726641D-04
           0.000000000D+00          -1.094726641D-04
          -2.000000000D+01          -2.000000000D+01
 Accept the new chief ray?  [YES]:  yes
 MACOS> ors                  Find new reference surfaces for diffraction propagation
 Enter number of element to be optimized: [1]: 5
 Reference surface optimization results:
 TTL OPD=  1.656171318D-09  nCalls=  18
   Old f=  6.329334641D+00   New f=  6.329334641D+00
   Old z=  6.329334641D+00   New z=  6.329334641D+00
 Old psi=  0.000000000D+00 New psi= -2.730547038D-06
           0.000000000D+00           2.730547038D-06
           1.000000000D+00           1.000000000D+00
 Old Vpt=  0.000000000D+00 New Vpt=  3.728254596D-05
           0.000000000D+00          -3.728254596D-05
          -4.829334641D+00          -4.829334641D+00
 Accept the new element data?  [YES]:  yes
 MACOS> ors
Enter number of element to be optimized: [1]: 7
 Reference surface optimization results:
 TTL OPD=  1.656171779D-09  nCalls=  18
   Old f=  6.329334641D+00   New f=  6.329334641D+00
   Old z= -6.329334641D+00   New z= -6.329334641D+00
 Old psi=  0.000000000D+00 New psi=  2.730547038D-06
           0.000000000D+00          -2.730547038D-06
          -1.000000000D+00          -1.000000000D+00
 Old Vpt=  0.000000000D+00 New Vpt=  2.717454041D-06
           0.000000000D+00          -2.717454041D-06
           7.829334641D+00           7.829334641D+00
 Accept the new element data?  [YES]:  yes
 MACOS> fex                                          Find new exit pupil as well
 Enter number of exit pupil return surface: [15]: 15
  Tracing   333 rays...
              Chief ray location: x=-2.0000000D-05 y= 2.0000000D-05 z=
1.5300000D+01
               Centroid location: x=-2.0042495D-05 y= 2.0042495D-05 z=
1.5300000D+01
  Centroid offset from chief ray: x= 4.2495119D-08 y=-4.2495119D-08 z=-
6.7501560D-14
 Exit pupil finder results:
   Old f=  6.329334641D+00   New f=  7.324539719D+00
   Old z=  6.329334641D+00   New z=  7.324539719D+00
 Old psi=  0.000000000D+00 New psi= -2.736348769D-06
           0.000000000D+00           2.736348769D-06
           1.000000000D+00           1.000000000D+00
 Old Vpt=  0.000000000D+00 New Vpt=  1.263889675D-13
           0.000000000D+00          -1.263889675D-13
```

```
          8.970665359D+00           7.975460281D+00
 Accept the new element?  [YES]:  yes
 MACOS> mod                  This object is 100 times dimmer than the on-axis object
 MOD: [q]:  flux=0.01
 Flux =   1.000000000D-02
 MOD: [q]:  q
 MACOS> stretch
 Enter image stretch type (LINEAR, LOG10, SQRT): [LINEAR]:  log10
 MACOS> int                                          This image is shown in Figure 54
 Enter number of element where data is to be generated: [16]: 6
 Tracing  12661 rays and propagating  262144 grid points...
  Prop to reference point Elt  5 and Elt  6 to WF  1:
  z1= 6.3293D+00 dx1= 8.8773D-03 min= 8.8703D-03 max= 8.8832D-03 dev=
3.7730D-06 lin= 99.96%
  z2= 1.0000D+30 dx2= 1.3925D-06
 Compute time was    8.9688 sec
 Wavefront Propagation Data Summary:
 Wavelength= 1.0000000D-06;  Source Flux= 1.0000000D-02
 u-v plane diam= 4.5362873D+00  du= 8.8772746D-03
 x-y plane diam= 7.1158918D-04  dx= 1.3925425D-06
 Peak intensity= 3.7145605D-04; Peak occurs at i=  257, j=  257
 Sum of intensity= 8.6867251D-03
```

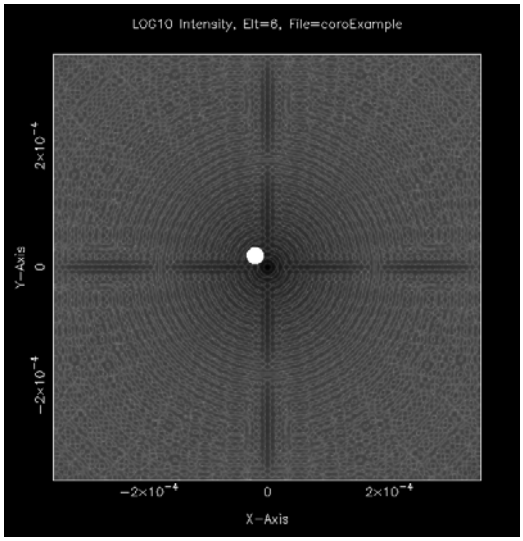


FIGURE 54 Off-axis image at the first focus, showing closeness to occulting mask.

```
 MACOS> add        Now the off-axis star is added to the composed image (Figure 55)
 Tracing  12661 rays and propagating  262144 grid points...
  Prop from reference point Elt  6 and Elt  7 to WF  1:
  z1= 1.0000D+30 dx1= 1.3925D-06
  z2=-6.3293D+00 dx2= 8.8773D-03 min= 8.8703D-03 max= 8.8832D-03 dev=
3.7728D-06 lin= 99.96%
  Scalar FF Prop between Elt 15 and Elt 16 to WF  1:
  z1= 7.3245D+00 dx1= 1.0273D-02 min= 1.0265D-02 max= 1.0280D-02 dev=
4.3667D-06 lin= 99.96%
  z2= 0.0000D+00 dx2= 1.3925D-06
 Compute time was   14.0371 sec
 Wavefront Propagation Data Summary:
 Wavelength= 1.0000000D-06;  Source Flux= 1.0000000D-02
```

```
 u-v plane diam= 5.2495594D+00  du= 1.0273110D-02
 x-y plane diam= 7.1158918D-04  dx= 1.3925425D-06
 Peak intensity= 2.2644562D-04; Peak occurs at i=  257, j=  257
 Sum of intensity= 6.8026488D-03
 Image added. Plot composed image?  [YES]:  yes
```

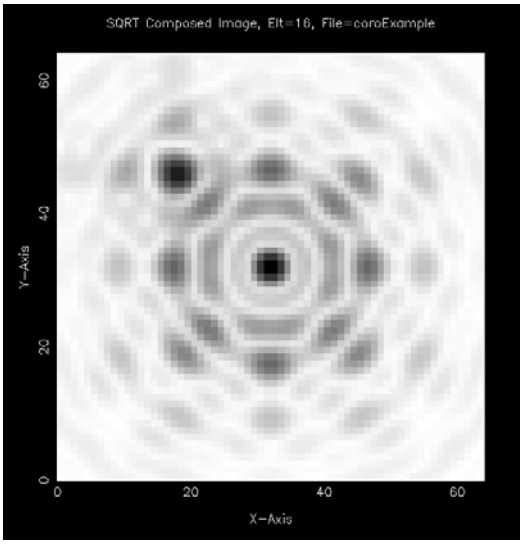


FIGURE 55 Off-axis image added to composed detector image. Note that the image dynamic range is about 1, while the objects differ by 100 in flux.

7.2.8 GBLUR

The GBLUR command is used to simulate "long-exposure" jitter effects or other phenomena that cause smooth spreading of the image. It propagates the beam to the specified surface, and then convolves the resulting intensity with a gaussian. It adds the result to the pixel array buffer (PixArray) and optionally displays it. The blurring is done at the WF sampling density, rather than the pixel density, which will generally be more accurate if the blurring occurs prior to the digitization of the image at the detector. Before using GBLUR, the user must invoke the COMPose command to set the detector pixel geometry. The width of the blur kernel is defined in the function arguments.

The following example sets up a typical case. The detector is first defined, at element 21, by specifying the number and size of the pixels. Note that the pixel size is specified to be the same as the WF pixel size (dx) for this example, which shows the blurred image at its highest resolution.

The GBLUR function is then invoked, first with a kernel width much smaller than the pixel size, and then with a kernel size much larger. The results

```
MACOS> compose
 Enter element where image is to be composed: [21]: 21
 Enter number of pixels per side: [128]: 128
 Enter size of pixel: [1.000000E-03]: 1.0007177D-05
 MACOS> gblur
 Wavefront Propagation Data Summary:
 Wavelength= 1.0000000D-06;  Source Flux= 1.0000000D+00
 u-v plane diam= 1.6365868D-01  du= 1.2886510D-03
 x-y plane diam= 1.2709115D-03  dx= 1.0007177D-05
 Peak intensity= 9.5305603D-02; Peak occurs at i=   65, j=   65
 Sum of intensity= 8.6054104D-01
 Enter Gaussian blur kernel width in base units (0,0): [0.,0.]: 1d-6,1d-6
Image added. Plot composed image?  [YES]:  yes
```

```
Type <RETURN> for next page:
MACOS> gblur
Wavefront Propagation Data Summary:
Wavelength= 1.0000000D-06;  Source Flux= 1.0000000D+00
u-v plane diam= 1.6365868D-01  du= 1.2886510D-03
x-y plane diam= 1.2709115D-03  dx= 1.0007177D-05
Peak intensity= 9.5305603D-02; Peak occurs at i=   65, j=   65
Sum of intensity= 8.6054104D-01
Enter Gaussian blur kernel width in base units (0,0): [0.,0.]: 300d-6,300d-6
Image added. Plot composed image?  [YES]:  yes
Type <RETURN> for next page:
MACOS>
```

FIGURE 56 GBLUR example.

7.2.9 DAD

The DAD command displays the current COMposed image. This is useful if you have forgotten what the composed image looks like, or if you want to view it using a different stretch or plot type. It has no arguments.

7.2.10 GAIN

The GAIn command computes the ratio of the light diffracted by the optical system to a uniformly radiating point source of the same intensity. It is used mainly to evaluate the far-field performance of radio-frequency antennas. This command has one argument, the surface on which to compute the gain. An example is provided by the Luneberg Lens Antenna (Appendix A.3.3), and shown in Figure 58.

7.2.11 AMPLITUDE and PHASE

Each element in the MACOS wavefront propagation matrix is a complex number which can be expressed in terms of its real and imaginary components (i.e., a + b*i where a and b are real numbers). This complex number is a vector in the complex plane with two components, real and imaginary as shown in Figure 59.

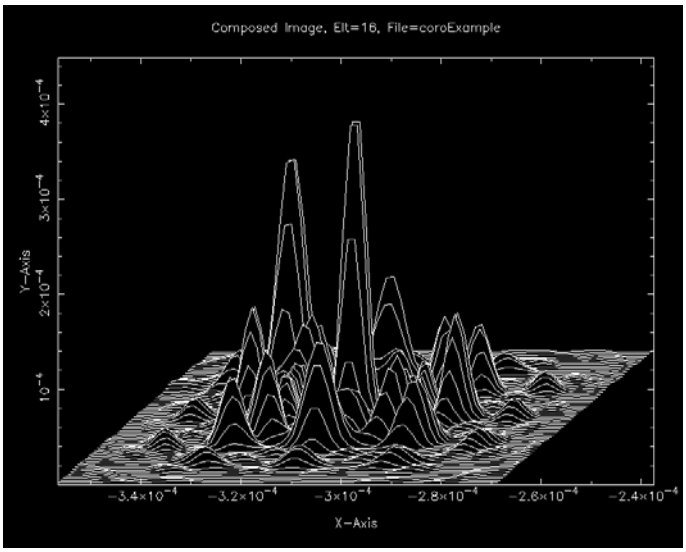


FIGURE 57 Composed detector image displayed at linear stretch, slice plot type.

FIGURE 58 Far-field gain of Luneberg Lens Antenna.

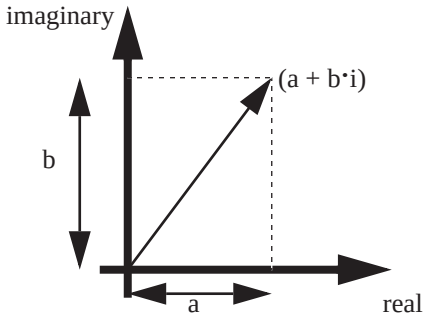


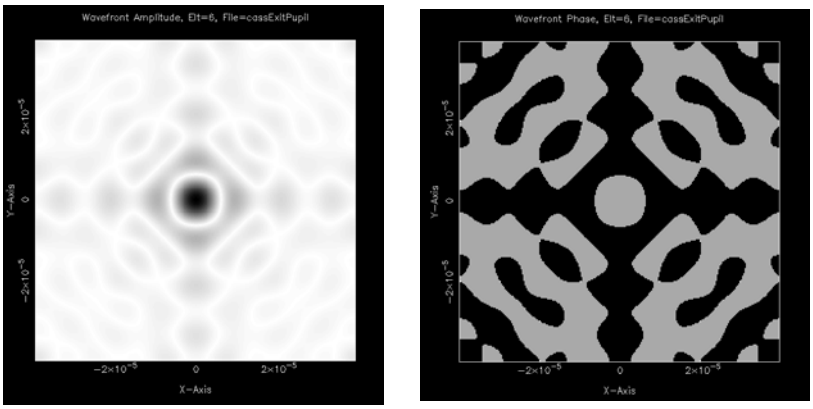
FIGURE 59 Graphing numbers in the complex plane

The AMplitude and PHASE commands directly plot the corresponding components of the complex amplitude matrix. Figure 60 provides an example. The first plot shows the

amplitude or length of the vector. The second plot shows the phase of the vector, or the angle which the vector makes with the real axis modulo 360 degrees. The AMplitude command requires the wavefront element number nElt as input.

This example uses CassWithExitPupil.in in Appendix A.1.4.

```
MACOS>amp
Enter element where WF is to be plotted:5
Wavefront plane  1 is evaluated at this element.
Wavefront Propagation Data Summary:
Wavelength= 1.0000000D-06;  Transmission Distance= 5.5601459D+00
u-v plane diam= 8.4230375E+00  du= 3.3031520E-02
x-y plane diam= 1.6767097E-04  dx= 6.5753318E-07
```



Amplitude Phase

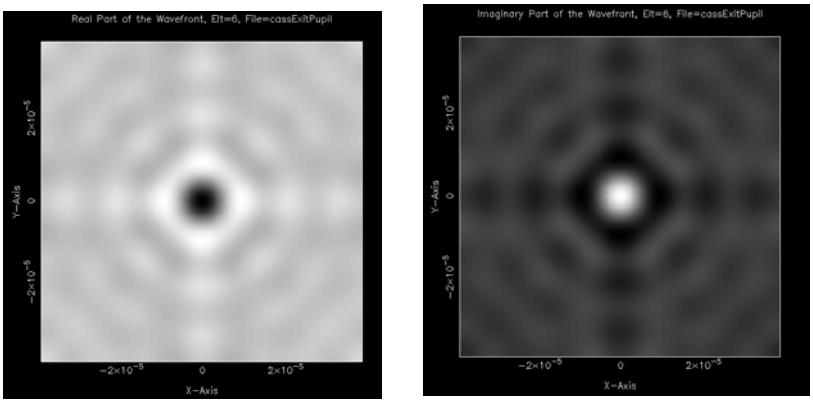
FIGURE 60 AMplitude example

7.2.12 REAL and IMAGINARY

As discussed in Section 7.2.11, the wavefront propagation matrix elements are complex numbers. The REAl and IMAGINARY commands display these two components of the complex amplitude matrix. Figure 61 shows the real and imaginary components. The first graph displays the real component of each complex pixel number in the array. The second graph shows the imaginary component. The single command argument is the element number of interest.

This example uses CassWithExitPupil.in in Appendix A.1.4.

```
MACOS>real
Enter element where WF is to be plotted:5
Wavefront plane  1 is evaluated at this element.
Wavefront Propagation Data Summary:
Wavelength= 1.0000000D-06;  Transmission Distance= 5.5601459D+00
u-v plane diam= 8.4230375E+00  du= 3.3031520E-02
x-y plane diam= 1.6767097E-04  dx= 6.5753318E-07
Type <RETURN> for next page:
Type <RETURN> for next page:
MACOS>
```



Real part Imaginary part

FIGURE 61 REAl example

7.3 Plot Type Commands

7.3.1 GRAY

The GRAy command resets the type of plot to GRAy scale. This is the default plot type. It has no arguments.

7.3.2 WIRE

The WIRe command takes all the values in an array and first plots them as three dimensional points. Then each point is then connected to its nearest neighbor on the grid by a line. Perspective is added and the obscured lines are removed. The result is a surface-like representation of the data as shown in Figure 62. There is no scale on the graph. The WIRe command is always used in conjunction with other commands and has no arguments.

```
MACOS>wire
MACOS>in
Enter element where WF is to be plotted:5
Wavefront plane  1 is evaluated at this element.
Wavefront Propagation Data Summary:
Wavelength= 1.0000000D-06;  Transmission Distance= 5.5601459D+00
u-v plane diam= 8.4230375E+00  du= 3.3031520E-02
x-y plane diam= 1.6767097E-04  dx= 6.5753318E-07
Type <RETURN> for next page:
MACOS>
```

7.3.3 SLICE

The SLIce command is very similar to the WIRe command in that it displays the array of data as a surface. In this instance, the graph also contains a scale so the data values can be read directly. Like the WIRe command, SLIce is used with other commands and has no arguments.

Figure 57 provides an illustration of the SLICE plot type.

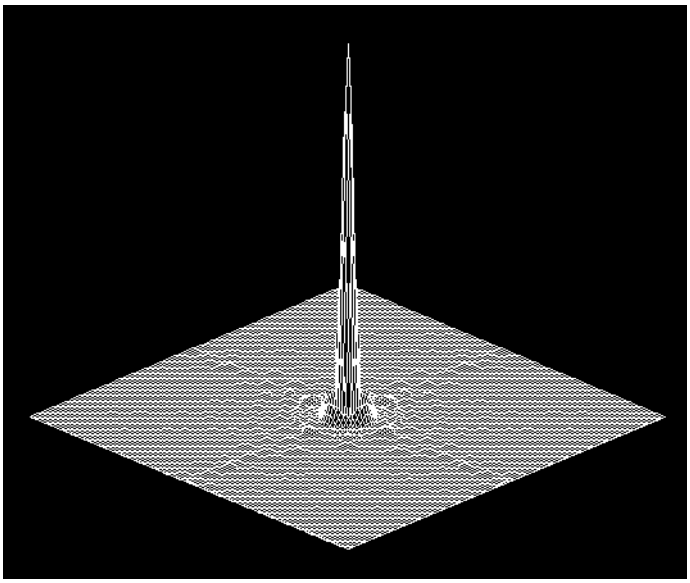


FIGURE 62 WIRe plot

7.3.4 CONTOUR

The CONtour command sets the plot type to display data using a contour map (an elevation map). Data points with similar values are connected by the same line. The CONtour map shows graphically those points with similar values. The CONtour command has no arguments.

This example uses CassWithExitPupil.in in Appendix A.1.4. Figure 63 shows the results.

```
MACOS>con
MACOS>opd
Enter element where OPD is to be evaluated:5
Average OPD is -1.894794E-12
Average total path is  1.5622291803998E+01
Average delta path is -3.7770819600190E-01
RMS OPD error is  1.091947E-12
Type <RETURN> for next page:
MACOS>
```

7.3.5 COLUMN and ROW

The COLumn and ROW commands set the plot type to display a slice across the data surface. The slice is along a column or row of the data matrix, respectively. The user is prompted for the column or row number when the data is to be plotted out (for instance, when an INTensity command is issued). The COLumn and ROW commands have no arguments.

This example uses CassWithExitPupil.in in Appendix A.1.4. Figure 64 shows the results.

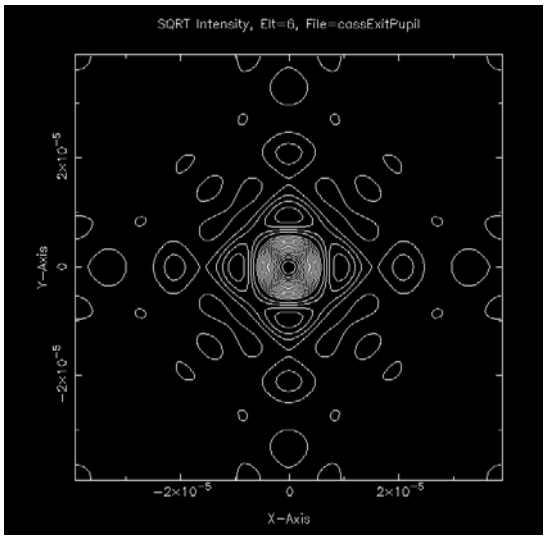


FIGURE 63 `CONtour` example

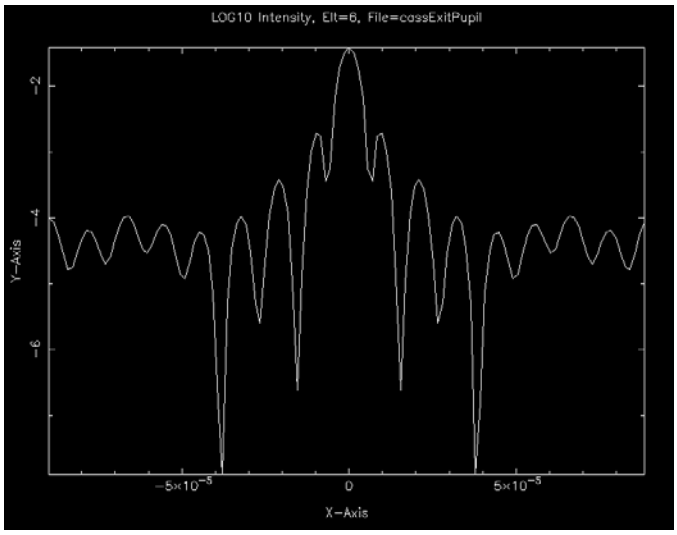


FIGURE 64 `COLumn` example

7.3.6 TEXT

The `TEXt` command writes the data points in the array to a file for further analysis. The file is in ASCII format and can be printed or read into a text editor. There are up to `mdttl` rows of numbers, each with up to `mdttl` numbers across: this can be quite a large file! The data points are the same as numbers used to create the graphical display. They are single precision. The `TEXt` command creates files as output and has no arguments. The filenames specify the contents and are of the format: [filename].[image type][surface number].txt where [image type] can be in (intensity), am (amplitude), ph (phase), re (real), im (imaginary), spot, and OPD.

This example uses `CassWithExitPupil.in` in Appendix A.1.4. The `TEXt` file is stored in **`CassWithExitPupil.int5.txt`**

```
MACOS>text
MACOS>intensity
Enter element where WF is to be plotted:5
Wavefront plane  1 is evaluated at this element.
Wavefront Propagation Data Summary:
Wavelength= 1.0000000D-06;  Transmission Distance= 5.5601459D+00
u-v plane diam= 8.4230375E+00  du= 3.3031520E-02
x-y plane diam= 1.6767097E-04  dx= 6.5753318E-07
Writing Wavefront Intensity, Elt=5
FORMATTED File=CassWithExitPupil.int5.txt
MACOS>
```

7.3.7 BINARY

The `BINary` command writes numerical files which can be used by other signal/image processing software. It takes the array data and writes it out to a file in `BINary` format. The `BINary` command produces files for output and has no arguments. The filenames specify the contents and are of the format: [filename].[image type][surface number] where [image type] can be in (intensity), am (amplitude), ph (phase), re (real), im (imaginary), spot, and OPD.

7.3.8 MATLAB

The `MATlab` command writes the data arrays to `MATlab` format. The `MATlab` command produces files for output and has no arguments. The filenames specify the contents and are of the format: [filename].[image type][surface number].dat where [image type] can be in (intensity), am (amplitude), ph (phase), re (real), im (imaginary), spot, and OPD.

7.3.9 FITS

The `FITs` command writes the data arrays to fits format files. The `FITs` command produces files for output and has no arguments. The filenames specify the contents and are of the format: [filename].[image type][surface number].fit, where [image type] can be in (intensity), am (amplitude), ph (phase), re (real), im (imaginary), and OPD.

SECTION 8 *Differential Ray-Tracing and Linear Optical Models*

The MACOS differential ray-trace functions are used to compute optical sensitivity matrices. These provide linear models of the effects of optical element perturbations on optical performance. The linear models can be exercised internally to MACOS using functions such as `LPErturb`, `LOPd` and `LINtensity` which parallel commands introduced earlier. Linear optical models can also be `EXPported` for use external to MACOS.

```
BUild
DMBuild
PArtials
LPErturb
LPRead
LREset
LSPot
LOPd
LPIxillate
LINtensity
EXPort
```

8.1 Background

linear models of optical systems provide a linear matrix transformation between element perturbations and ray perturbations. These optical models can be used in place of the full-blown ray-trace calculations for systems that undergo only small changes. They can also be used in ways that the nonlinear models cannot, such as in linear statistical studies.

Linear models are generally valid where

- The small-angle approximation is valid for changes in ray direction
- The ray transverse aberration is “small” relative to the curvature of the optical elements

The limits are not absolute, but vary from system to system. The user can check the range of validity of a linear model for a particular system. This is done by using the MACOS `LPErtrurb`, `LOPd`, `LSPot` and `LINtensity` commands to exercise the linear model. Results are compared with exact results produced for the same cases using the MACOS `PERturb`, `OPD`, `SPOt` and `INtensity` commands. If, over the expected range of perturbation values, the linear models produce results that are within the accuracy requirements of the particular analysis, then they can be safely used for that analysis. Otherwise a nonlinear SMACOS model should be used.

8.1.1 Linear Ray States

Rays are defined at any particular point in a beam train by their location, direction and length. In MACOS, the current *exact ray state* is defined as the position, direction and path length of a ray immediately after the current element (see Figure 65). The *position* of the `iRay`th ray is a 3-vector giving the point of incidence with the current element. MACOS stores it in the array `RayPos` as `RayPos(1:3,iRay)`. The *direction* of the `iRay`th ray is a unit-magnitude 3-vector stored in `RayDir(1:3,iRay)`. Note that if the current element is a mirror or lens, the direction is of the reflected or refracted ray. The *optical path length* of the `iRay`th ray from the source to the current element is a scalar and is stored in `CumRayL(iRay)`.

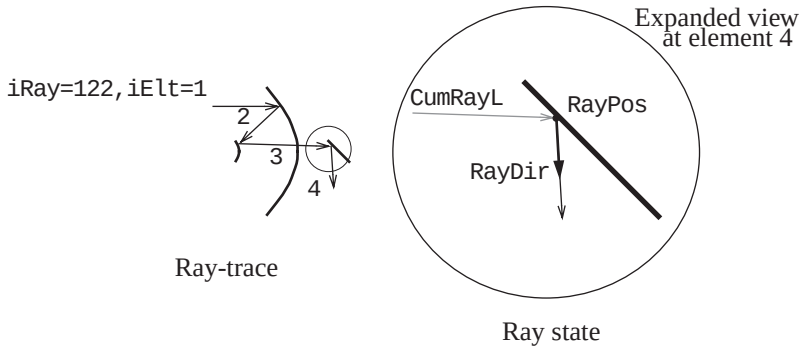


FIGURE 65 Ray state

The effect of a *perturbation* in the optical system, such as a rotation or translation of an optical element, is to *change the ray state* at every downstream element. Such perturbations can be applied using the `PERturb` command, and the ray retraced to find the new ray state.

If a perturbation is sufficiently small, it will effect the rays *linearly.* That is, the new ray state at every downstream element is very nearly a linear function of the perturbation. This is true for some range of perturbation magnitudes that varies with each system, but is in general many times the wavelength of light.

MACOS linear modeling features use *analytic differential* ray-trace equations to compute the partial derivatives of the rays with respect to element perturbations. The differential ray-trace equations are derived in References 1, 8, and 9. They provide a set of analytical formulas giving the ray partial derivative matrices in a form that is easily computed.

The *linear ray state* differs slightly from the exact ray state. The linear ray state is a single 7-vector, composed of a 3-vector of *ray direction perturbation*, a 3-vector of *ray transverse aberration*, and a scalar for *ray optical path difference*. It is written $\underset{\sim}{x}$

$$\underset{\sim}{x} = \begin{bmatrix} d\vec{r} \\ d\vec{\eta} \\ dL \end{bmatrix} \tag{8.1}$$

where $d\vec{r}$ is the ray direction perturbation. The ray direction perturbation is a 3-vector whose magnitude is small compared to 1 and which is perpendicular to the ray direction, `RayDir`. The total ray direction is equal to the vector sum of `RayDir` and $d\vec{r}$ (to the accuracy of the small angle approximation). Figure 66 shows the linear ray state.

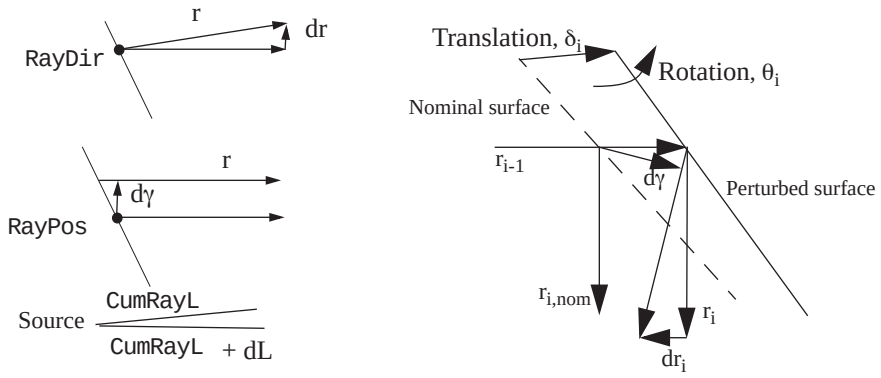


FIGURE 66 Linear ray state

The transverse aberration $\hat{\gamma}$ is the lateral displacement of the ray. It is a vector perpendicular to the ray direction, from the nominal ray position to a point on the perturbed ray. It is not necessarily located on the surface of the element and is not, in general, the derivative of the ray position, RayPos. Exceptions are the focal-plane element type and any surface that is perpendicular to the nominal ray direction.

The path length perturbation, dL, is the difference between the optical path length of the perturbed and nominal rays. It is not the OPD, as that is the difference of the ray and the chief ray. However, the OPD is easily computed from dL as discussed below.

8.1.2 Linear Models

The linear ray state at a particular element is a function of perturbations of the upstream elements and of perturbations in the input ray. These perturbations are with respect to a "nominal state" $(x_n)_{nom}$, which may or may not be zero. For each ray at an element n, the linear ray state has the functional form:

$$\hat{x}_n = (\hat{x}_n)_{nom} + \frac{\partial \hat{x}_n}{\partial \hat{u}_1}\hat{u}_1 + \ldots + \frac{\partial \hat{x}_n}{\partial \hat{u}_n}\hat{u}_n + \frac{\partial \hat{x}_n}{\partial \hat{x}_0}\hat{x}_0 \qquad \textbf{(8.1)}$$

Here the u vectors are 6-vectors of element perturbations, composed of 3-vectors $\hat{\theta}$ for element rotational perturbation and $\hat{\delta}$ for element translation perturbation:

$$\hat{u} = \begin{bmatrix} \hat{\theta} \\ \hat{\delta} \end{bmatrix} \qquad \textbf{(8.2)}$$

The individual optical sensitivity matrices can be printed using the PARtials command. The sensitivity of the nth ray to perturbations of the input (source) ray is a 7 by 7 matrix of the form:

$$\frac{\partial \hat{x}_n}{\partial \hat{x}_0} = \begin{bmatrix} \frac{\partial \hat{r}_n}{\partial \hat{r}_0} & \frac{\partial \hat{r}_n}{\partial \hat{\gamma}_0} & \frac{\partial \hat{r}_n}{\partial L_0} \\ \frac{\partial \hat{\gamma}_n}{\partial \hat{r}_0} & \frac{\partial \hat{\gamma}_n}{\partial \hat{\gamma}_0} & \frac{\partial \hat{\gamma}_n}{\partial L_0} \\ \frac{\partial L_n}{\partial \hat{r}_0} & \frac{\partial L_n}{\partial \hat{\gamma}_0} & \frac{\partial L_n}{\partial L_0} \end{bmatrix} \tag{8.3}$$

The sensitivity of the nth ray to perturbations of the ith element is a 7 by 6 matrix of the form:

$$\frac{\partial \hat{x}_n}{\partial \hat{u}_i} = \begin{bmatrix} \frac{\partial \hat{r}_n}{\partial \hat{\theta}_i} & \frac{\partial \hat{r}_n}{\partial \hat{\delta}_i} \\ \frac{\partial \hat{\gamma}_n}{\partial \hat{\theta}_i} & \frac{\partial \hat{\gamma}_n}{\partial \hat{\delta}_i} \\ \frac{\partial L_n}{\partial \hat{\theta}_i} & \frac{\partial L_n}{\partial \hat{\delta}_i} \end{bmatrix} \tag{8.4}$$

The complete expression for the linear ray state for a single ray in terms of element and input-ray perturbations is written in matrix form as:

$$\hat{x} = (\hat{x}_n)_{nom} + \begin{bmatrix} \frac{\partial \hat{x}_n}{\partial \hat{x}_0} & \frac{\partial \hat{x}_n}{\partial \hat{u}_1} & \cdots & \frac{\partial \hat{x}_n}{\partial \hat{u}_n} \end{bmatrix} \begin{bmatrix} \hat{x}_0 \\ \hat{u}_1 \\ \cdots \\ \hat{u}_n \end{bmatrix} \tag{8.5}$$

The same form is extended to give the linear ray state for multiple rays at the same element

$$\begin{bmatrix} \hat{x}_{ray1} \\ \cdots \\ \hat{x}_{raym} \end{bmatrix} = \begin{bmatrix} \hat{x}_{ray1} \\ \cdots \\ \hat{x}_{raym} \end{bmatrix}_{nom} + \begin{bmatrix} \begin{bmatrix} \frac{\partial \hat{x}_n}{\partial \hat{x}_0} & \frac{\partial \hat{x}_n}{\partial \hat{u}_1} & \cdots & \frac{\partial \hat{x}_n}{\partial \hat{u}_n} \end{bmatrix}_{ray1} \\ \cdots \\ \begin{bmatrix} \frac{\partial \hat{x}_n}{\partial \hat{x}_0} & \frac{\partial \hat{x}_n}{\partial \hat{u}_1} & \cdots & \frac{\partial \hat{x}_n}{\partial \hat{u}_n} \end{bmatrix}_{raym} \end{bmatrix} \begin{bmatrix} \hat{x}_0 \\ \hat{u}_1 \\ \cdots \\ \hat{u}_n \end{bmatrix} \tag{8.6}$$

Both of these equations have the form $x = x_{nom} + Cu$, where the *C-matrix* is the *optical sensitivity matrix*, giving the change in ray state due to perturbations of the optical elements. The C-matrix provides a linear model of the optical system. The MACOS `BUild` command assembles these sensitivities from the individual ray partial derivatives.

It can be extremely useful to incorporate local coordinates directly into the optical sensitivity C-matrices. If the perturbations for one element are to be applied in coordinates other than the global coordinate system–perhaps in an actuator coordinate system–then that local coordinate transformation matrix can be included directly within the C-matrix. The transform from element `iElt` coordinates to global coordinates is written ${}^{G}T^{iElt}$.

Similarly, local coordinates can be used at the output end of the linear model. This allows for the creation of models for ray direction only, or transverse aberration or path length only. Focal-plane axes are easily incorporated. We write the transform for ray state from global coordinates to output coordinates as ${}^{out}T^{G}$. As an example, the 1x7 output transform matrix that captures only the path length ray state in global coordinates is

$$ {}^{pathlength}T^{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{8.7} $$

As another example, the 2x7 output transform which takes the transverse aberration in ray state and casts it into detector coordinates x and y is

$$ {}^{detector}T^{G} = \begin{bmatrix} 0 & 0 & 0 & \hat{x} & 0 \\ 0 & 0 & 0 & \hat{y} & 0 \end{bmatrix} \tag{8.8} $$

Both actuator and detector coordinates are specified by the user as part of the .in-file prescription.

With the input and output coordinate matrices included, the C-matrix takes the form:

$$ \begin{bmatrix} \check{x}_{ray1} \\ \cdots \\ \check{x}_{raym} \end{bmatrix} = \begin{bmatrix} {}^{out}T^{G}\left[\frac{\partial \check{x}_n}{\partial \check{x}_0}{}^{G}T^{0} \;\; \frac{\partial \check{x}_n}{\partial \check{u}_1}{}^{G}T^{1} \;\; \cdots \;\; \frac{\partial \check{x}_n}{\partial \check{u}_n}{}^{G}T^{n}\right]_{ray1} \\ \cdots \\ {}^{out}T^{G}\left[\frac{\partial \check{x}_n}{\partial \check{x}_0}{}^{G}T^{0} \;\; \frac{\partial \check{x}_n}{\partial \check{u}_1}{}^{G}T^{1} \;\; \cdots \;\; \frac{\partial \check{x}_n}{\partial \check{u}_n}{}^{G}T^{n}\right]_{ray\mathrm{nRay}} \end{bmatrix} \begin{bmatrix} \check{x}_0 \\ \check{u}_1 \\ \cdots \\ \check{u}_n \end{bmatrix} \tag{8.9} $$

8.2 Commands for Building Models

8.2.1 BUILD

The `BUild` command traces all of the rays in a beam from the source to the specified final element. It then computes the full C-matrix linear model, giving the sensitivity of each ray at the last element to perturbations of each element in the beam train. It has one argument, the number `iElt` of the last element in the propagation.

`BUild` performs the MACOS differential ray-trace calculations, and must be performed before exercising commands that manipulate linear models, such as `LPE`, `LOP`, `LSP`, `LIN`, or before `EXPorting` C-matrix information.

Note that the `BUild` command ignores the presence of reference surfaces and obscuring surfaces unless they are the last surface. The `BUild` command currently does not work if there are return surfaces in the beam train.

As an example, consider the `Cassegrain.in` (see Appendix A.1.2). This prescription has output coordinates set up to compute a wavefront model as discussed previously. Once the prescription is loaded using the `OLD` command, the C-matrix is easily built:

```
MACOS>build
Enter terminal element number:4
Tracing 150 rays and BUILDing linear model...
Compute time was 0.4141 sec
MACOS>
```

8.2.2DMBUILD

The `DMBuild` command is not used in this version.

8.3 Commands for Changing Models

8.3.1 LPERTURB

The `LPErturb` command allows the user to enter an element perturbation for use with the linear model analysis commands. `LPErturb` takes different arguments depending on whether element input coordinates are to be used. The first argument `iElt` is the number of the element to be perturbed. Note that element 0 (the source) cannot be perturbed using `LPErturb`. The second argument is a yes-or-no response indicating whether input coordinates are to be used. If `NO` is selected, the next two prompts are for rotational and translational 3-vector perturbations. If `YES` is selected, a vector of the same length as the number of input coordinates is input in the input coordinate system.

After each `LPErturb` command, the current perturbation vector (in global coordinates) is printed out. The `LPErturb` command *adds* the perturbation data to the existing perturbation vector. Multiple `LPErturb` commands sum each perturbation with the previous perturbation vector. The perturbation vector can be reset to zero using the `LREset` command. Using `Cassegrain.in`:

```
MACOS>lperturb
Enter element to be perturbed:3
Use element coordinates?  (NO):
Enter rotational perturbation vector (x,y,z):0,1e-7,0
Enter translational perturbation vector (x,y,z):0,0,0


Element    1 Linear Perturbations:
Rotational=   0.000000D+00  0.000000D+00  0.000000D+00
Translational=   0.000000D+00  0.000000D+00  0.000000D+00
Element    2 Linear Perturbations:
Rotational=   0.000000D+00  0.000000D+00  0.000000D+00
Translational=   0.000000D+00  0.000000D+00  0.000000D+00
Element    3 Linear Perturbations:
Rotational=   0.000000D+00  1.000000D-07  0.000000D+00
Translational=   0.000000D+00  0.000000D+00  0.000000D+00
Element    4 Linear Perturbations:
Rotational=   0.000000D+00  0.000000D+00  0.000000D+00
Translational=   0.000000D+00  0.000000D+00  0.000000D+00
Element    5 Linear Perturbations:
Rotational=   0.000000D+00  0.000000D+00  0.000000D+00
Translational=   0.000000D+00  0.000000D+00  0.000000D+00
MACOS>
```

8.3.2 LPREAD

The LPRead command reads element perturbations from a specified datafile. MACOS assumes that the name of the perturbation file is infile.pert. If this file does not exist, MACOS prompts for the perturbation file name.

The perturbation file must be in the following all numeric format. The perturbations must be in global coordinates. The order of the perturbation vectors in the .pert-file must be identical to the order of the elements (the first perturbation vector corresponds to the first element, the sixth perturbation vector corresponds with the sixth element). Each element must have a perturbation vector. If an element is unperturbed, then a string of zeroes should be entered as its perturbation vector. The perturbation vectors have 6 components: a 3-vector rotational perturbations and a 3-vector translational perturbations. The 6 components must be separated by spaces.

```
MACOS>lpr
FileCassegrain.pert read.
MACOS>
```

8.3.3 LRESET

The LREset command zeros any previously entered linear perturbations. It should be used, when necessary, to reset the linear perturbation vector after LPErturb commands.

8.4 Commands for Exercisizing Models

8.4.1 PARTIALS

The PARtials command prints individual optical sensitivity matrices for specified rays. The user is asked whether to print in input/output or global coordinates. For Cassegrain.in, the partials of the chief ray in global coordinates are:

```
MACOS>partials
Enter ray number for partials (0=done, 1=chief ray):1
Print partials in actuator/sensor coordinates?  (YES): no
Partial of ray    1 at Element    4 (Ref_surf) to ray at Element    0 (Inpu-
tRay)
   0.3036E+01   0.0000E+00   0.0000E+00  -0.4458E-01   0.0000E+00   0.0000E+00
0.0000E+00
   0.0000E+00   0.3036E+01   0.0000E+00   0.0000E+00  -0.4458E-01   0.0000E+00
0.0000E+00
   0.0000E+00   0.0000E+00   0.1000E+01   0.0000E+00   0.0000E+00   0.0000E+00
0.0000E+00
   0.5549E+01   0.0000E+00   0.0000E+00   0.2479E+00   0.0000E+00   0.0000E+00
0.0000E+00
   0.0000E+00   0.5549E+01   0.0000E+00   0.0000E+00   0.2479E+00   0.0000E+00
0.0000E+00
   0.0000E+00   0.0000E+00   0.1006E+02   0.0000E+00   0.0000E+00   0.1000E+01
0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
0.1000E+01
Partial of ray    1 at Element    4 (Ref_surf) to Element    2 (Primary ) per-
turbations
   0.0000E+00  -0.6606E+01   0.0000E+00   0.6117E+00   0.0000E+00   0.0000E+00
   0.6606E+01   0.0000E+00   0.0000E+00   0.0000E+00   0.6117E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
```

```
   0.0000E+00  -0.8122E+01   0.0000E+00   0.7521E+00   0.0000E+00   0.0000E+00
   0.8122E+01   0.0000E+00   0.0000E+00   0.0000E+00   0.7521E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.2000E+01
Partial of ray    1 at Element    4 (Ref_surf) to Element    3 (Secondar) per-
turbations
   0.0000E+00   0.1241E+01   0.0000E+00  -0.5671E+00   0.0000E+00   0.0000E+00
  -0.1241E+01   0.0000E+00   0.0000E+00   0.0000E+00  -0.5671E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00  -0.2000E+01
Partial of ray    1 at Element    4 (Ref_surf) to Element    4 (Ref_surf) per-
turbations
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
Enter ray number for partials (0=done, 1=chief ray):
```

The specific partials take the form in Eqns. 8.3 and 8.4. The first matrix shows that source chief ray direction perturbations are magnified 3.036 times at the reference surface. The effect on ray direction at the reference surface of primary mirror tilts and decenters are magnified by 6.606 and 0.5671 times, respectively.

The path length partials show a factor-of-two sensitivity for this ray. Looking at the OPD partials only for a marginal ray, these numbers change:

```
Enter ray number for partials (0=done, 1=chief ray):2
Print partials in actuator/sensor coordinates?  (YES): y
Partial of ray    2 at Element    4 (Ref_surf) to ray at Element    0 (Inpu-
tRay)
   0.2996E+01  -0.2639E-17   0.1976E-16  -0.4450E-01   0.3920E-19   0.0000E+00
0.0000E+00
  -0.2639E-17   0.2984E+01   0.8899E-01   0.3920E-19  -0.4432E-01   0.0000E+00
0.0000E+00
   0.5643E+01  -0.4971E-17   0.1977E-15   0.2500E+00  -0.2202E-18   0.1976E-
16   0.0000E+00
  -0.4971E-17   0.5620E+01   0.8902E+00  -0.2202E-18   0.2490E+00   0.8899E-
01   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
0.1000E+01
Partial of ray    2 at Element    4 (Ref_surf) to Element    2 (Primary ) per-
turbations
  -0.4462E-16  -0.6401E+01   0.0000E+00   0.5827E+00  -0.8407E-17   0.4262E-16
   0.6200E+01   0.4462E-16   0.0000E+00  -0.8407E-17   0.5448E+00   0.1920E+00
  -0.1069E-15  -0.8239E+01   0.0000E+00   0.7500E+00  -0.1533E-16   0.7920E-16
   0.7757E+01   0.1069E-15   0.0000E+00  -0.1533E-16   0.6810E+00   0.3567E+00
  -0.3934E+01   0.8735E-15   0.0000E+00  -0.7951E-16  -0.3581E+00   0.1934E+01
Partial of ray    2 at Element    4 (Ref_surf) to Element    3 (Secondar) per-
turbations
  -0.3440E-17   0.1228E+01   0.5551E-16  -0.5382E+00   0.8367E-17  -0.4262E-16
  -0.1244E+01   0.3440E-17   0.0000E+00   0.8367E-17  -0.5005E+00  -0.1920E+00
   0.3146E-16  -0.6986E-32   0.0000E+00   0.3062E-32   0.1379E-16  -0.9888E-16
   0.1417E+00  -0.3146E-16   0.0000E+00   0.1379E-16   0.6210E-01  -0.4453E+00
   0.6140E+00  -0.1363E-15   0.0000E+00   0.5975E-16   0.2691E+00  -0.1930E+01
Partial of ray    2 at Element    4 (Ref_surf) to Element    4 (Ref_surf) per-
turbations
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
```

```
   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00   0.0000E+00
Enter ray number for partials (0=done, 1=chief ray):
```

8.4.2 LOPD

The LOPd command generates an OPD map of the beam at the final surface of the linear model. It parallels the OPD command discussed in Section 5.2.5. Using Cassegrain.in:

```
MACOS>lopd
Tracing perturbed system
Average OPD is  1.822578E-12
RMS OPD including piston is  3.023756E-08
RMS WFE excluding piston is  3.023755E-08
Type <RETURN> for next page:
```

8.4.3 LSPOT

The LSPot command generates a spot diagram of the beam at the final surface of the linear model. It parallels the SPOt command discussed in Section 5.2.6. Using Cassegrain.in:

```
MACOS>lspot
Tracing perturbed system
Average OPD is -9.502741E-07
RMS OPD including piston is  1.081200E-01
RMS WFE excluding piston is  1.081200E-01
Type <RETURN> for next page:
```

8.4.4 LINTENSITY

The LINtensity command computes beam wavefront phase at the final surface of the linear model and then performs a single far-field propagation to the next element in the prescription. If the last element in the linear model is the last element in the prescription, the user is prompted for transmission distance and other information. LINtensity performs calculations that parallel the propagation commands discussed in Sections 6.3 and 7.2.1, respectively. Using Cassegrain.in:

```
MACOS>lintensity
Tracing perturbed system
Average OPD is  1.822578E-12
RMS OPD including piston is  3.023756E-08
RMS WFE excluding piston is  3.023755E-08
FFT/Point Spread Function Data Summary:
Wavelength= 9.9999999748E-07;  Transmission Distance= 5.5601458549E+00
u-v plane diam= 1.8044868469E+01  du= 7.0764191449E-02
x-y plane diam= 7.8265948105E-05  dx= 3.0692527275E-07
Type <RETURN> for next page:
```

8.4.5 LPIXILATE

The LPIxilate command computes beam wavefront phase at the final surface of the linear model and then performs a single far-field propagation to the next element in the prescription. If the last element in the linear model is the last element in the prescription, the user is prompted for transmission distance and other information. Finaly, the image

is pixilated. The LPIxilate command parallels the PIxilate command discussed Section 6.3. Using Cassegrain.in:

```
MACOS>lpix
Tracing perturbed system
Average OPD is  1.822375E-12
RMS OPD including piston is  2.104591E-12
RMS WFE excluding piston is  1.052736E-12
FFT/Point Spread Function Data Summary:
Wavelength= 9.9999999748E-07;  Transmission Distance= 5.5601458549E+00
u-v plane diam= 1.8044868469E+01  du= 7.0764191449E-02
x-y plane diam= 7.8265948105E-05  dx= 3.0692527275E-07
Enter number of pixels per side:10
Enter size of pixel:1e-7
Type <RETURN> for next page:
```

8.5 Commands for Exporting Models

The EXPort command provides a way to export the C-Matrix of the linear model and is discussed in Section 3.8.5.

SECTION 9 Subroutine MACOS and Nonlinear Optical Models

Subroutine MACOS (SMACOS) provides the full functionality of MACOS in a Fortran subroutine. Regular MACOS commands are available to any other program via a Fortran CALL statement. This provides a nonlinear optics modeling capability for dynamic simulations or optimization programs. This section describes SMACOS in 4 subsections:

- Background
- Passing data between SMACOS and user applications
- Example
- Command reference

9.1 Background

Certain tasks require a full nonlinear optical modeling capability in a form that can be directly called by another program. Examples include

- Dynamic simulation of integrated optics/structures/controls systems where structural motions determine optical outputs such as focal-plane images and drive controller actions
- Monte Carlo studies of optical systems where uncertain parameters are varied randomly, collected, and analyzed statistically
- Optical parameter estimation where optical model parameters are determined by driving modelled optical outputs, such as images, to match actual measured outputs
- Design optimization where sets of parameters are varied to extremalize an optical performance metric

SMACOS consists of source code files, object code files, and include files that are compiled and linked with the user's main program.

To use SMACOS, several steps must be followed:

1. Develop .in-file optical prescriptions for each system to be modeled
2. In the user's main program, use Fortran include statements to define SMACOS variables (see Section 9.2)
3. Set up the sequence of commands in the main program using SMACOS subroutine calls
4. Compile, link, and execute the code

The process of using SMACOS begins with the creation of standard MACOS .in-file optical prescriptions. This can be done interactively using the standard MACOS application as described in Section 4 or an existing .in-file can be modified using a text editor.

The prescription should be checked to verify that the light PROpagates through the systems and the expected images are produced.

SMACOS normally includes the standard MACOS graphics. In some cases this may not be desirable: these routines take up appreciable memory in the computer. An alternative library of routines is available for applications that do not require graphical output.

9.2 Call Line Variables

The user's main program must define the SMACOS call line dummy variables. SMACOS is called:

```
      CALL SMACOS(command,CARG,DARG,IARG,LARG,RARG,
&     OPDMat,RaySpot,RMSWFE,PixArray)
```

The variables are explained in below.

9.2.1 Dummy Command Variables

SMACOS calls using the following dummy variables to pass the arguments needed to carry out SMACOS commands:

1. The SMACOS dummy character variable command is set to the MACOS command name: `command='OLD'`
2. Any character arguments are stored in the SMACOS character argument array: `CARG(1)='Cassegrain'`
3. Any logical arguments are stored in the SMACOS logical argument array: `LARG(1)=.TRUE.`
4. Any double-precision arguments are stored in the SMACOS double precision argument array: `DARG(1)`=3.1415926d0
5. Any single-precision arguments are stored in the SMACOS double precision argument array: `RARG(1)`=3.1416e0
6. Any integer arguments are stored in the SMACOS integer argument array: `IARG(1)`=12

These dummy variables are dimensioned:

```
CHARACTER*132 command
CHARACTER*32 CARG(9)
REAL*8 DARG(9))
INTEGER IARG(9)
LOGICAL LARG
REAL*4 RARG(9)
```

9.2.2 Dummy Output Variables

SMACOS uses the following dummy variables to pass output:

1. `OutRaySpot`: spot diagram data generated by `SPOt` and `LSPot`
2. `RMSWFE`: RMS wavefront error generated by `OPD` and `LOPd`
3. `OPDMat`: wavefront generated by `OPD` and `LOPd`
4. `PixArray`: image data generated by `ADD, BLUR, PIX, LPIX` and other commands

These dummy variables are dimensioned:

```
REAL*8 RaySpot(mRay, 2), OPDMat(mpts, mpts), RMSWFE
REAL*4 PixArray(mPix, mPix)
```

9.2.3 Common Blocks and Include Files

An alternate means of addressing SMACOS data structures is via the various SMACOS common blocks. Inclusion of the SMACOS commons allows the user to directly modify data through Fortran assignment statements in the calling program. It also allows access to all of the data products generated by SMACOS analysis functions, not just those in the call line.

Access to the SMACOS commons from a user-written Fortran program can be provided by including the provided `smacosvars.cmn` file. This file is shown here:

```
C**********************************************************************
C       Begin file smacosvars.cmn
C    +-------------------------------------------------------------+
C    |  Copyright (C) 1995-9, California Institute of Technology.  |
C    |  U.S. Government Sponsorship under NASA Contract NAS7-918   |
C    |  is acknowledged.                                           |
C    +-------------------------------------------------------------+
C**********************************************************************

C   This file defines call-line variables and common blocks that give
C   user programs direct access to MACOS variables. The form of the
C   SMACOS call line is:

C         CALL SMACOS(command,CARG,DARG,IARG,LARG,RARG,
C    &    OPDMat,RaySpot,RMSWFE,PixArray)

C   The param.cmn file defines array sizes for SMACOS call-line and
C   internal MACOS variables. The param.cmn file should be copied from
C   param128.cmn, param256.cmn, etc. -- whichever size of SMACOS you
C   are using:

        INCLUDE 'param.cmn'

C   These are the SMACOS call-line variables:

        CHARACTER*132 command
        CHARACTER*32 CARG(9)
        REAL*8 DARG(9), RaySpot(mRay, 2), OPDMat(mpts, mpts)
        INTEGER IARG(9)
        LOGICAL LARG
        REAL*4 RARG(9), RMSWFE, PixArray(mPix, mPix)

C   These are other SMACOS common blocks, that you may use or comment
C   out as you wish. The ifEcho flag, if set to .TRUE., provides for
C   full printout of SMACOS operations. If set to .FALSE., reduced
C   printout is provided.

        LOGICAL ifEcho
        COMMON /SCIO/ifEcho

        INTEGER npts,NoiseSeed(2),iSpot
        REAL*8 xcent,ycent,xLocal(3),yLocal(3),zLocal(3),
     &  CntrSpot(3),RefSpot(2)
        COMMON /UserCommon4/npts,NoiseSeed,iSpot
        COMMON /UserCommon8/xcent,ycent,xLocal,yLocal,zLocal,
     &  CntrSpot,RefSpot

C   These includes provide access to internal MACOS variables. They can
```

```
C   be commented out if you do not plan to use them:

        INCLUDE 'elt.cmn'
        INCLUDE 'src.cmn'

C   These includes provide access to internal MACOS scratch arrays. They
C   should be commented out if you do not plan to use them:

        LOGICAL L1(md2)
        INTEGER DrawEltVec(mDrawElt,mDrawRay)
        REAL*4 R1(mdttl,mdttl),RV1(md2),RV2(md2)
        REAL*8 D1(mdttl,mdttl),D2(mdttl,mdttl)
        REAL*8 DV1(md2),DV2(md2)
        COMMON /SCRATCH/ D1,D2
        EQUIVALENCE (D1(1,1),RV1(1)),(D1(1,1),DV1(1)),(D2(1,1),DV2(1)),
     &  (D1(1,1),R1(1,1)),(D2(1,1),RV2(1))

c END of smacosvars.cmn

C*********************************************************************
```

Most of the useful data is buried in the elt.cmn and src.cmn include files invoked above. The elt.cmn file is shown below:

```
C*********************************************************************
C       Begin file elt.cmn
C    +-------------------------------------------------------------+
C    |  Copyright (C) 1995-9, California Institute of Technology.  |
C    |  U.S. Government Sponsorship under NASA Contract NAS7-918   |
C    |  is acknowledged.                                           |
C    +-------------------------------------------------------------+
C*********************************************************************

C   Variable definitions

        LOGICAL LRayToElt(0:mElt,bRay),LRayOK(mRay),LEltOK(mElt),
     &  LRayPass(mRay),LEltObs(mElt),ifComputeDZ(mIntSrf),LGlass(mElt)

        CHARACTER*16 EltName(0:mElt),EltTypeName(20),SrfTypeName(20),
     &  ApTypeName(0:2),ObsTypeName(-5:5),PropTypeName(11),
     &  GridTypeName(5),ZernTypeNameL(3),GlassElt(mElt),BaseUnits,
     &  WaveUnits,GlassName(mGlass)

        INTEGER nElt,nRay,EltType(0:mElt),nECoord(0:mElt),nOutCord,
     &  nSeg,SegCoord(3,mElt),EltToSegMap(mElt),SegToEltMap(mEm1),
     &  RayToSegMap(mRay,mSegMirs),RayID(mdttl,mdttl),RayWFMap(2,mRay),
     &  PropType(0:mElt),ApType(mElt),nObs(mElt),ObsType(mObs,mElt),
     &  iEltToiWF(mElt),SrfType(mElt),OpType(mElt),iEltToIntSrf(mElt),
     &  IWORK(LIWORK,mIntSrf),nDP(mIntSrf),LensArrayType(mElt),
     &  nCoatElt(mElt),UDSrfType(0:mElt),EltID(0:mElt),ZernaType(mElt),
     &  ZernTypeL(mElt),nRayToSeg(mRay),iEltToGridSrf(mElt),
     &  nGridMat(mElt),spcOption

        REAL*4 dxidxim1(7,7,mElt),dxidui(7,6,mElt),
     &  MWFFT(mdttl,mdttl),XFP(mdttl),YFP(mdttl)

        REAL*8 Wavelen,VptElt(3,mElt),RptElt(3,mElt),fElt(mElt),Flux,
     &  eElt(mElt),psiElt(3,mElt),RayPos(3,mRay),
     &  RayDir(3,mRay),RayL(mRay),dxElt(0:mElt),
     &  IndRef(0:mElt),CumRayL(mRay),TElt(6,6,mElt),Tout(7,7),
     &  ApVec(6,mElt),ObsVec(6,mObs,mElt),xObs(3,mElt),zElt(0:mElt),
     &  KcElt(mElt),KrElt(mElt),AsphCoef(4,mElt),MonCoef(120,mElt),
     &  ZernCoef(45,mElt),pMon(3,mElt),xMon(3,mElt),yMon(3,mElt),
     &  zMon(3,mElt),lMon(mElt),zSource,XYZSRF(mDP,3,mIntSrf),
```

```
     &  DZSRF(2,mDP,mIntSrf),WavelenUM,WavelenConv,
     &  LensArrayWidth(mElt),h1HOE(3,mElt),h2HOE(3,mElt),
     &  OrderHOE(mElt),WaveHOE(mElt),PinHole(mElt),Extinc(0:mElt),
     &  CoatIndxElt(20,mElt),CoatThkElt(20,mElt),AnaCoef(4,mElt),
     &  RuleWidth(mElt),UDSrfCoef(1500,mElt),yObs(3,mElt),zObs(3,mElt),
     &  DWORK(mDP,mIntSrf),UDSrfParam(10,mElt),
     &  GridMat(mGridMat,mGridMat,mGridSrf),GridSrfdx(mGridSrf),
     &  GlassCoef(6,mElt),GlassTable(6,mGlass),WaveBU,CBM,CWM,CWB

        COMPLEX*16 WFElt(mdttl,mdttl,mWF),RayE(3,mRay),
     &  WFbuff(mdttl,mdttl),JmatElt(2,2,mElt)

C   COMMON block definitions

        COMMON /EltLog/ LRayToElt,LRayOK,LEltOK,LRayPass,LEltObs,
     &  ifComputeDZ,LGlass

        COMMON /EltChar/ EltName,EltTypeName,SrfTypeName,
     &  ApTypeName,ObsTypeName,PropTypeName,GridTypeName,
     &  GlassElt,ZernTypeNameL,BaseUnits,WaveUnits,GlassName

        COMMON /EltInt/ nElt,nRay,EltType,nECoord,nOutCord,nSeg,
     &  SegCoord,EltToSegMap,SegToEltMap,RayToSegMap,
     &  RayID,RayWFMap,PropType,ApType,ObsType,nObs,
     &  iEltToiWF,SrfType,OpType,iEltToIntSrf,IWORK,nDP,LensArrayType,
     &  nCoatElt,UDSrfType,EltID,ZernTypeL,nRayToSeg,iEltToGridSrf,
     &  nGridMat,spcOption

        COMMON /EltReal/ dxidxim1,dxidui,MWFFT,XFP,YFP

        COMMON /EltDble/ Wavelen,VptElt,RptElt,fElt,eElt,psiElt,
     &  RayPos,RayDir,RayL,IndRef,CumRayL,TElt,Tout,Flux,ApVec,ObsVec,
     &  xObs,dxElt,zElt,KcElt,KrElt,AsphCoef,MonCoef,ZernCoef,pMon,
     &  xMon,yMon,zMon,lMon,zSource,XYZSRF,DZSRF,LensArrayWidth,
     &  h1HOE,h2HOE,OrderHOE,WaveHOE,RuleWidth,
     &  PinHole,Extinc,CoatIndxElt,CoatThkElt,AnaCoef,
     &  UDSrfCoef,yObs,zObs,DWORK,UDSrfParam,GridMat,GridSrfdx,
     &  WavelenUM,WavelenConv,GlassCoef,GlassTable,WaveBU,CBM,CWM

        COMMON /EltCmplx/ WFElt,RayE,JmatElt

        SAVE /EltCmplx/
        SAVE /EltDble/
        SAVE /EltReal/
        SAVE /EltInt/
        SAVE /EltLog/
        SAVE /EltChar/

C End Element Definition Include File - 'elt.cmn'
C*****************************************************************
```

Some of the important variables in this common file have been identified elsewhere in this manual. See Section 4.11 for a summary of variables used in the basic optical prescription. Variables containing the data generated by ray-trace, diffraction, and linearized ray-trace commands include:

- `CumRayL(iRay)`: Cumulative ray optical pathlength at the current element
- `RayDir(iRay)`: Ray direction at the current element
- `RayPos(iRay)`: Ray position at the current element

- `WFElt(i,j,iWF)`: Complex amplitude at the current element
- `dxElt(iElt)`: Ray spacing at the current element
- `Cmatrix`: Matrix of ray partial derivatives at the current element

9.3 Specification of Model Size in SMACOS

A new subroutine is added to MACOS 3.2b, as part of the SMACOS library, that allows the user to change model size at runtime. The subroutine call syntax is

CALL SMACOS_Set_Model_Size(model_size)

where the argument *model_size* is a either a constant or a variale for the specified model size. The call to SMACOS_Set_Model_Size will release the dynamic storage used for the existing model size, and allocate storage for the new model size. It also does some necessary setup to make sure a proper SMACOS initialization is performed when the model size is changed. The model parameter file *macos_param.txt* must be accessible from the user's execution directory as discussed in Section 3.3.

9.4 Example

The user code accesses MACOS functions using the SMACOS call line in a fashion analogous to the MACOS command line. For instance, the MACOS command sequence to load `Cassegrain.in` and compute the intensity at the final element is:

```
MACOS>old
Enter file name:  Cassegrain
Input file Cassegrain.in being loaded.
Optical Train Summary:
 Tracing rays past   5 elements.
 Aperture type =   1 (Annular             ) with  15 grid points.
 Elt   1: Obscuration          with Kc= 0.0000D+00, Kr=-1.0000D+20, nECoords
=  -6
 Elt   2: Conic Mirror         with Kc=-1.0000D+00, Kr=-1.0800D+01, nECoords
=  -6
 Elt   3: Conic Mirror         with Kc=-2.6706D+00, Kr=-3.5268D+00, nECoords
=  -6
 Elt   4: Reference Surface    with Kc= 0.0000D+00, Kr=-5.5601D+00, nECoords
=  -6
 Elt   5: Focal Plane          with Kc= 0.0000D+00, Kr=-1.0000D+18, nECoords
=  -6
 Number of output coordinates is   5
MACOS>intensity
Enter last element for propagation:5
Tracing    4396 rays and propagating   65536 grid points...
 FF Prop between Elt  4 and Elt  5 to WF  1:
 z1= 5.5601D+00 dx1= 7.0764D-02 min= 7.0704D-02 max= 7.0825D-02 dev= 3.5308D-
05 lin= 99.95%
 z2= 0.0000D+00 dx2= 3.0693D-07
Compute time was 0.3236 sec
MACOS>
```

The same functions can be performed using SMACOS calls as follows:

```
C  Load Cassegrain.in prescription file
        command='OLD'
        CARG(1)='Cassegrain'
        CALL SMACOS(command,CARG,DARG,IARG,RARG,
     &  OutRaySpot,NomRayDir,RMSWFE,NomOPDMat,PertOPDMat,
     & PixArray,Cmatrix)
```

```
C  Propagate
       command='INTENSITY'
       IARG(1)=5
       CALL SMACOS(command,CARG,DARG,IARG,RARG,
     &  OutRaySpot,NomRayDir,RMSWFE,NomOPDMat,PertOPDMat,
     & PixArray,Cmatrix)
```

The SMACOS call line passes results of some calculations in dummy output arrays (see Section 9.2.2).

9.5 Commands

Every MACOS command is available as an SMACOS command, with the exception of NEW. The MOD command has a different form in SMACOS than in MACOS, as it is expected that the SMACOS user will prefer to directly access and change any data values, rather than incur the overhead of a full SMACOS call. The SMACOS MOD command simply resets the current ray-trace and WF element counters to 0, so that subsequent computations will start from the source. It has no arguments.

One command is available for SMACOS that is not used in MACOS: the NONe command, which turns off the MACOS plot functions.

This section summarizes most of the SMACOS commands and their arguments.

9.5.1 File and Data Manipulation

The SMACOS commands for loading .in-files, displaying element data and exporting data are:

1. LOAD or OLD: load existing optical system dataset. This command has one argument, the name of the prescription file (minus the .in suffix). The command is set up in the main program as follows:
   ```
   command='OLD'
   CARG(1)=filename
   ```
   and is executed using the call statement:
   ```
   CALL SMACOS(command,CARG,DARG,IARG,RARG,…)
   ```
   The call is the same for all commands and so is not repeated for each.
2. Help: display a summary of the SMACOS commands. This command has no arguments. The command is set up in the main program as follows:
   ```
   command='HELP'
   ```
3. SUMmarize: display a summary of the optical system data on the user's terminal. This command has no arguments. It is set up:
   ```
   command='SUMMARIZE'
   ```
4. STAtus: display a summary of the status of the current calculations. This command has no arguments. It is set up:
   ```
   command='STATUS'
   ```
5. RESet: reset various internal flags and counters to their initial values. This has the effect of forcing the next calculations to be performed as if the current file was just loaded. This command has no arguments. It is set up:
   ```
   command='RESET'
   ```

6. SHOw: display the data defining a specified element. This command has 1 integer argument: the number of the element to be displayed. The source is considered to be element 0. It is set up:
```
command='SHOW'
IARG(1)=iElt
```
7. SAVe: write out the current optical data as a new .in-file. This command has 1 character argument: the name of the new file. It is set up:
```
command='SAVE'
CARG(1)=new_filename
CARG(2)='YES' (to overwrite if file exists)
```
8. EXPort: this function writes specified data to disk using one of three formats: MATLAB ".m-file", fits, or Fortran binary formats. It is set up differently depending on the particular variable to be exported, its form, etc. The user is advised to prototype the particular command sequence using MACOS, and then set up the appropriate sequence of arguments.
```
command='EXPORT'
IARG(1)=…
CARG(1)=…
```
9. EXEcute: runs a journal file. This command has 1 character argument, namely the name of the journal file to be executed. The command is set up in the main program as follows:
```
command='EXECUTE'
CARG(1)='journal_file_name'
```
10. JOUrnal: writes a journal file. This command has 1 character argument, namely the name of the journal file to be written. The command is set up as follows:
```
command='JOURNAL'
CARG(1)='journal_file_name'
CARG(2)='YES' (to overwrite if file exists)
```
11. MODify: resets the surface counters for ray-trace and diffraction computations, so that changes to elements will be picked up. This command has no arguments. It is set up as follows:
```
command='MODIFY'
```

Note that the SMACOS MODify command (unlike the MACOS MODify command) does not change any data elements. It is up to the user to do this directly, by assignment statements in the calling program.

9.5.2 Ray-Trace Analysis

1. MAP: this command displays and prints a cross-sectional map of the source light beam, as described in Section 5.2.2. It has no arguments. The command is set up as follows:
```
command='MAP'
```
2. RAY: trace a specified ray through the entire system. This command has one argument, the ray number. The command is set up as follows:
```
command='RAY'
IARG(1)=iRay
```
3. FFP: find field point corresponding to a specified chief ray offset at a particular optical element. This command has 4 arguments: the element number of the surface at which the chief ray intercept is specified, the offset of the chief ray from the vertex of the element, and a prompt response to accept the new source parameters. The command is set up as follows:
```
command='FFP'
IARG(1)=iElt
DARG(1)=chief ray offset x in base units
DARG(2)=chief ray offset y in base units
CARG(1)='YES' (to accept changes)
```
4. PFP: find field point corresponding to a specified chief ray offset in pixel coordinates at a detector. This command has 4 arguments: the element number of the surface at

which the chief ray intercept is specified, the size of the detector pixels, the offset of the chief ray from the vertex, and a prompt response to accept the new source parameters. The command is set up as follows:

```
command='PFP'
IARG(1)=iElt
DARG(1)=pixel size in base units
DARG(2)=chief ray offset x in pixel units
DARG(3)=chief ray offset y in pixel units
CARG(1)='YES' (to accept changes)
```

5. `CENter`: center the beam in the aperture stop. This command has three arguments: the element number of the aperture stop and its offset from the vertex. The command is set up as follows:

```
command='CENTER'
IARG(1)=iElt
DARG(1)=stop offset x
DARG(2)=stop offset y
CARG(1)='YES' (to accept changes)
```

6. `POLarization`: turn on polarization ray-tracing. This command has 4 arguments specifying the polarization state of the source rays. The command is set up as follows:

```
command='POL'
DARG(1)=TEM X (real part)
DARG(2)=TEM X (imaginary part)
DARG(3)=TEM Y (real part)
DARG(4)=TEM Y (imaginary part)
```

7. `NOPolarization`: turn off polarization ray-tracing. This command has no arguments. The command is set up as follows:

```
command='NOPOL'
```

8. `PERturb`: rotate and/or translate a specified optical element. This command has several arguments. An integer argument is used to specify the number of the element to be perturbed (element 0 is the source). Up to 9 double precision arguments are used to specify the perturbations. A character argument is used to specify global or element coordinates. The command is set up as follows:

```
command='PERTURB'
IARG(1)=iElt
```

If `iElt=0` (the source is to be perturbed) then the perturbation is entered in global coordinates

```
DARG(1)=rotational perturbation x
DARG(2)=rotational perturbation y
DARG(3)=rotational perturbation z
DARG(4)=translational perturbation x
DARG(5)=translational y
DARG(6)=translational z
```

and the system aperture stop location is entered:

```
DARG(7)=stop x
DARG(8)=stop y
DARG(9)=stop z
```

If `iElt` is not zero (an optical element is to be perturbed) then the coordinate system for the perturbation is specified using a character argument:

```
CARG(1)=eltcoord or globalcoord
```

and if element coordinates were selected, the perturbations are :

```
DARG(1)=perturbation in element coordinate 1
DARG(2)=perturbation in element coordinate 2
DARG(3)=perturbation in element coordinate 3
DARG(4)=perturbation in element coordinate 4
DARG(5)=perturbation in element coordinate 5
```

```
DARG(6)=perturbation in element coordinate 6
```

or if global coordinates were selected:

```
DARG(1)=rotational perturbation x
DARG(2)=rotational perturbation y
DARG(3)=rotational perturbation z
DARG(4)=translational perturbation x
DARG(5)=translational perturbation y
DARG(6)=translational perturbation z
```

9. PREad: reads perturbations from an input file. This command has 1 argument, the name of the files. The command is set up as follows:

```
command='PREAD'
CARG(1)=filename
```

10. OPD: trace the full set of rays from the source (or from the current element if an OPD, SPOt, or PROpagate command has been performed previously) up to a specified element, and then calculate the OPD distribution across that element. The result is stored in the array NomOPDMat and passed in the SMACOS call line. Depending on the selected plot option, the OPD map may be plotted. This command has one argument, the element number. The command is set up as follows:

```
command='OPD'
IARG(1)=iElt
CARG(1)='YES' (to overwrite if a file-writing plot type is selected and a file of the same name exists)
```

11. SPOt: trace the full set of rays from the source (or from the current element if an OPD, SPOt, or PROpagate command has been performed previously) up to a specified element, and then calculate the spot diagram at that element. The result is stored in the array OutRaySpot and passed in the SMACOS call line. Depending on the selected plot option, the spot diagram may be plotted. The way obscured rays are handled depends on the setting of the OBS option. This command has one argument, the element number. The command is set up as follows:

```
command='SPOT'
IARG(1)=iElt
CARG(1)='YES' (to overwrite if a file-writing plot type is selected and a file of the same name exists)
```

The SPOt command expects 5 output coordinates.

12. OBScuration: specify the way obscured rays are handled for SPOt diagrams. This command has one argument, the obscuration option number. If this is set to 0, all rays are plotted; if set to 1, only unobscured rays are plotted; if set to 2, only obscured rays are plotted. The command is set up as follows:

```
command='OBS'
CARG(1)=Obscuration option
```

13. SINT: activates the surface interpolation algorithms. This command has no arguments. The command is set up as follows:

```
command='SINT'
```

9.5.3 Diffraction Analysis

1. INTensity: trace the full set of rays and propagate the diffraction wavefront from the source (or from the current element if a diffraction command has been performed previously) up to a specified element. The resulting complex amplitude is stored in the array WFElt, which is one of the variables defined in the include files. The modulus squared is plotted; it is stored in the MWFFT array. This command has one argument, the element number. The command is set up as follows:

```
command='INTENSITY'
IARG(1)=iElt
CARG(1)='YES' (to overwrite if a file-writing plot type is selected and a file of the same name exists)
```

2. `GAIn`: compute and plot the ratio of the light diffracted by the optical system to a point source. This command has one argument, the element number at which to comput the gain. The command is set up as follows:
```
command='GAIN'
IARG(1)=iElt
CARG(1)='YES'  (to overwrite if a file-writing plot type is selected and a file of the same name exists)
```
3. `SCAlar`: calculate using scalar diffraction algorithm, the default. This command has no arguments. The command is set up as follows:
```
command='SCALAR'
```
4. `VECtor`: calculate using scalar diffraction algorithm, the default. This command has no arguments. The command is set up as follows:
```
command='VECTOR'
```
5. `ORS`: optimize a reference surface for sphere-to-sphere diffraction. This command finds the best-fit sphere to the wavefront at a specified reference surface element, which is then reset to match the best-fit sphere. This command has one argument, the reference surface element number. The command is set up as follows:
```
command='ORS'
IARG(1)=iElt
```
6. `SRS`: slave a reference surface to a previously optimized reference surface for sphere-to-sphere near-field diffraction or for sphere-to-plane far-field diffraction. This command has two arguments: the element number of the surface to be slaved; and the element number of the surface to which it is to be slaved. It uses the parameters defining the second element to reset the first so that the diffraction is correctly defined. The command is set up as follows:
```
command='SRS'
IARG(1)=iSlv1
IARG(2)=iSlv2
```
7. `FEX`: find the exit pupil of the system. This command traces two chief rays through the aperture stop of the system to the element preceeding the specified element and finds their crossing point in image space. The specified surface is then reset to be a sphere located at the crossing point and centered at the intersection of the source chief ray with the preceeding element. Both the specified element and the one preceeding it must be return surfaces. The command has two arguments: the number of the element to be set to the exit pupil and the location of the system aperture stop in global coordinates. The command is set up as follows:
```
command='FEXIT'
IARG(1)=iElt
CARG(1)='YES'  (to accept changes)
```
8. `REGrid`: toggle the "regrid" option for sphere-to-sphere propagations. With regridding set on, the ray grid is recalculated after each near-field propagation to assure correct alignment of the rays with the diffraction wavefront grid points. This command is most useful in highly aberrated systems. In some circumstances it must be used with the OPD interpolation command (not currently implemented). The command has no arguments. It is set up as follows:
```
command='REGRID'
```
9. `BEAm`: allows the user to select between a uniform beam and a Gaussian beam. This command has three arguments: the beam type (uniform or Gaussian), the x beam diameter (for a gaussian beam) and y beam diameter (for a Gaussian beam). The command is set up as follows:
```
command='BEAM'
IARG(1)=BeamType
```

```
DARG(1)=x beam diamter
DARG(2)=y beam diamter
```

9.5.4 Wavefront and Image Plots

1. INtensity: compute and plot the intensity of the wavefront at the specified element. Plotting is done only if plot type is not NONe and the graphics options are included in the compile. The type of plot is as set by previous calls to SMACOS. This command has one argument, the element number. The command is set up as follows:
   ```
   command='IN'
   IARG(1)=iElt
   CARG(1)='YES' (to overwrite if a file-writing plot type is selected and a file of the same name exists)
   ```
2. AMplitude: compute and plot the amplitude and phase of the wavefront at the specified element. This command has one argument, the element number. The command is set up as follows:
   ```
   command='AM'
   IARG(1)=iElt
   CARG(1)='YES' (to overwrite if a file-writing plot type is selected and a file of the same name exists)
   ```
3. REAl: compute and plot the real and imaginary parts of the wavefront at the specified element. This command has one argument, the element number. The command is set up as follows:
   ```
   command='REAL'
   IARG(1)=iElt
   CARG(1)='YES' (to overwrite if a file-writing plot type is selected and a file of the same name exists)
   ```
4. LOG: compute and plot the logarithm (base 10) of the intensity of the wavefront at the specified element. This command has one argument, the element number. The command is set up as follows:
   ```
   command='LOG'
   IARG(1)=iElt
   CARG(1)='YES' (to overwrite if a file-writing plot type is selected and a file of the same name exists)
   ```
5. PIxilate: compute and plot the pixilated wavefront intensity at the specified element. This command has three arguments: the element number, the number of pixels across one side of the array (assumed square), and the size of a pixel. The minimum size of the pixel is half the wavefront grid spacing. This is not automatically checked, so be careful! The PIxilated image is stored in the PixArray variable and passed through the SMACOS call line. The PLOcate command sets the option to locate the center of the pixel array to the vertex point of the element. The command is set up as follows:
   ```
   command='PIXILATE'
   IARG(1)=iElt
   IARG(2)=number of pixels per side
   RARG(1)=size of pixel
   ```
6. COMpose: initiate a composed image to which subsequent intensity plots can be ADDed. This command has three arguments: the element number, the number of pixels across one side of the array (assumed square), and the size of a pixel. The minimum size of the pixel is half the wavefront grid spacing. This is not automatically checked, so be careful! The command is set up as follows:
   ```
   command='COMPOSE'
   IARG(1)=iElt
   IARG(2)=number of pixels per side
   RARG(1)=size of pixel in base units
   ```
7. WINdow: define a window on the detector that stays fixed wrt the detector. Subsequent images that fall on the window, as determined by the ray trace, are placed appropriately, provided the FEX command is used for each image. Used with the PIX,

COMPOSE, ADD, etc. commands. This command has 5 arguments. It is set up as follows:

```
command='WINDOW'
CARG(1)='TOUT' (detector coordinates option)
DARG(1)=size of pixel in base units
DARG(2)=x element vertex location in pixel coords
DARG(3)=y element vertex location in pixel coords
DARG(4)=x window center location in pixel coords
DARG(5)=y window center location in pixel coords
```

8. ADD: add an intensity plot to a composed image. This command has two arguments: the element number and a character argument to specify whether to plot the composed image. The composed image is stored in the PixArray variable and passed through the SMACOS call line. The command is set up as follows:

```
command='ADD'
IARG(1)=iElt
CARG(1)='PLOT' or 'NOPLOT'
CARG(2)='YES' (to overwrite if a file-writing plot type is selected and a file of the same name exists)
```

9. DADD: display the currently composed image. This command has no arguments. The command is set up as follows:

```
command='DADD'
CARG(1)='YES' (to overwrite if a file-writing plot type is selected and a file of the same name exists)
```

10. NOISE: add detection noise to the currently composed image. Noise here means random fluctuations in the values of the pixels of an image. This command has several arguments, which vary with context. The command is set up for Gaussian noise as follows:

```
command='NOISE'
CARG(1)='GAUSSIAN'
DARG(1)=standard deviation of noise in DN
CARG(2)='PLOT' or 'NOPLOT'
CARG(3)='YES' (to overwrite if a file-writing plot type is selected and a file of the same name exists)
```

For photon noise, a Poisson process where the standard deviation is the expected value:

```
command='NOISE'
CARG(1)='PHOTON'
CARG(2)='PLOT' or 'NOPLOT'
CARG(3)='YES' (to overwrite if a file-writing plot type is selected and a file of the same name exists)
```

For Poisson noise on a uniform background level:

```
command='NOISE'
CARG(1)='POISSON'
RARG(1)=background level in DN
DARG(1)=detector quantum efficiency (in the range 0 to 1)
CARG(2)='PLOT' or 'NOPLOT'
CARG(3)='YES' (to overwrite if a file-writing plot type is selected and a file of the same name exists)
```

11. BLUR: blur the currently composed image using a 9-pixel kernel. This command is intended to simulate detector charge leakage. It has one argument, the blur kernel peak, which can take values between 1 and 0. The command is set up as follows:

```
command='BLUR'
RARG(1)=blur kernel peak
CARG(1)='PLOT' or 'NOPLOT'
CARG(2)='YES' (to overwrite if a file-writing plot type is selected and a file of the same name exists)
```

12. GBLUR: propagate and blur an image, then add to the currently composed image. This command is more general than the BLUR command, using a Gaussian kernel. It is often used to simulate long-exposure jitter blurring. The command is set up as follows:

```
command='GBLUR'
DARG(1)=Gaussian blur kernel width in base units
CARG(1)='PLOT' or 'NOPLOT'
CARG(2)='YES' (to overwrite if a file-writing plot type is selected and a
file of the same name exists)
```

9.5.5 Plot Types

1. NONe: turns off plotting entirely. The command is set up as follows:
   `command='NONE'`
2. NULl: turns off plotting entirely. The command is set up as follows:
   `command='NULL'`
3. WIRe: sets surface plot type to "wireframe". The command is set up as follows:
   `command='WIRE'`
4. SLIce: sets surface plot type to "slice". The command is set up as follows:
   `command='SLICE'`
5. GRAy: sets surface plot type to "gray scale". The command is set up as follows:
   `command='GRAY'`
6. CONtour: sets surface plot type to "contour plot". The command is set up as follows:
   `command='CONTOUR'`
7. ROW: sets surface plot type to "row plot". The command is set up as follows:
   `command='ROW'`
8. COLumn: sets surface plot type to "column plot". The command is set up as follows:
   `command='ROW'`
9. TEXt: sets surface plot type to "text". This provides for the plot data to be written to a text file instead of being plotted. The command is set up as follows:
   `command='TEXT'`
10. BINary: sets surface plot type to "binary". This provides for the plot data to be written to a binary file instead of being plotted. The command is set up as follows:
    `command='BINARY'`
11. MATlab: sets surface plot type to "Matlab". This provides for the plot data to be written to a Matlab .m-file instead of being plotted. The command is set up as follows:
    `command='MATLAB'`
12. FITs: sets surface plot type to "fits". This provides for the plot data to be written to a fits file instead of being plotted. The command is set up as follows:
    `command='FITS'`

9.5.6 Linear Models

1. BUild: trace the full set of rays and the ray partial derivative matrices from the source up to a specified element, and then compute the composite C-matrix. The result is stored in the array Cmatrix and passed in the SMACOS call line. This command has one argument, the element number. The command is set up as follows:

```
command='BUILD'
IARG(1)=iElt
```

2. DMBuild: performs as the DMBuild, but also includes interpolated surface partials. The result is stored in the array C-matrix and passed in the SMACOS call line. This command has one argument, the element number. The command is set up as follows:

```
command='DMBUILD'
IARG(1)=iElt
```

3. PArtials: displays and prints the full set of ray partial derivatives for a specified ray. The command is set up as follows:

```
command='PARTIALS'
IARG(1)=iRay
```

4. `LPErturb`: rotate and/or translate a specified optical element. This command has several arguments. An integer argument is used to specify the number of the element to be perturbed (`LPErturb` does not work with element 0). Up to 9 double precision arguments are used to specify the perturbations. A character argument is used to specify global or element coordinates. The command is set up as follows:

   ```
   command='LPERTURB'
   IARG(1)=iElt
   ```

   and the coordinate system for the perturbation is specified using a character argument:

   ```
   CARG(1)=eltcoord or globalcoord
   ```

   If element coordinates are choosen:

   ```
   DARG(1)=perturbation in element coordinate 1
   DARG(2)=perturbation in element coordinate 2
   DARG(3)=perturbation in element coordinate 3
   DARG(4)=perturbation in element coordinate 4
   DARG(5)=perturbation in element coordinate 5
   DARG(6)=perturbation in element coordinate 6
   ```

   else if global coords are choosen:

   ```
   DARG(1)=rotational perturbation x
   DARG(2)=rotational perturbation y
   DARG(3)=rotational perturbation z
   DARG(4)=translational perturbation x
   DARG(5)=translational perturbation y
   DARG(6)=translational perturbation z
   ```

5. `LPRead`: rotate and/or translate a specified optical element using data stored in a separate file. This command has no arguments. It assumes that properly formatted data (see Section 8.3.2) is stored in a file named `filnam.pert where filnam` is the name of the .in-file. The command is set up as follows:

   ```
   command='LPREAD'
   ```

6. `LPReset`: reset the linear perturbation vector to zero. This command has no arguments. The command is set up as follows:

   ```
   command='LRESET'
   ```

7. `LSPot`: use the linear model generated in the `BUild` and `LPErturb` commands to compute the spot diagram at the last element specified in the `BUild` command. This element should be a focal plane (`EltType=3`). The output of this command is stored in `OutRaySpot` and (depending on the active plot option) may be plotted on the screen. This command has no arguments. The command is set up as follows:

   ```
   command='LSPOT'
   CARG(1)='YES' (to overwrite if a file-writing plot type is selected and a
   file of the same name exists)
   ```

8. `LOPd`: use the linear model generated in the `BUILD` and `LPERTURB` commands to compute the OPD map at the last element specified in the `BUILD` command. This element should be a reference surface (`EltType=4`). The output of this command is stored in `PertOPDMat` and (depending on the active plot option) may be plotted on the screen. This command has no arguments. The command is set up as follows:

   ```
   command='LOPD'
   CARG(1)='YES' (to overwrite if a file-writing plot type is selected and a
   file of the same name exists)
   ```

9. `LINtensity`: use the linear model generated in the `BUild` and `LPErturb` commands to compute and plot the intensity of the wavefront at the last element specified in the `BUild` command. This element should be a reference surface (`EltType=4`) followed

by a focal plane set up for a far-field diffraction. This command has no arguments. The command is set up as follows:

```
command='LINTENSITY'
CARG(1)='YES' (to overwrite if a file-writing plot type is selected and a file of the same name exists)
```

10. `LPIxilate`: use the linear model generated in the `BUild` and `LPErturb` commands to compute and plot the pixilated wavefront intensity at the last element specified in the `BUild` command. This element should be a reference surface (`EltType=3`) followed by a focal plane set up for a far-field diffraction. This command has two arguments: the number of pixels across one side of the array (assumed square) and the size of a pixel. The minimum size of the pixel is half the wavefront grid spacing. This is not automatically checked, so be careful! The `PLOcate` command does not apply to the `LPIxilate` command. The command is set up as follows:

```
command='LPIXILATE'
IARG(1)=number of pixels per side
RARG(1)=size of pixel
CARG(1)='YES' (to overwrite if a file-writing plot type is selected and a file of the same name exists)
```

REFERENCES

1. D. Redding and W. Breckenridge, “Modeling Optics for Dynamics and Control Analysis,” Journal of Guidance, Control and Dynamics, September, 1991.
2. J. W. Goodman, *Introduction to Fourier Optics*, McGraw-Hill, New York, 1968.
3. Sziklas and Siegman, “Mode Calculations in Unstable Resonators with Flowing Saturable Gain2: Fast Fourier Transform Method,” Applied Optics, v14, p. 1874-1889, 1975.
4. Oppenheim and Schaefer, *Digital Signal Processing.*
5. Brigham,
6. G. Lawrence, OASIS Manual, Version 5.0, August, 1985 with August 1986 updates. Published by the Rocketdyne Division of Rockwell International Corporation, Albuquerque Optical Support Office.
7. G. Lawrence, Optical Modeling preprint
8. D. Redding and W. Breckenridge, “Coordinate-Free Raytrace Equations for Aspheric and Deformed Optical Surfaces,” Memo DCR-91-38, July 13, 1991.
9. D. Redding, “Mathematical Description of Generalized Optical Element Surfaces,” Memo DCR-91-26, May 27, 1991.
10. P. C. Hughes, *Spacecraft Attitude Dynamics*,1986.
11. H. Akima, “A method of bivariate interpolation and smooth surface fitting for irrr-regularly distributed data points,” ACM Trans. on Mathematical Software, V4, No 2, June 1978.
12. H. Akima, “ALGORITH 526:Bivariate interpolation and smooth surface fitting for irrrregularly distributed data points,” ACM Trans. on Mathematical Software, V4, No 2, June 1978.
13. D. C. Redding, D, P. Dumont, and J. Yu, “Hubble Space Telescope Prescription Retrieval,” Optical Society of America, Space Optics Meeting, Paper MF4. Williamsburg VA. 11/18-19/91.
14. D. C. Redding and W. Breckenridge, “Optical Modeling for Dynamics and Control Analysis,” Journal of Guidance, Control, and Dynamics, Vol. 14, No. 5. Sept.-Oct. 1991.

APPENDIX A Examples and Demonstrations

This appendix contains prescriptions and macros that illustrate various aspects of MACOS usage. Electronic versions of these files have been provided with the MACOS distribution tapes. The interested user can execute these examples to gain experience using MACOS.

A.1 Cassegrain Telescope Examples

A.1.1 Cassegrain.jou

```
% This macro generates example figures
% for Section 2 of the MACOS User Manual

 old Cassegrain
 modify ngridpts=51 q
 spot 1 Tout                                                    % Fig 3a
 draw;;;;                                                       % Fig 3b

 spot 5 Tout                                                    % Fig 4a
 perturb 2 global 0,1e-4,0 0,0,0
 spot 5 Tout                                                    % Fig 4b
```

A.1.2 Cassegrain.in

```
% Cassegrain telescope example

ChfRayDir=  0d0  0d0  1d0
ChfRayPos=  0d0  0d0 -5d0
 zSource=   1.d+21
  IndRef=   1d0
  Extinc=   0d0
 Wavelen=   1.d-06
    Flux=   1d0
GridType=   Circular
Aperture=   4d0
Obscratn=   0d0
nGridpts=   15
   xGrid=  -1d0  0d0  0d0
   yGrid=   0d0 -1d0  0d0
    nElt=    5

    iElt=    1
 EltName= SecMirObs
 Element= Obscuring
 Surface= Conic
   KrElt=  -1.d+20
```

```
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0 -4.2d+00
  RptElt=   0d0  0d0 -4.2d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    3
 ObsType=   Circle
  ObsVec=   5.d-01  0d0  0d0
 ObsType=   Rectangle
  ObsVec=  -0.003125  0.003125 -3  3
 ObsType=   Rectangle
  ObsVec=  -3  3 -0.003125  0.003125
    xObs=   1d0  0d0  0d0
  ApType=   Circular
   ApVec=   2d0  0d0  0d0
    zElt=   1.d+20
PropType=   Geometric
 nECoord=   -6

    iElt=    2
 EltName= Primary
 Element= Reflector
 Surface= Conic
   KrElt=  -1.08d+01
   KcElt=  -1d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  0d0
  RptElt=   0d0  0d0  0d0
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   Circular
   ApVec=   2.1d0  0d0  0d0
    zElt=   5.4d+00
PropType=   Geometric
 nECoord=   -6

    iElt=    3
 EltName= Secondary
 Element= Reflector
 Surface= Conic
   KrElt=  -3.526787501D+00
   KcElt=  -2.670556022D+00
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0 -4.061145902D+00
  RptElt=   0d0  0d0 -5.4d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.338854098D+00
PropType=   Geometric
 nECoord=   -6

    iElt=    4
 EltName= Ref_surf
 Element= Reference
 Surface= Conic
   KrElt=  -5.560145902D+00
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -4.060145902D+00
  RptElt=   0d0  0d0 -4.060145902D+00
```

```
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   5.560145902D+00
PropType=   FarField
 nECoord=   -6

    iElt=    5
 EltName= Detector
 Element= FocalPlane
 Surface= Flat
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  1.5d+00
  RptElt=   0d0  0d0  1.5d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   0d0
PropType=   Geometric
 nECoord=   -6

nOutCord=    5
    Tout=   1d0  0d0  0d0  0d0  0d0  0d0  0d0
            0d0  1d0  0d0  0d0  0d0  0d0  0d0
            0d0  0d0  0d0  1d0  0d0  0d0  0d0
            0d0  0d0  0d0  0d0  1d0  0d0  0d0
            0d0  0d0  0d0  0d0  0d0  0d0  1d0
```

A.1.3 CassWithExitPupil.jou

```
% This macro generates more example figures
% for Section 2 of the MACOS User Manual

 old CassWithExitPupil
 modify ngridpts=128 q
 perturb 3 global 0,0,0 0,0,1e-4
 stop obj 0,0,0
 fex 5 yes
 opd 5 Tout                                                    % Fig 6a
 perturb 2 global 0,1e-4,0 0,0,0
 opd 5 Tout                                                    % Fig 6b

 old CassWithExitPupil
 modify ngridpts=64 q
 stop obj 0,0,0
 fex 5 yes
 intensity 6                                                   % Fig 10a
 perturb 2 global 0,1e-4,0 0,0,0
 fex 5 yes
 intensity 6                                                   % Fig 10b
```

A.1.4 CassWithExitPupil.in

```
% Cassegrain telescope example
```

```
% This version has an exit pupil

ChfRayDir=  0d0  0d0  1d0
ChfRayPos=  0d0  0d0 -5d0
 zSource=   1.d+21
  IndRef=   1d0
  Extinc=   0d0
 Wavelen=   1.d-06
    Flux=   1d0
GridType=   Circular
Aperture=   4d0
Obscratn=   0d0
nGridpts=   15
   xGrid=  -1d0  0d0  0d0
   yGrid=   0d0 -1d0  0d0
    nElt=    6

    iElt=    1
 EltName= SecMirObs
 Element= Obscuring
 Surface= Conic
   KrElt=  -1.d+20
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0 -4.2d+00
  RptElt=   0d0  0d0 -4.2d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    3
 ObsType=   Circle
  ObsVec=   5.d-01  0d0  0d0
 ObsType=   Rectangle
  ObsVec=  -0.003125  0.003125 -3  3
 ObsType=   Rectangle
  ObsVec=  -3  3 -0.003125  0.003125
    xObs=   1d0  0d0  0d0
  ApType=   Circular
   ApVec=   2d0  0d0  0d0
    zElt=   1.d+20
PropType=   Geometric
 nECoord=   -6

    iElt=    2
 EltName= Primary
 Element= Reflector
 Surface= Conic
   KrElt=  -1.08d+01
   KcElt=  -1d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  0d0
  RptElt=   0d0  0d0  0d0
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   Circular
   ApVec=   2.1d0  0d0  0d0
    zElt=   5.4d+00
PropType=   Geometric
 nECoord=   -6

    iElt=    3
 EltName= Secondary
 Element= Reflector
 Surface= Conic
```

```
   KrElt=  -3.526787501D+00
   KcElt=  -2.670556022D+00
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0 -4.061145902D+00
  RptElt=   0d0  0d0 -5.4d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.338854098D+00
PropType=   Geometric
 nECoord=   -6

    iElt=    4
 EltName= Return1
 Element= Return
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  1.5d+00
  RptElt=   0d0  0d0  1.5d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   0d0
PropType=   Geometric
 nECoord=   -6

    iElt=    5
 EltName= ExitPupil
 Element= Return
 Surface= Conic
   KrElt=  -5.560145902D+00
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -4.060145902D+00
  RptElt=   0d0  0d0 -4.060145902D+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   5.560145902D+00
PropType=   FarField
 nECoord=   -6

    iElt=    6
 EltName= Detector
 Element= FocalPlane
 Surface= Flat
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  1.5d+00
  RptElt=   0d0  0d0  1.5d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
```

```
  ApType=   None
    zElt=   0d0
PropType=   Geometric
 nECoord=   -6

nOutCord=    5
    Tout=   1d0  0d0  0d0  0d0  0d0  0d0  0d0
            0d0  1d0  0d0  0d0  0d0  0d0  0d0
            0d0  0d0  0d0  1d0  0d0  0d0  0d0
            0d0  0d0  0d0  0d0  1d0  0d0  0d0
            0d0  0d0  0d0  0d0  0d0  0d0  1d0
```

A.2 Diffractive Elements Examples

A.2.1 GratingExample.jou

```
% This macro generates  example figures
% for Section 4 of the MACOS User Manual

 old gratingExample

 mod OrderHOE(1)=0 q
 draw;;;;

 mod OrderHOE(1)=1 q
 draw;;;;                                                    % Fig ??

 mod OrderHOE(1)=2 q
 draw;;;;

 mod OrderHOE(1)=-1 q
 draw;;;;

 mod OrderHOE(1)=-2 q
 draw;;;;                                                    % Fig ??
```

A.2.2 GratingExample.in

```
ChfRayDir=  0d0  0d0  1d0
ChfRayPos=  0d0  0d0 -6d0
 zSource=   1d22
  IndRef=   1d0
  Extinc=   0d0
 Wavelen=   6.d-07
    Flux=   1d0
GridType=   Circular
Aperture=   1d0
Obscratn=   0d0
nGridpts=   65
   xGrid=   1d0  0d0  0d0
   yGrid=   0d0  1d0  0d0
    nElt=    2

    iElt=    1
 EltName= Grating
 Element= Grating
 Surface= Flat
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  0d0
  RptElt=   0d0  0d0  0d0
```

```
  IndRef=   1d0
  Extinc=   1d22
   h1HOE=   1d0  0d0  0d0
OrderHOE=   1d0
RuleWidth=   18.97366596d-7
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    2
 EltName= FP
 Element= FocalPlane
 Surface= Flat
   KrElt=   1d22
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -3d0
  RptElt=   0d0  0d0  0d0
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

nOutCord=    5
    Tout=   1d0  0d0  0d0  0d0  0d0  0d0  0d0
            0d0  1d0  0d0  0d0  0d0  0d0  0d0
            0d0  0d0  0d0  1d0  0d0  0d0  0d0
            0d0  0d0  0d0  0d0  1d0  0d0  0d0
            0d0  0d0  0d0  0d0  0d0  0d0  1d0
```

A.2.3 HOEExample.jou

```
% This macro generates example figures
% for Section 4 of the MACOS User Manual

 loa HOEExample
 mod ngridpts=15 q
 map
 ray 76
     2
     69
     0
 stop elt 1 0,0
 fex;;;
 int;;
 draw;;;;                                                    % Fig ??

 mod Element(1)=reflector q
 ray 76
     2
     69
     0
 fex;;;
 int;;
```

```
 draw;;;;                                                        % Fig ??
```

A.2.4 HOEExample.in

```
ChfRayDir=  0d0  0d0  1d0
ChfRayPos=  0d0  0d0 -4d0
 zSource=   1d22
  IndRef=   1d0
  Extinc=   0d0
 Wavelen=   6.d-07
    Flux=   1d0
GridType=   Circular
Aperture=   5.d-01
Obscratn=   0d0
nGridpts=   64
   xGrid=   1d0  0d0  0d0
   yGrid=   0d0  1d0  0d0
    nElt=    2

    iElt=    1
 EltName= HOEonMirror
 Element= HOE
 Surface= Conic
   KrElt=  -6d0
   KcElt=  -1d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  0d0
  RptElt=   0d0  0d0  0d0
  IndRef=   1d0
  Extinc=   1d22
   h1HOE=   0d0  0d0 -1d22
   h2HOE=   1d0  0d0 -3d0
OrderHOE=   1d0
 WaveHOE=   6.d-07
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    2
 EltName= FP
 Element= FocalPlane
 Surface= Flat
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -3d0
  RptElt=   0d0  0d0  0d0
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

nOutCord=    5
    Tout=   1d0  0d0  0d0  0d0  0d0  0d0  0d0
            0d0  1d0  0d0  0d0  0d0  0d0  0d0
            0d0  0d0  0d0  1d0  0d0  0d0  0d0
            0d0  0d0  0d0  0d0  1d0  0d0  0d0
            0d0  0d0  0d0  0d0  0d0  0d0  1d0
```

A.3 Non-Sequential Surface Examples

A.3.1 CornerCube.jou

```
old CornerCube

draw;;;xz

mod ChfRayPos(1)=0.762
    ChfRayPos(2)=0.762
    Aperture=0.125
    nGridpts=11
    q
draw;;;yz
```

A.3.2 CornerCube.in

```
ChfRayDir=  0d0  0d0 -1d0
ChfRayPos=  1.d-05  1.d-05  1.d+01
 zSource=   1.d22
  IndRef=   1d0
  Extinc=   0d0
 Wavelen=   1.059d-04
    Flux=   1d0
GridType=   Circular
Aperture=   2.54d+00
Obscratn=   0d0
nGridpts=   65
   xGrid=   0d0  1d0  0d0
   yGrid=  -1d0  0d0  0d0
    nElt=    5

    iElt=    1
 EltName= Ref Srf
 Element= Reference
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  3.5d+00
  RptElt=   0d0  0d0  0d0
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   0d0
PropType=   NFP1surf
 nECoord=    0

    iElt=    2
 EltName= CC1
 Element= NSReflector
 Surface= Conic
   KrElt=   1d22
   KcElt=   0d0
  psiElt=   8.164994677D-01  0d0  5.773461867D-01
  VptElt=   0d0  0d0  0d0
  RptElt=   0d0  0d0  0d0
```

```
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    3
 EltName= CC2
 Element= NSReflector
 Surface= Conic
   KrElt=   1d22
   KcElt=   0d0
  psiElt=  -4.082497338D-01  7.071092812D-01  5.773461867D-01
  VptElt=   0d0  0d0  0d0
  RptElt=   0d0  0d0  0d0
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    4
 EltName= CC3
 Element= NSReflector
 Surface= Conic
   KrElt=   1d22
   KcElt=   0d0
  psiElt=  -4.082497338D-01 -7.071092812D-01  5.773461867D-01
  VptElt=   0d0  0d0  0d0
  RptElt=   0d0  0d0  0d0
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    5
 EltName= FP
 Element= FocalPlane
 Surface= Flat
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  3.5d+00
  RptElt=   0d0  0d0  0d0
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.d+03
PropType=   Geometric
 nECoord=    0

nOutCord=    5
    Tout=   1d0  0d0  0d0  0d0  0d0  0d0  0d0
            0d0  1d0  0d0  0d0  0d0  0d0  0d0
            0d0  0d0  0d0  1d0  0d0  0d0  0d0
            0d0  0d0  0d0  0d0  1d0  0d0  0d0
```

```
            0d0  0d0  0d0  0d0  0d0  0d0  1d0
```

A.3.3 Luneberg.jou

```
%  Luneberg.jou: a MACOS example macro. This macro
%  demonstrates use of non-sequential refracting
%  elements, polarization, and the gain command.

 old Luneberg
 draw 0 16 XZ
 beam cos 8.7500000000D-01 1.5
 mod nGridPts=128 q
 int 16
 mod nGridPts=64 q
 gain 17
 col
 gain 17 129
 pol 1,0 0,0
 mod nGridPts=128 q
 int 16
 mod nGridPts=64 q
 gain 17
 col
 gain 17 129
```

A.3.4 Luneberg.in

```
%  Luneberg.in: a MACOS example .in-file. This
%  macro demonstrates use of non-sequential
%  refractors, polarization, and the gain command.

ChfRayDir=  0d0  0d0  1d0
ChfRayPos=  0d0  0d0 -2.121332605D-01
 zSource=  -1.d-02
  IndRef=   1d0
  Extinc=   0d0
 Wavelen=   2.15d-02
    Flux=   1d0
GridType=   Circular
Aperture=   1.75d+00
Obscratn=   0d0
nGridpts=   33
   xGrid=   1d0  0d0  0d0
   yGrid=   0d0  1d0  0d0
    nElt=   17

    iElt=    1
 EltName= Shell1
 Element= NSRefractor
 Surface= Conic
   KrElt=  -2.0353d-01
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -2.0353d-01
  RptElt=   0d0  0d0  0d0
  IndRef=   1.056692495D+00
  Extinc=   0d0
    nObs=    0
```

```
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    2
 EltName= Shell2
 Element= NSRefractor
 Surface= Conic
   KrElt=  -1.887578424D-01
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -1.887578424D-01
  RptElt=   0d0  0d0  0d0
  IndRef=   1.084805420D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    3
 EltName= Shell3
 Element= NSRefractor
 Surface= Conic
   KrElt=  -1.872644104D-01
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -1.872644104D-01
  RptElt=   0d0  0d0  0d0
  IndRef=   1.147803142D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    4
 EltName= Shell4
 Element= NSRefractor
 Surface= Conic
   KrElt=  -1.771655689D-01
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -1.771655689D-01
  RptElt=   0d0  0d0  0d0
  IndRef=   1.182543332D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    5
 EltName= Shell5
 Element= NSRefractor
 Surface= Conic
   KrElt=  -1.529604025D-01
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -1.529604025D-01
```

```
  RptElt=   0d0  0d0  0d0
  IndRef=   1.235415610D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    6
 EltName= Shell6
 Element= NSRefractor
 Surface= Conic
   KrElt=  -1.387929631D-01
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -1.387929631D-01
  RptElt=   0d0  0d0  0d0
  IndRef=   1.265593240D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    7
 EltName= Shell7
 Element= NSRefractor
 Surface= Conic
   KrElt=  -1.297999186D-01
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -1.297999186D-01
  RptElt=   0d0  0d0  0d0
  IndRef=   1.283513736D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    8
 EltName= Shell8
 Element= NSRefractor
 Surface= Conic
   KrElt=  -1.087287483D-01
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -1.087287483D-01
  RptElt=   0d0  0d0  0d0
  IndRef=   1.312911218D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6
```

```
    iElt=    9
 EltName= Shell9
 Element= NSRefractor
 Surface= Conic
   KrElt=  -9.780754704D-02
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -9.780754704D-02
  RptElt=   0d0  0d0  0d0
  IndRef=   1.340947910D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   10
 EltName= Shell10
 Element= NSRefractor
 Surface= Conic
   KrElt=  -8.777974770D-02
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -8.777974770D-02
  RptElt=   0d0  0d0  0d0
  IndRef=   1.338212289D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   11
 EltName= Shell11
 Element= NSRefractor
 Surface= Conic
   KrElt=  -7.943573990D-02
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -7.943573990D-02
  RptElt=   0d0  0d0  0d0
  IndRef=   1.371777076D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   12
 EltName= Shell12
 Element= NSRefractor
 Surface= Conic
   KrElt=  -6.183854602D-02
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -6.183854602D-02
  RptElt=   0d0  0d0  0d0
  IndRef=   1.386006296D+00
  Extinc=   0d0
    nObs=    0
```

```
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   13
 EltName= Shell13
 Element= NSRefractor
 Surface= Conic
   KrElt=  -4.663931773D-02
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -4.663931773D-02
  RptElt=   0d0  0d0  0d0
  IndRef=   1.402911090D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   14
 EltName= Shell14
 Element= NSRefractor
 Surface= Conic
   KrElt=  -2.956340242D-02
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -2.956340242D-02
  RptElt=   0d0  0d0  0d0
  IndRef=   1.412d+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   15
 EltName= Aperture
 Element= Return
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  2.221359167D-01
  RptElt=   0d0  0d0  0d0
  IndRef=   1.0488d+00
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.d+01
PropType=   Geometric
 nECoord=   -6

    iElt=   16
 EltName= ExitPupil
 Element= Return
```

```
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  0d0
  RptElt=   0d0  0d0  0d0
  IndRef=   1.0488d+00
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.d+01
PropType=   FarField
 nECoord=   -6

    iElt=   17
 EltName= FP
 Element= FocalPlane
 Surface= Flat
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  1d3
  RptElt=   0d0  0d0  0d0
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   0d0
PropType=   Geometric
 nECoord=   -6

nOutCord=    5
    Tout=   1d0  0d0  0d0  0d0  0d0  0d0  0d0
            0d0  1d0  0d0  0d0  0d0  0d0  0d0
            0d0  0d0  0d0  1d0  0d0  0d0  0d0
            0d0  0d0  0d0  0d0  1d0  0d0  0d0
            0d0  0d0  0d0  0d0  0d0  0d0  1d0
```

A.4 Adaptive Optics Example

A.4.1 AOExample.jou

```
%  MACOS Adaptive Optics Example Macro

%   First show system response without atmosphere:

 old AOexample;                         % Load prescription
 draw 0 26 xz                           % Draw full system
 draw 6 26 xy                           % Draw AO optics only
 spo 1 b y                              % Pupil spot diagram
 opd 16 y                               % OPD in exit pupil shows deformable
                                        %   mirror test pattern
 spo 26 Tout y                          % Hartman sensor spots on detector

%   Next show system response with atmosphere but no
%   deformable mirror:

 modify Surface(7)=Flat                 % Flatten deformable mirror
        q
 atmos 1 20                             % Add atmospheric phase disturbance
   2.854331e-5 0 atm 12.5               %   at entrance pupil
```

```
 opd 16 y                                % OPD in exit pupil
 spo 26 Tout y                           % Hartman sensor spots on detector
```

A.4.2 AOExample.in

```
%  MACOS Adaptive Optics Example Prescription

ChfRayDir=  0d0  0d0  1d0
ChfRayPos=  0d0  0d0 -2d+02
 zSource=   1d22
  IndRef=   1d0
  Extinc=   0d0
 Wavelen=   2.854330709D-05
    Flux=   1d0
GridType=   Circular
Aperture=   1.991873225632d02
Obscratn=   40.5d0
nGridpts=   128
   xGrid=   0d0 -1d0  0d0
   yGrid=  -1d0  0d0  0d0
    nElt=   26

    iElt=    1
 EltName= Pupil
 Element= Reference
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0 -6d0
  RptElt=   0d0  0d0  0d0
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    2
 EltName= PM
 Element= Reflector
 Surface= Conic
   KrElt=  -1.3357d+03
   KcElt=  -1d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  0d0
  RptElt=   0d0  0d0  0d0
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   6.6785d+02
PropType=   Geometric
 nECoord=   -6

    iElt=    3
 EltName= SM
```

```
 Element= Reflector
 Surface= Conic
   KrElt=  -3.184999999D+02
   KcElt=  -2.352756999D+00
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0 -5.423064572D+02
  RptElt=   0d0  0d0 -5.423064572D+02
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1.256970591D+02
PropType=   Geometric
 nECoord=   -6

    iElt=    4
 EltName= FM1
 Element= Reflector
 Surface= Flat
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   7.071067812D-01  0d0  7.071067812D-01
  VptElt=   0d0  0d0  7.5375d+01
  RptElt=   0d0  0d0  7.5375d+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    5
 EltName= OAP1
 Element= Reflector
 Surface= Conic
   KrElt=  -1.364d+02
   KcElt=  -1d0
  psiElt=   9.668919598D-01 -2.551860851D-01  0d0
  VptElt=  -4.140665047D+01  1.740405528D+01  7.5375d+01
  RptElt=  -4.140665047D+01  1.740405528D+01  7.5375d+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   6.82d+01
PropType=   Geometric
 nECoord=   -6

    iElt=    6
 EltName= FSM
 Element= Reflector
 Surface= Flat
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   9.999929341D-01  3.759223637D-03  0d0
  VptElt=   1.318856255D+01 -1.530759269D+01  7.5375d+01
  RptElt=   1.318856255D+01 -1.530759269D+01  7.5375d+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
```

```
 nECoord=   -6

    iElt=    7
 EltName= DM
 Element= Reflector
 Surface= UserDefined
   KrElt=  -1d22
   KcElt=   0d0
UDSrfType=    1                                                          % 1 or 5
UDSrfFile= AOmirror.test
    pMon=  -2.249426003D+00 -1.951098758D+01  7.5375d+01
    xMon=  -4.697040833D-01  8.828239202D-01  0d0
    yMon=   0d0  0d0  1d0
    zMon=   8.828239202D-01  4.697040833D-01  0d0
    lMon=   0d0
  psiElt=   8.828239202D-01  4.697040833D-01  0d0
  VptElt=  -2.250697813D+00 -1.950691308D+01  7.5375d+01
  RptElt=  -2.250697813D+00 -1.950691308D+01  7.5375d+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    8
 EltName= FM2
 Element= Reflector
 Surface= Flat
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   9.037029675D-02  9.959082335D-01  0d0
  VptElt=   2.331918268D+01  2.576404533D+00  7.5375d+01
  RptElt=   2.331918268D+01  2.576404533D+00  7.5375d+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    9
 EltName= OAP2
 Element= Reflector
 Surface= Conic
   KrElt=  -1.364d+02
   KcElt=  -1d0
  psiElt=  -6.268090394D-01  7.791729129D-01  0d0
  VptElt=   2.538458250D+01 -2.822361844D+01  7.5375d+01
  RptElt=   2.538458250D+01 -2.822361844D+01  7.5375d+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   6.82d+01
PropType=   Geometric
 nECoord=   -6
```

```
    iElt=   10
 EltName= SSM1
 Element= Reflector
 Surface= Flat
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=  -9.937679192D-01  1.114689322D-01  0d0
  VptElt=   5.421565944D+00  8.117154092D+00  7.5375d+01
  RptElt=   5.421565944D+00  8.117154092D+00  7.5375d+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   11
 EltName= SSM2
 Element= Reflector
 Surface= Flat
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=  -9.970527522D-01  7.671902813D-02  0d0
  VptElt=   1.733428143D+01  1.332169152D+01  7.5375d+01
  RptElt=   1.733428143D+01  1.332169152D+01  7.5375d+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   12
 EltName= FS
 Element= Reflector
 Surface= Flat
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=  -9.593139745D-01  2.823414568D-01  0d0
  VptElt=   4.387255193D+00  2.153812421D+01  7.5375d+01
  RptElt=   4.387255193D+00  2.153812421D+01  7.5375d+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   13
 EltName= WFSRL1A
 Element= Refractor
 Surface= Conic
   KrElt=  -1.550872636D+00
   KcElt=   0d0
  psiElt=   9.999756307D-01 -6.981260298D-03  0d0
  VptElt=   8.496015971D+00  2.150943918D+01  7.5375d+01
  RptElt=   8.496015971D+00  2.150943918D+01  7.5375d+01
  IndRef=   1.512425970D+00
  Extinc=   0d0
    nObs=    0
```

```
  ApType=   None
    zElt=   5.288796394D+00
PropType=   Geometric
 nECoord=   -6

    iElt=   14
 EltName= WFSRL1B
 Element= Refractor
 Surface= Conic
   KrElt=  -4.419109771D+00
   KcElt=   0d0
  psiElt=  -9.999756307D-01  6.981260298D-03  0d0
  VptElt=   8.996003786D+00  2.150594855D+01  7.5375d+01
  RptElt=   8.996003786D+00  2.150594855D+01  7.5375d+01
  IndRef=   1.789618121D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   3.411018511D+00
PropType=   Geometric
 nECoord=   -6

    iElt=   15
 EltName= WFSRL1C
 Element= Refractor
 Surface= Conic
   KrElt=  -2.771554824D+01
   KcElt=   0d0
  psiElt=   9.999756307D-01 -6.981260298D-03  0d0
  VptElt=   9.495991602D+00  2.150245792D+01  7.5375d+01
  RptElt=   9.495991602D+00  2.150245792D+01  7.5375d+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   2.198859945D+01
PropType=   Geometric
 nECoord=   -6

    iElt=   16
 EltName= LensArrayReference
 Element= Reference
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   9.999756303D-01 -6.981321726D-03  3.414481680D-15
  VptElt=   1.272900024D+01  2.147992068D+01  7.537500000D+01
  RptElt=   1.272900024D+01  2.147992068D+01  7.537500000D+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   17
 EltName= LensArray1A
 Element= LensArray
```

```
 Surface= Conic
   KrElt=  -0.47244094488189d+00
   KcElt=   0d0
  psiElt=   9.999756303D-01 -6.981321725D-03  0d0
  VptElt=   12.7291897290d0  21.4720113661d0  75.3671259843d0
  RptElt=   1.272924470D+01  2.147988519D+01  7.5375d+01
LensArrayType=  2
LensArrayWidth= 0.01574803149606
    pMon=  1.272924470D+01  2.147988519D+01  7.5375d+01
    xMon=  0d0  0d0 -1d0
    yMon=  -6.981321725D-03 -9.999756303D-01  0d0
    zMon=  -9.999756303D-01  6.981321725D-03  0d0
  IndRef=   1.512425970D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   5.17d+00
PropType=   Geometric
 nECoord=   -6

    iElt=   18
 EltName= LensArray1B
 Element= Refractor
 Surface= Flat
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=  -9.999756307D-01  6.981260298D-03  0d0
  VptElt=   1.297923860D+01  2.147813988D+01  7.5375d+01
  RptElt=   1.297923860D+01  2.147813988D+01  7.5375d+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   19
 EltName= FieldLensA
 Element= Refractor
 Surface= Conic
   KrElt=  -2.122047244D+00
   KcElt=   0d0
  psiElt=   9.999756307D-01 -6.981260298D-03  0d0
  VptElt=   1.358583579D+01  2.147390496D+01  7.5375d+01
  RptElt=   1.358583579D+01  2.147390496D+01  7.5375d+01
  IndRef=   1.677871170D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1.359427521D+00
PropType=   Geometric
 nECoord=   -6

    iElt=   20
 EltName= FieldLensB
 Element= Refractor
 Surface= Conic
   KrElt=  -3.224409449D+00
   KcElt=   0d0
  psiElt=   9.999756307D-01 -6.981260298D-03  0d0
  VptElt=   1.374331196D+01  2.147280555D+01  7.5375d+01
  RptElt=   1.374331196D+01  2.147280555D+01  7.5375d+01
  IndRef=   1d0
```

```
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1.359427521D+00
PropType=   Geometric
 nECoord=   -6

    iElt=   21
 EltName= WFSLAR1A
 Element= Refractor
 Surface= Conic
   KrElt=  -8.055118110D-01
   KcElt=   0d0
  psiElt=   9.999756307D-01 -6.981260298D-03  0d0
  VptElt=   1.880962705D+01  2.143743542D+01  7.5375d+01
  RptElt=   1.880962705D+01  2.143743542D+01  7.5375d+01
  IndRef=   1.611736488D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1.359427521D+00
PropType=   Geometric
 nECoord=   -6

    iElt=   22
 EltName= WFSLAR1B
 Element= Refractor
 Surface= Conic
   KrElt=  -5.090551181D-01
   KcElt=   0d0
  psiElt=  -9.999756307D-01  6.981260298D-03  0d0
  VptElt=   1.896631614D+01  2.143634151D+01  7.5375d+01
  RptElt=   1.896631614D+01  2.143634151D+01  7.5375d+01
  IndRef=   1.677871170D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   6.038543863D-01
PropType=   Geometric
 nECoord=   -6

    iElt=   23
 EltName= WFSLAR1C
 Element= Refractor
 Surface= Conic
   KrElt=  -2.980708661D+00
   KcElt=   0d0
  psiElt=  -9.999756307D-01  6.981260298D-03  0d0
  VptElt=   1.901946445D+01  2.143597046D+01  7.5375d+01
  RptElt=   1.901946445D+01  2.143597046D+01  7.5375d+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   2.027987141D+00
PropType=   Geometric
 nECoord=   -6

    iElt=   24
```

```
 EltName= CCDWinA
 Element= Refractor
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=  -9.999756307D-01  6.981260298D-03  0d0
  VptElt=   2.027861775D+01  2.142717977D+01  7.5375d+01
  RptElt=   2.027861775D+01  2.142717977D+01  7.5375d+01
  IndRef=   1.512425970D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   25
 EltName= CCDWinB
 Element= Refractor
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=  -9.999756307D-01  6.981260298D-03  0d0
  VptElt=   2.040361470D+01  2.142630711D+01  7.5375d+01
  RptElt=   2.040361470D+01  2.142630711D+01  7.5375d+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   26
 EltName= CCD
 Element= FocalPlane
 Surface= Flat
   KrElt=  -1.000000000D+22
   KcElt=   0.000000000D+00
  psiElt=  -9.999764894D-01  6.857153362D-03  0d0
  VptElt=   2.049655742D+01  2.142565820D+01  7.537500000D+01
  RptElt=   2.049655742D+01  2.142565820D+01  7.537500000D+01
  IndRef=   1.000000000D+00
  Extinc=   0.000000000D+00
    nObs=    0
  ApType=   None
    zElt=   1.000000000D+22
PropType=   Geometric
 nECoord=   -6

nOutCord=    5
    Tout=  -6.981260298D-03 -9.999756307D-01  0d0  0d0  0d0  0d0  0d0
            0d0  0d0 -1d0  0d0  0d0  0d0  0d0  0d0
            0d0  0d0  0d0 -6.981260298D-03 -9.999756307D-01  0d0  0d0
            0d0  0d0  0d0  0d0  0d0 -1d0  0d0
            0d0  0d0  0d0  0d0  0d0  0d0  1d0
```

A.4.3 AOmirror.test

```
  5.5000000e+01   4e-05
  7.6000000e+01  -4e-05
  7.7000000e+01  -4e-05
  9.5000000e+01  -4e-05
  9.6000000e+01  -4e-05
```

```
1.0200000e+02   3e-05
1.0300000e+02   3e-05
1.0400000e+02   3e-05
1.0500000e+02   3e-05
1.0600000e+02   3e-05
1.0700000e+02   3e-05
1.2300000e+02   4e-05
1.2400000e+02   4e-05
1.2500000e+02   4e-05
1.2600000e+02   4e-05
1.2700000e+02   4e-05
1.2800000e+02   4e-05
2.0700000e+02   5e-06
2.2800000e+02   5e-06
2.0800000e+02   1e-05
2.2900000e+02   1e-05
2.4900000e+02   1e-05
2.6800000e+02   1e-05
2.0900000e+02   1.5e-05
2.3000000e+02   1.5e-05
2.5000000e+02   1.5e-05
2.6900000e+02   1.5e-05
2.8700000e+02   1.5e-05
2.1000000e+02   2e-05
2.3100000e+02   2e-05
2.5100000e+02   2e-05
2.7000000e+02   2e-05
2.8800000e+02   2e-05
3.0400000e+02   2e-05
2.1100000e+02   2.5e-05
2.3200000e+02   2.5e-05
2.5200000e+02   2.5e-05
2.7100000e+02   2.5e-05
2.8900000e+02   2.5e-05
3.0500000e+02   2.5e-05
3.1900000e+02   2.5e-05
2.1200000e+02   3e-05
2.3300000e+02   3e-05
2.5300000e+02   3e-05
2.7200000e+02   3e-05
2.9000000e+02   3e-05
3.0600000e+02   3e-05
3.2000000e+02   3e-05
3.3200000e+02   3e-05
2.1300000e+02   3.5e-05
2.3400000e+02   3.5e-05
2.5400000e+02   3.5e-05
2.7300000e+02   3.5e-05
2.9100000e+02   3.5e-05
3.0700000e+02   3.5e-05
3.2100000e+02   3.5e-05
3.3300000e+02   3.5e-05
2.1400000e+02   4e-05
2.3500000e+02   4e-05
2.5500000e+02   4e-05
2.7400000e+02   4e-05
2.9200000e+02   4e-05
3.0800000e+02   4e-05
3.2200000e+02   4e-05
```

```
   3.3400000e+02   4e-05
   3.4300000e+02   4e-05
   2.4200000e+02   4e-05
   2.4300000e+02   4e-05
   2.4400000e+02   4e-05
   2.4500000e+02   4e-05
   2.2000000e+02   4e-05
   2.2100000e+02   4e-05
   2.2200000e+02   4e-05
   2.2300000e+02   4e-05
   2.2400000e+02   4e-05
   1.9900000e+02   4e-05
   2.0000000e+02   4e-05
   2.0100000e+02   4e-05
   2.0200000e+02   4e-05
   2.2000000e+02   4e-05
   1.7800000e+02   4e-05
   1.7900000e+02   4e-05
   1.8000000e+02   4e-05
   1.8100000e+02   4e-05
   1.8200000e+02   4e-05
   1.7500000e+02   4e-05
```

A.5 Segmented Telescope Example

A.5.1 SegDemo.jou

```
%  SegmentDemo.jou -- a MACOS demonstration macro that illustrates the gener-
ation
%   of a linear model of a segmented mirror system. The linear model is used
to
%   generate an image and OPD numbers. These are compared with the same num-
bers
%   computed using the full nonlinear model. They agree very closely.

%  Run this macro with macos256

old SegmentDemo;                                        % Load prescription file
                                                                              %
draw 0 10 xz;                                             % Sketch optics layout
                                                                              %
map 0                        % Displays the ray mapping in the entrance pupil; also
                                                    %  shows the segment mapping
                                                                              %
intensity 10                    % Generate simulated image at the focal plane -- no
                                                                   %  aberrations
                                                                              %
log 10                      % Display simulated unaberrated image with log10 stretch
                                                                              %
                                                                              %
perturb 3                   % Perturb the 3rd element using the full nonlinear model
   NO                                                    %  Use global coordinates
   0,1e-4,0                      %  Rotational perturbation: rotate 100 urad about y
   0,0,0                                         %  No translational perturbation
                                                                              %
opd 9                                  % Display wavefront error using linear model
                                                                              %
log 10                           % Display image (log10 stretch) using full nonlinear
                                                                    %  model
                                                                              %
old SegmentDemo;                                       % Reload prescription file
                                                                              %
```

```
build 9                                                  % Compute linear optical model
                                                                                        %
lperturb 3                          % Perturb the 3rd element using the linear model
   NO                                                    %  Use global coordinates
   0,1e-4,0                      %  Rotational perturbation: rotate 100 urad about y
   0,0,0                                          %  No translational perturbation
                                                                                        %
lopd                                      % Display wavefront error using linear model
                                                                                        %
llog                                   % Display image (log10 stretch) using linear model
```

A.5.2 SegDemo.in

```
%  SegmentDemo.in -- prescription file for SegmentDemo.jou, a MACOS demon-
stration
%   macro.

ChfRayDir=  0d0  0d0 -1d0
ChfRayPos=  0d0  0d0  1.5d+00
 zSource=   1.d+19
  IndRef=   1d0
  Extinc=   0d0
 Wavelen=   1.d-04
    Flux=   1d0
GridType=   Pie
Aperture=   3.65d+00
Obscratn=   3.d-01
nGridpts=   25
   xGrid=   1d0  0d0  0d0
   yGrid=   0d0  1d0  0d0
    nElt=   10
    nSeg=    7
   width=   1.2d+00
     gap=   0d0
SegXgrid=   1d0    0d0    0d0
SegCoord=     0         0         0
              1        -1        -2
              2         1        -1
              1         2         1
             -1         1         2
             -2        -1         1
             -1        -2        -1

    iElt=    1
 EltName= S1
 Element= Segment
 Surface= Conic
   KrElt=  -2.92d+00
   KcElt=  -1d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  0d0
  RptElt=   0d0  0d0  0d0
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.46d+00
PropType=   Geometric
```

```
 nECoord=   -6

    iElt=    2
 EltName= S2
 Element= Segment
 Surface= Conic
   KrElt=  -2.92d+00
   KcElt=  -1d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  0d0
  RptElt=   6.124997330D-01 -1.060881274D+00  2.569563356D-01
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.46d+00
PropType=   Geometric
 nECoord=    6
    TElt=   4.610697370D-01  8.660255296D-01 -1.934282287D-01  0d0  0d0  0d0
           -7.985966745D-01  4.999997821D-01  3.350277145D-01  0d0  0d0  0d0
            3.868566261D-01 -1.885624059D-18  9.221398760D-01  0d0  0d0  0d0
            0d0  0d0  0d0  4.610697370D-01  8.660255296D-01 -1.934282287D-01
            0d0  0d0  0d0 -7.985966745D-01  4.999997821D-01  3.350277145D-01
            0d0  0d0  0d0  3.868566261D-01 -1.885624059D-18  9.221398760D-01

    iElt=    3
 EltName= S3
 Element= Segment
 Surface= Conic
   KrElt=  -2.92d+00
   KcElt=  -1d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  0d0
  RptElt=   1.225d+00 -3.699397078D-07  2.569563356D-01
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.46d+00
PropType=   Geometric
 nECoord=    6
    TElt=   9.221398760D-01  3.019915982D-07 -3.868566261D-01  0d0  0d0  0d0
           -2.784784949D-07  1d0  1.168274508D-07  0d0  0d0  0d0
            3.868566261D-01  3.238991466D-24  9.221398760D-01  0d0  0d0  0d0
            0d0  0d0  0d0  9.221398760D-01  3.019915982D-07 -3.868566261D-01
            0d0  0d0  0d0 -2.784784949D-07  1d0  1.168274508D-07
            0d0  0d0  0d0  3.868566261D-01  3.238991466D-24  9.221398760D-01

    iElt=    4
 EltName= S4
 Element= Segment
 Surface= Conic
   KrElt=  -2.92d+00
   KcElt=  -1d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  0d0
  RptElt=   6.125003738D-01  1.060880904D+00  2.569563356D-01
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.46d+00
PropType=   Geometric
 nECoord=    6
```

```
    TElt=   4.610702194D-01 -8.660252276D-01 -1.934284311D-01  0d0  0d0  0d0
            7.985963960D-01  5.000003051D-01 -3.350275976D-01  0d0  0d0  0d0
            3.868566261D-01  2.270467800D-17  9.221398760D-01  0d0  0d0  0d0
            0d0  0d0  0d0  4.610702194D-01 -8.660252276D-01 -1.934284311D-01
            0d0  0d0  0d0  7.985963960D-01  5.000003051D-01 -3.350275976D-01
            0d0  0d0  0d0  3.868566261D-01  2.270467800D-17  9.221398760D-01

    iElt=    5
 EltName= S5
 Element= Segment
 Surface= Conic
   KrElt=  -2.92d+00
   KcElt=  -1d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  0d0
  RptElt=  -6.124995728D-01  1.060881366D+00  2.569563356D-01
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.46d+00
PropType=   Geometric
 nECoord=    6
    TElt=  -4.610696164D-01 -8.660256051D-01  1.934281781D-01  0d0  0d0  0d0
            7.985967441D-01 -4.999996513D-01 -3.350277437D-01  0d0  0d0  0d0
            3.868566261D-01 -2.095137844D-19  9.221398760D-01  0d0  0d0  0d0
            0d0  0d0  0d0 -4.610696164D-01 -8.660256051D-01  1.934281781D-01
            0d0  0d0  0d0  7.985967441D-01 -4.999996513D-01 -3.350277437D-01
            0d0  0d0  0d0  3.868566261D-01 -2.095137844D-19  9.221398760D-01

    iElt=    6
 EltName= S6
 Element= Segment
 Surface= Conic
   KrElt=  -2.92d+00
   KcElt=  -1d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  0d0
  RptElt=  -1.225d+00  5.549095617D-07  2.569563356D-01
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.46d+00
PropType=   Geometric
 nECoord=    6
    TElt=  -9.221398760D-01 -4.529873973D-07  3.868566261D-01  0d0  0d0  0d0
            4.177177424D-07 -1d0 -1.752411762D-07  0d0  0d0  0d0
            3.868566261D-01 -1.071601397D-23  9.221398760D-01  0d0  0d0  0d0
            0d0  0d0  0d0 -9.221398760D-01 -4.529873973D-07  3.868566261D-01
            0d0  0d0  0d0  4.177177424D-07 -1d0 -1.752411762D-07
            0d0  0d0  0d0  3.868566261D-01 -1.071601397D-23  9.221398760D-01

    iElt=    7
 EltName= S7
 Element= Segment
 Surface= Conic
   KrElt=  -2.92d+00
   KcElt=  -1d0
```

```
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  0d0
  RptElt=  -6.125005340D-01 -1.060880811D+00  2.569563356D-01
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.46d+00
PropType=   Geometric
 nECoord=    6
    TElt=  -4.610703399D-01  8.660251521D-01  1.934284817D-01  0d0  0d0  0d0
           -7.985963264D-01 -5.000004359D-01  3.350275684D-01  0d0  0d0  0d0
            3.868566261D-01 -1.153428518D-17  9.221398760D-01  0d0  0d0  0d0
            0d0  0d0  0d0 -4.610703399D-01  8.660251521D-01  1.934284817D-01
            0d0  0d0  0d0 -7.985963264D-01 -5.000004359D-01  3.350275684D-01
            0d0  0d0  0d0  3.868566261D-01 -1.153428518D-17  9.221398760D-01

    iElt=    8
 EltName= SecMir
 Element= Reflector
 Surface= Conic
   KrElt=  -2.229620353D-01
   KcElt=  -1.174529970D+00
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  1.353d+00
  RptElt=   0d0  0d0  1.353d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.07d-01
PropType=   Geometric
 nECoord=   -6

    iElt=    9
 EltName= RefSurf
 Element= Reference
 Surface= Conic
   KrElt=  -2.6d+00
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  1.291d+00
  RptElt=   0d0  0d0  1.291d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   2.6d+00
PropType=   FarField
 nECoord=   -6

    iElt=   10
 EltName= FclPlane
 Element= FocalPlane
 Surface= Flat
   KrElt=   0d0
   KcElt=  -1.d+38
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0 -1.309d+00
  RptElt=   0d0  0d0 -1.309d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
```

```
    zElt=   1.d+19
PropType=   Geometric
 nECoord=   -6

nOutCord=    5
    Tout=   1d0  0d0  0d0  0d0  0d0  0d0  0d0
            0d0  1d0  0d0  0d0  0d0  0d0  0d0
            0d0  0d0  0d0  1d0  0d0  0d0  0d0
            0d0  0d0  0d0  0d0  1d0  0d0  0d0
            0d0  0d0  0d0  0d0  0d0  0d0  1d0
```

A.6 Near-Field Propagation Example

A.6.1 coroExample.jou

```
%  MACOS demonstration macro

%  This system is a simple coronograph. The demo
%  generates a bright on-axis (occulted) image, then
%  adds a dim, nearby but slightly off-axis image.

%  Run this with macos512!

 old coroExample;
 draw 0 16 XZ                    % sketch system
 stretch sqrt
 mod ngridpts=128
     nObs(6)=0                   % undo occulting mask
     q
 int 1                           % pupil function
 int 6                           % unocculted PSF
 pix 6 64 ;                      % blowup of PSF
 mod ngridpts=128
     nObs(6)=1                   % redo occulting mask
     q
 pix 6 64 ;                      % blowup of occulted PSF
 stretch linear
 int 10                          % pupil
 int 11                          % pupil at Lyot stop
 stretch sqrt
 int 16                          % on-axis PSF at detector

 win Tout 1.392542d-6 0,0 0,0    % set window for composite image
 compose 16 64 1.392542E-06      % build composite image
 add yes                         % blowup of on-axis PSF

 stop elt 11 0,0                 % set system stop at Lyot stop
 ffp 6 2d-5,2d-5 yes             % move new image off occulting mask
 ors 5 yes                       % reset return surface
 ors 7 yes                       % reset return surface
 fex 15 yes                      % reset return surface
 mod flux=0.01                   % 100x dimmer object
     q
 stretch log10
 int 6                           % new dim image at occulting mask
 stretch sqrt
```

```
 add yes                          % add dim image to composite image
```

A.6.2 coroExample.in

```
ChfRayDir=  0d0  0d0  1d0
ChfRayPos=  0d0  0d0 -2.d+01
 zSource=   1.d+30
  IndRef=   1d0
  Extinc=   0d0
 Wavelen=   1.d-06
    Flux=   1d0
GridType=   Circular
Aperture=   4d0
Obscratn=   0d0
nGridpts=   512
   xGrid=   1d0  0d0  0d0
   yGrid=   0d0  1d0  0d0
    nElt=   16

    iElt=    1
 EltName= SMObs_1
 Element= Obscuring
 Surface= Conic
   KrElt=  -1.d+30
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0 -5.4d+00
  RptElt=   0d0  0d0 -5.4d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    3
 ObsType=   Circle
  ObsVec=   5d-01  0d0  0d0
 ObsType=   Rectangle
  ObsVec=  -5d-02  5d-02 -2d0  2d0
 ObsType=   Rectangle
  ObsVec=  -2d0  2d0 -5d-02  5d-02
    xObs=   1d0  0d0  0d0
  ApType=   Circular
   ApVec=   2d0  0d0  0d0
    zElt=   1.d+30
PropType=   Geometric
 nECoord=   -6

    iElt=    2
 EltName= PM_1
 Element= Reflector
 Surface= Conic
   KrElt=  -1.08d+01
   KcElt=  -1d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  0d0
  RptElt=   0d0  0d0  0d0
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   5.4d+00
PropType=   Geometric
 nECoord=   -6

    iElt=    3
 EltName= SM_1
 Element= Reflector
```

```
 Surface= Conic
   KrElt=  -3.526787501D+00
   KcElt=  -2.670556022D+00
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0 -4.061145902D+00
  RptElt=   0d0  0d0 -5.4d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.338854098D+00
PropType=   Geometric
 nECoord=   -6

    iElt=    4
 EltName= ret1_1
 Element= Return
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  1.5d+00
  RptElt=   0d0  0d0  1.5d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   0d0
PropType=   Geometric
 nECoord=   -6

    iElt=    5
 EltName= ret2_1
 Element= Return
 Surface= Conic
   KrElt=  -6.329334641D+00
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -4.829334641D+00
  RptElt=   0d0  0d0 -4.829334641D+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   6.329334641D+00
PropType=   NF1
 nECoord=   -6

    iElt=    6
 EltName= CoroMask
 Element= Obscuring
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  1.5d+00
  RptElt=   0d0  0d0  1.5d+00
  IndRef=   1d0
  Extinc=   1d22
```

```
    nObs=    1
 ObsType=   Circle
  ObsVec=   1.5d-05  0d0  0d0
    xObs=   1d0  0d0  0d0
  ApType=   Circular
   ApVec=   2d0  0d0  0d0
    zElt=   1.d+30
PropType=   NF2
 nECoord=   -6

    iElt=    7
 EltName= ret2_2
 Element= Return
 Surface= Conic
   KrElt=  -6.329334641D+00
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  7.829334641D+00
  RptElt=   0d0  0d0  7.829334641D+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=  -6.329334641D+00
PropType=   Geometric
 nECoord=   -6

    iElt=    8
 EltName= ret1_2
 Element= Return
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  1.5d+00
  RptElt=   0d0  0d0  1.5d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   0d0
PropType=   Geometric
 nECoord=   -6

    iElt=    9
 EltName= SM_2
 Element= Reflector
 Surface= Conic
   KrElt=  -3.526787501D+00
   KcElt=  -2.670556022D+00
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  7.061145902D+00
  RptElt=   0d0  0d0  8.4d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.338854098D+00
PropType=   Geometric
 nECoord=   -6

    iElt=   10
 EltName= PM_2
 Element= Reflector
```

```
 Surface= Conic
   KrElt=  -1.08d+01
   KcElt=  -1d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  3d0
  RptElt=   0d0  0d0  3d0
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   5.4d+00
PropType=   Geometric
 nECoord=   -6

    iElt=   11
 EltName= LyotStop
 Element= Obscuring
 Surface= Conic
   KrElt=  -1.d+30
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  8.4d+00
  RptElt=   0d0  0d0  8.4d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    3
 ObsType=   Circle
  ObsVec=   5.d-01  0d0  0d0
 ObsType=   Rectangle
  ObsVec=  -2.5d-01  2.5d-01 -2d0  2d0
 ObsType=   Rectangle
  ObsVec=  -2d0  2d0 -2.5d-01  2.5d-01
    xObs=   1d0  0d0  0d0
  ApType=   Circular
   ApVec=   2d0  0d0  0d0
    zElt=   1.d+30
PropType=   Geometric
 nECoord=   -6

    iElt=   12
 EltName= PM_3
 Element= Reflector
 Surface= Conic
   KrElt=  -1.08d+01
   KcElt=  -1d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  1.38d+01
  RptElt=   0d0  0d0  1.38d+01
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   5.4d+00
PropType=   Geometric
 nECoord=   -6

    iElt=   13
 EltName= SM_3
 Element= Reflector
```

```
 Surface= Conic
   KrElt=  -3.526787501D+00
   KcElt=  -2.670556022D+00
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  9.738854098D+00
  RptElt=   0d0  0d0  8.4d+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1.338854098D+00
PropType=   Geometric
 nECoord=   -6

    iElt=   14
 EltName= ret1_3
 Element= Return
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  1.53d+01
  RptElt=   0d0  0d0  1.53d+01
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   0d0
PropType=   Geometric
 nECoord=   -6

    iElt=   15
 EltName= ret2_3
 Element= Return
 Surface= Conic
   KrElt=  -6.329334641D+00
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  8.970665359D+00
  RptElt=   0d0  0d0  8.970665359D+00
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   6.329334641D+00
PropType=   FarField
 nECoord=   -6

    iElt=   16
 EltName= foc_pln
 Element= FocalPlane
 Surface= Flat
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  1.53d+01
  RptElt=   0d0  0d0  1.53d+01
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   0d0
PropType=   Geometric
 nECoord=   -6
```

```
nOutCord=    5
    Tout=   1d0  0d0  0d0  0d0  0d0  0d0  0d0
            0d0  1d0  0d0  0d0  0d0  0d0  0d0
            0d0  0d0  0d0  1d0  0d0  0d0  0d0
            0d0  0d0  0d0  0d0  1d0  0d0  0d0
            0d0  0d0  0d0  0d0  0d0  0d0  1d0
```

A.7 Image Simulation Example

A.7.1 ImageDemo.jou

A.7.2 ImageDemo.in

```
ChfRayDir=  0d0  0d0 -1d0
ChfRayPos=  0d0  0d0  6.d+02
 zSource=   1d22
  IndRef=   1d0
  Extinc=   0d0
 Wavelen=   7.d-04
    Flux=   1.093342085D+00
GridType=   Circular
Aperture=   3.8d+02
Obscratn=   0d0
nGridpts=   33
   xGrid=  -1d0  2.033814221D-09  0d0
   yGrid=  -2.033814221D-09 -1d0  0d0
    nElt=   13

    iElt=    1
 EltName= SEC_OBS
 Element= Obscuring
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  4.002557400D+02
  RptElt=   0d0  0d0  4.002557400D+02
  IndRef=   1d0
  Extinc=   1d22
    nObs=    2
 ObsType=   Circle
  ObsVec=   4.67036d+01  0d0  0d0
 ObsType=   Rectangle
  ObsVec=   0d0  1.75d+02  2.39d+01  2.69d+01
    xObs=   9.896513868D-01  1.434926220D-01  0d0
  ApType=   Circular
   ApVec=   1.75d+02  0d0  0d0
    zElt=   1d22
PropType=   Annulus
 nECoord=   -6

    iElt=    2
 EltName= SPIDER_2
 Element= Obscuring
 Surface= Conic
```

```
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  4.002557399D+02
  RptElt=   0d0  0d0  4.002557400D+02
  IndRef=   1d0
  Extinc=   1d22
    nObs=    1
 ObsType=   Rectangle
  ObsVec=   0d0  1.75d+02  2.39d+01  2.69d+01
    xObs=  -6.190939493D-01  7.853169309D-01  0d0
  ApType=   Circular
   ApVec=   1.75d+02  0d0  0d0
    zElt=   1d22
PropType=   Annulus
 nECoord=   -6

    iElt=    3
 EltName= SPIDER_3
 Element= Obscuring
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0  4.002557398D+02
  RptElt=   0d0  0d0  4.002557400D+02
  IndRef=   1d0
  Extinc=   1d22
    nObs=    1
 ObsType=   Rectangle
  ObsVec=   0d0  1.75d+02  2.39d+01  2.69d+01
    xObs=  -3.705574375D-01 -9.288095529D-01  0d0
  ApType=   Circular
   ApVec=   1.75d+02  0d0  0d0
    zElt=   1d22
PropType=   Annulus
 nECoord=   -6

    iElt=    4
 EltName= PRI_MIR
 Element= Reflector
 Surface= Zernike
   KrElt=  -9.337886403D+02
   KcElt=  -1.009724001D+00
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0  0d0
  RptElt=   0d0  0d0  0d0
  IndRef=   1d0
  Extinc=   1d22
ZernCoef=   0d0  0d0  0d0 -1.415805330D-04 -3.404653098D-05  5.412244128D-06
            1.464334396D-05 -5.046596565D-05  1.587508743D-05 -3.536721061D-
05 -1.951889059D-06 -2.971194787D-05
            1.561541879D-05 -8.027617601D-06  8.191919142D-06  0d0  0d0  0d0
            0d0  0d0  0d0  0d0  0d0  0d0
            0d0  0d0  0d0  0d0  0d0  0d0
            0d0  0d0  0d0  0d0  0d0  0d0

    pMon=   0d0  0d0  0d0
    xMon=   1d0  0d0  0d0
    yMon=   0d0  1d0  0d0
    zMon=   0d0  0d0  1d0
    lMon=   1.75d+02
    nObs=    0
  ApType=   None
    zElt=   1d22
```

```
PropType=   Geometric
 nECoord=   -6

    iElt=    5
 EltName= SEC_MIR
 Element= Reflector
 Surface= Conic
   KrElt=  -1.565935400D+02
   KcElt=  -1.932283999D+00
  psiElt=   0d0  0d0  1d0
  VptElt=  -7.887151838D-04 -6.647047308D-05  4.002602400D+02
  RptElt=  -7.887151838D-04 -6.647047308D-05  4.002602400D+02
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    6
 EltName= COR_S1
 Element= Refractor
 Surface= Conic
   KrElt=  -4.241546000D+01
   KcElt=   0d0
  psiElt=   0d0  0d0  1d0
  VptElt=   0d0  0d0 -3.988054000D+01
  RptElt=   0d0  0d0 -4.260596000D+01
  IndRef=   1.61783d+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    7
 EltName= COR_S2
 Element= Refractor
 Surface= Conic
   KrElt=  -3.15595d+01
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0 -4.138168000D+01
  RptElt=   0d0  0d0 -4.260596000D+01
  IndRef=   1.657863000D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    8
 EltName= COR_S3
 Element= Refractor
 Surface= Conic
   KrElt=  -5.533237600D+02
```

```
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0 -4.533138000D+01
  RptElt=   0d0  0d0 -4.260596000D+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=    9
 EltName= FLT_S1
 Element= Refractor
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0 -4.763008000D+01
  RptElt=   0d0  0d0 -4.9d+01
  IndRef=   1.535018000D+00
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   10
 EltName= FLT_S2
 Element= Refractor
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0 -5.163058000D+01
  RptElt=   0d0  0d0 -4.9d+01
  IndRef=   1d0
  Extinc=   0d0
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   11
 EltName= IMG_RET
 Element= Return
 Surface= Conic
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   0d0  0d0 -5.175758000D+01
  RptElt=   0d0  0d0 -5.175758000D+01
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

    iElt=   12
```

```
 EltName= EXIT_RET
 Element= Return
 Surface= Conic
   KrElt=  -6.874662920D+01
   KcElt=   0d0
  psiElt=   1.095041158D-04 -5.786697499D-06 -9.999999940D-01
  VptElt=  -7.904014798D-05 -6.661258889D-06  1.698904900D+01
  RptElt=  -7.904014798D-05 -6.661258889D-06  1.698904900D+01
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   6.874662920D+01
PropType=   FarField
 nECoord=   -6

    iElt=   13
 EltName= IMG_PL
 Element= FocalPlane
 Surface= Flat
   KrElt=  -1d22
   KcElt=   0d0
  psiElt=   0d0  0d0 -1d0
  VptElt=   7.421487214D-01 -4.232318881D+00 -5.175758000D+01
  RptElt=   0d0  0d0 -5.175758000D+01
  IndRef=   1d0
  Extinc=   1d22
    nObs=    0
  ApType=   None
    zElt=   1d22
PropType=   Geometric
 nECoord=   -6

nOutCord=    5
    Tout=   1d0  0d0  0d0  0d0  0d0  0d0  0d0
            0d0  1d0  0d0  0d0  0d0  0d0  0d0
            0d0  0d0  0d0  1d0  0d0  0d0  0d0
            0d0  0d0  0d0  0d0  1d0  0d0  0d0
            0d0  0d0  0d0  0d0  0d0  0d0  1d0
```

A.8 S-MACOS Example Files

A.8.1 smacosvars.cmn

```
C*********************************************************************
C       Begin file smacosvars.cmn

C   This file defines call-line variables and common blocks that give
C   user programs direct access to MACOS variables. The form of the
C   SMACOS call line is:

C         CALL SMACOS(command,CARG,DARG,IARG,LARG,RARG,
C     &    OutRaySpot,NomRayDir,RMSWFE,NomOPDMat,PertOPDMat,
C     &    PixArray,Cmatrix)


C   The param.cmn file defines array sizes for SMACOS call-line and
```

```
C   internal MACOS variables. The param.cmn file should be copied from
C   param128.cmn, param256.cmn, etc. -- whichever size of SMACOS you
C   are using:

       INCLUDE 'param.cmn'

C   These are the SMACOS call-line variables:

       CHARACTER*132 command
       CHARACTER*32 CARG(9)
       REAL*8 DARG(9)
       INTEGER IARG(9)
       LOGICAL LARG
       REAL*4 RARG(9), OutRaySpot(mRay, 4), NomRayDir(2, mRay),
     &  RMSWFE, NomOPDMat(mpts, mpts), PertOPDMat(mpts, mpts),
     &  PixArray(mPix, mPix), Cmatrix(7, mCm, bRay)

C   These are other SMACOS common blocks, that you may use or comment
C   out as you wish. The ifEcho flag, if set to .TRUE., provides for
C   full printout of SMACOS operations. If set to .FALSE., reduced
C   printout is provided.

       LOGICAL ifEcho
       COMMON /SCIO/ifEcho

       INTEGER npts,NoiseSeed(2)
       REAL*4 CntrSpot(3)
       COMMON /UserCommon/npts,NoiseSeed,CntrSpot

C   These includes provide access to internal MACOS variables. They can
C   be commented out if you do not plan to use them:

       INCLUDE 'elt.cmn'
       INCLUDE 'src.cmn'

C   These includes provide access to internal MACOS scratch arrays. They
C   should be commented out if you do not plan to use them:

       LOGICAL L1(md2)
       INTEGER DrawEltVec(mDrawElt,mDrawRay)
       REAL*4 R1(mdttl,mdttl),RV1(md2),RV2(md2)
       REAL*8 D1(mdttl,mdttl),D2(mdttl,mdttl)
       REAL*8 DV1(md2),DV2(md2)
       COMMON /SCRATCH/ D1,D2
       EQUIVALENCE (D1(1,1),RV1(1)),(D1(1,1),DV1(1)),(D2(1,1),DV2(1)),
     &  (D1(1,1),R1(1,1)),(D2(1,1),RV2(1))

c END of smacosvars.cmn

C*********************************************************************
```

A.8.2 smacosExample.f

```
C*********************************************************************
C       Begin file smacosExample.f

       IMPLICIT NONE

C  SMACOS variable declarations

       INCLUDE 'smacosvars.cmn'

C  These are local variables:
```

```
      INTEGER

      REAL*4

      REAL*8

C   Load prescription file

      command='OLD'
      CARG(1)='cassExitPupil'
      CALL SMACOS(command,CARG,DARG,IARG,LARG,RARG,
     &  OutRaySpot,NomRayDir,RMSWFE,NomOPDMat,PertOPDMat,
     &  PixArray,Cmatrix)

C   Set plot type

      command='GRAY'
      CALL SMACOS(command,CARG,DARG,IARG,LARG,RARG,
     &  OutRaySpot,NomRayDir,RMSWFE,NomOPDMat,PertOPDMat,
     &  PixArray,Cmatrix)

C   Set system stop

      command='STOP'
      CARG(1)='ELT'
      IARG(1)=3
      DARG(1)=0d0
      DARG(2)=0d0
      CALL SMACOS(command,CARG,DARG,IARG,LARG,RARG,
     &  OutRaySpot,NomRayDir,RMSWFE,NomOPDMat,PertOPDMat,
     &  PixArray,Cmatrix)

C   Set exit pupil

      command='FEX'
      IARG(1)=5
      CARG(1)='YES'
      CALL SMACOS(command,CARG,DARG,IARG,LARG,RARG,
     &  OutRaySpot,NomRayDir,RMSWFE,NomOPDMat,PertOPDMat,
     &  PixArray,Cmatrix)

C   Plot OPD at exit pupil after resetting ray grid size

      nGridPts=128
      command='RESET'
      CALL SMACOS(command,CARG,DARG,IARG,LARG,RARG,
     &  OutRaySpot,NomRayDir,RMSWFE,NomOPDMat,PertOPDMat,
     &  PixArray,Cmatrix)

      command='OPD'
      IARG(1)=5
      CALL SMACOS(command,CARG,DARG,IARG,LARG,RARG,
     &  OutRaySpot,NomRayDir,RMSWFE,NomOPDMat,PertOPDMat,
     &  PixArray,Cmatrix)

C   Plot detector intensity after resetting ray grid size

      nGridPts=96
      command='RESET'
```

```
         CALL SMACOS(command,CARG,DARG,IARG,LARG,RARG,
     &  OutRaySpot,NomRayDir,RMSWFE,NomOPDMat,PertOPDMat,
     &  PixArray,Cmatrix)

         command='INTENSITY'
         IARG(1)=6
         CALL SMACOS(command,CARG,DARG,IARG,LARG,RARG,
     &  OutRaySpot,NomRayDir,RMSWFE,NomOPDMat,PertOPDMat,
     &  PixArray,Cmatrix)

C   Plot detector using a log10 stretch

         command='STRETCH'
         CARG(1)='LOG10'
         CALL SMACOS(command,CARG,DARG,IARG,LARG,RARG,
     &  OutRaySpot,NomRayDir,RMSWFE,NomOPDMat,PertOPDMat,
     &  PixArray,Cmatrix)

         command='INTENSITY'
         IARG(1)=6
         CALL SMACOS(command,CARG,DARG,IARG,LARG,RARG,
     &  OutRaySpot,NomRayDir,RMSWFE,NomOPDMat,PertOPDMat,
     &  PixArray,Cmatrix)

C   Quit!

         command='QUIT'
         CALL SMACOS(command,CARG,DARG,IARG,LARG,RARG,
     &  OutRaySpot,NomRayDir,RMSWFE,NomOPDMat,PertOPDMat,
     &  PixArray,Cmatrix)

         STOP
         END

C       End file smacosExample.f
C*************************************************************************
```