SPLAT v5.0 USERS' GUIDE

by

Derek Lee, Dave Newmark, Kenny Toh,
Phil Flanner, and A. R. Neureuther

Memorandum No. UCB/ERL M95/13

1 March 1995

SPLAT v5.0 USERS' GUIDE

by

Derek Lee, Dave Newmark, Kenny Toh,
Phil Flanner, and A. R. Neureuther

Memorandum No. UCB/ERL M95/13

1 March 1995

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

1.0 Table of Contents

A User's Guide for SPLAT version 5.0

Electronics Research Laboratory

University of California, Berkeley

1.1 Copyright

Notice: It is the intention of the SAMPLE Group to facilitate both the rapid incorporation of its programs into commercial products and the reuse in code distributed by research projects of other universities. Such redistribution of the Berkeley code requires a separate and more detailed license from the Office of Technology Licensing. A standard nonexclusive and royalty free agreement is available for a nominal fee provided the licensee agrees to preset terms such as not charging for the Berkeley code itself, acknowledging the source of the code and sponsor, and controlling redistribution.

1.2 Letter of Assurance

Dear Department of Electrical Engineering and Computer Sciences:

[Name of Organization or Institution] hereby assures the University of California that the software and documentation provided hereunder or any immediate product (including processes and services) produced directly by use of the software and/or documentation is not intended to and will not be shipped, either directly or indirectly, to Afghanistan, the People's Republic of China, or any country listed in Country Group Q, S, W, Y, or Z, as specified Part 770 of the Export Administration Regulations of the United States Department of Commerce, as of May 30, 1989.

[Name of Organization or Institution] hereby further assures the University of California that the software and documentation provided hereunder and the immediate product (including processes and services) produced directly by use of the software and or/documentation will not be made available to or for use by or for military or police entities of the Republic of South Africa or Namibia and that the software and documentation will not be made available to or for use by or for the apartheid-enforcing entities identified in Part 785 of the Export Administration Regulations of the United States Department of Commerce, as of May 30, 1989.

Licensee agrees that it will abide by any new Export Administration Regulations regarding the re-export of technical data as well as the use of technical data by the Republic of South Africa and Namibia issued by the United States Department of Commerce subsequent to execution of this License Agreement. Regulations and information may be obtained from the Office of Export Administration, International Trade Administration, Department of Commerce, Washington D.C. 20230.

The provisions of this letter of assurance shall survive and continue after any termination of any agreement under which the software and documentation is provided.

Signed by member of your organization

1.3 Contributors

Phil Flanner	Initial Version
Kenny Toh	Enhanced Version including 2D Numerical Integration, Primary Aberrations, and Annular Illumination.
David Newmark	Quadrapole Illumination, Hitachi Filter
Derek Lee	Release 4.2 and 5.0 Enhancements

2.0 Introduction

2.1 Overview

SPLAT is a FORTRAN program, based on the Hopkins' theory of partially coherent imaging, that simulates two-dimensional projection-printing with partial coherence.

The program is capable of simulating the effects of lens aberrations, apodization, spatial filtering, focus and magnification effects for high NA, and modified illumination. Annular, offaxis and quadrupole illumination sources can be also be simulated. True focus and magnification effects are included. The program can simulate the five primary lens aberrations: coma, astigmatism, distortion, spherical aberration, and curvature/defocus. Apodization schemes, such as spatial filters on the lens, can be added to the illumination system. New in this version of SPLAT are thin-film interference effects, table-lookup of the pupil function, representation of lens aberrations by Zernike polynomials and CodeV data, simulation of arbitrarily shaped sources with non-uniform intensity, and source amplitude apodization.

2.2 Basic Features

The mask is specified in the input file as a set of rectangles and triangles whose size, position, relative transmittance, and phase are specified by the user. The mask can be previewed with the DRAWMASK. The 2D masks allow the user to examine the effects of two-dimensional structures such as elbows and squares or simulate phase-shifted masks. In addition, the transmission cross-coefficients (TCCs) associated with a particular mask size and illumination condition can be saved and reused with different mask patterns.

For analysis of the resulting images, 2D or 3D image intensity data points can be saved. The 3D data points can be plotted on a contour plot using a commercial graphics package such as DISSPLA(ISSC). Alternatively, the plotting programs DRAWPLOT, CONTOUR, and PDRAW are included and can be used to draw 2D intensity cutlines, intensity contours, and 3D pupil map plots respec-

tively on an X11 window or in PostScript. The 2-D data points can also be fed into SAMPLE for simulation of processing steps based on that intensity profile. The CONTOUR, PDRAW, and DRAWPLOT programs are included as part of the UNIX-based SPLAT imaging package but are not with the version of SPLAT released as part of SAMPLE 1.8. The PC and Macintosh versions of SPLAT also do not come with plotting packages.

2.3 Basic Aberration Features

At present, the SPLAT program can handle any combination of the five primary lens aberrations:

1. Spherical Aberration

   This aberration is normally present on all uncorrected lenses, and is due mainly to the failure of the paraxial approximation in the ideal lens law. Rays that pass though the outer zones of a lens are deflected more than those that pass through the inner zones, and as a result, these rays do not pass through a common focus. Spherical aberration is the only aberration that is independent of the object position relative to the lens axis: all points on the image plane are affected similarly.

2. Coma

   Coma results from the unequal bending of parallel rays from an off-axis object. Rays parallel to the lens axis will come to a common focal point, but if these rays are shifted slightly (approx. 5 degrees) off axis, they will not focus, primarily because each ray "sees" a different amount of glass. In effect, coma is caused by unequal magnification in different zones of the lens, due to failure of the paraxial lens law.

3. Astigmatism

   Off-axis objects cause different focus points for different ray planes. The object sees a thicker lens width along the sagittal plane than the tangential plane, which results in different focal points for each of these planes. All spherical lenses have astigmatism for off-axis objects.

4. Curvature

   Strictly speaking, curvature is not an aberration at all, but comes about because flat images are desired instead of the natural spherical images. Curvature is similar to defocus.

5. Distortion

   Distortion only occurs when the other lens aberrations are present, and when the aperture is placed some distance away from the lens. This aberration is due mainly to different magnification at the outer zones of the lens.

These five primary aberrations, called both Seidel Aberrations and 3rd (lowest) order ray aberrations, can be expressed as functions of the object location (rela-

tive to the lens axis) and the image location or spatial frequency (and harmonics). For a more detailed treatment, please refer to the paper "Identifying and Monitoring Effects of Lens Aberrations in Projection Printing", *SPIE Vol 772: Optical Microlithography VI*, Santa Clara, CA, March 1987, by Toh and Neureuther.

The primary as well as high-order aberrations can be expressed using Zernike polynomials. (MORE).

2.4 Major Changes for SPLAT Version 4.2

1. In lens filtering was added (Statement 29) using a piecewise linear magnitudes and phase as well as the mathematical functions for two focus spots introduced by Fukuda of Hitachi.
2. Modified illumination was added. For annular illumination (Statement 5) a sigma inner radius is used. For quadropole illuminators (Statement 30) individual illumination spot locations and sizes are used.
3. True defocus (Statement 4) was added in which $\sqrt{1-ap^2}$ is used instead of the approximation $1-\frac{a}{2}p^2$.
4. An obliquity factor (Statement 31) was added to account for the field orientation relative to the direction of propagation in reduction systems.
5. When using Statement 30 it is no longer necessary to specify the system partial coherence (Statement 5). It is now automatically computed as the smallest sigma which will overlap all illumination spots. It is also not required to specify 'force = 2' in Statement 10. This option will automatically be invoked when Statement 30 is used.
6. Specifying 'mode = 1' in Statement 10 stores the TCCs for N harmonics where N counts both positive and negative harmonics. Statement 19 previously reported that N/2 harmonics were read from the saved TCC files. Statement 19 now reports that N harmonics are read from the saved TCC files to be consistent with Statement 10.

2.5 Major Changes for SPLAT Version 5.0

1. Thin-film interference effects (Statement 35) have been included. The mask image can be calculated within the photoresist which includes reflections off the underlying thin-film stack. Additional unix-scripts are provided to run SPLAT in a batch manner for each layer in the photoresist.
2. A table-lookup of the pupil function (Statement 32) was implemented to speed up the thin-film calculations.
3. Instead of always calculating the Fourier transform of the ideal mask, an option for reading the mask Fourier coefficients from a file (Statement 37)

generated by the simulation program TEMPEST which models mask topography effects has been added.

4. The phase of pupil function can now be represented by up to 64 Zernike Polynomials (Statement 36).
5. A pupil function can be initialized by reading wavefront data from the output of CODEV (Statement 34).
6. The pupil function can be output to a PDRAW or PLOTMTV file for examination (Statement 33).
7. Arbitrary shaped illumination sources with non-uniform intensity can now be simulated. They are constructed by specifying any number of geometric primitives.
8. Optical transmission with apodization (OTA) was implemented.
9. Aperture blocks can now be specified (Statement 38).
10. Illumination spots (Statement 30) can be specified with inner radii to allow for annular illumination type schemes.
11. The earlier versions of SPLAT left it up to the user to calculate the partial coherence of the imaging system when using spot illumination. Now the system partial coherence is automatically set as the smallest radius that will cover all source spots regardless of what the user specifies.

2.6 About this Manual

Section 2.0	**Introduction** This section provides a general description of both SPLAT and this manual.
Section 3.0	**Installation** A guide to installing the software is given here.
Section 4.0	**Tutorial** Using this quick, simple example serves as both a verification for installation and a tutorial for the new user.
Section 5.0	**Command Descriptions** This section describes each of the commands used in a SPLAT input file.
Section 6.0	**Examples** Eighteen examples are given, demonstrating the different features of SPLAT.

Section 7.0	**Miscellaneous Utilities** Miscellaneous utilities for running SPLAT in batch mode for its new features are described.
Section 8.0	**Appendix A : Zernike Polynomials** A list of the first 64 Zernike Polynomials is provided.

3.0 Installation

This program has been compiled and run success- fully on a number of different platforms, including UNIX (Vax 11/780, SUN 4/280, IBM 6000/520, DEC 3100), VMS (IBM 3090 mainframe), the IBM PC and the Apple MacIntosh.

To install SPLAT 5.0 on a UNIX system:

12. **mkdir** *destdir* where *destdir* the directory to which you want to extract the files
13. **cd** *destdir*
14. **tar** xf */dev/tape* where */dev/tape* is the file name of the cartridge tape drive.
15. **make** using options described in Makefile

The PC and Apple versions of SPLAT come with an executable file which can be transferred directly to the respective machine.

4.0 Tutorial

This example provides the user both a test to verify installation and a tutorial to introduce the basic concepts of SPLAT. The input file, output file, and DRAW-PLOT plot are given.

4.1 Input File

Splat uses a parser routine to read its input statements; the routine, in essence, recognizes numbers and characters, but not keywords. Statements can be separated by semicolons or placed on separate lines. For interactive use of the program, semicolons should be used as statement separators. Without the semicolon, the program must read an extra line to see if it is a continuation line before acting on that input statement. A statement may be continued on more than one line by placing an ampersand (&) as the first character of each continuation line. All characters on the input line are ignored except numbers, quoted strings, semicolons, an ampersand in the first column, and pound signs. The

pound sign (#) indicates that the remainder of the line is a comment to be ignored by the parser.

For example, each of the following three inputs cause the same action, setting the light source wavelength $\lambda = 0.436$ microns and numerical aperture NA = 0.28:

```
2 0.436; 3 0.28;

Statement 2 : wavelength = 0.436 microns
Statement 3 : na = 0.28;

Statement 2 : Set wavelength
&             to 0.436 microns;
Statement 3 : 0.28 # Set numerical aperture to 0.28
```

Below is an input file that demonstrates this syntax. The simulation is of a 1.0μm x 1.0μm opaque diamond, constructed from triangles in STATEMENT 8, in a transparent mask. For any simulation, the mask size must be limited since the computation time is proportional to the fourth power of the area of the mask. A practical maximum, where results appear in five minutes on a Sun 4, is an unfolded(total) mask size of 4.0 λ/NA x 4.0 λ/NA. Please note that statement 10 causes SPLAT to calculate the Fourier coefficients. Any change to the imaging system after that statement will not have any effect.

The file is as follows:

```
#
# Projection Lithography using SPLAT -- Test
#
Statement 1 : Printlevel 3              ;# set print level
Statement 2 : lambda = 0.436 um         ;# wavelength
Statement 3 : NA = 0.28                 ;# numerical aperture
Statement 4 : Defocus= 0.0 um           ;# defocus
Statement 5 : Sigma = 0.7               ;# coherence factor
Statement 6 : mask = 2um x 2um
&             at 1 transmittance        ;# define the working area
Statement 8 : cutout = (0.0, 0.0)        # triangular opaque pattern
&             0.5 x 0.5 at -1           ;# define the mask openings
Statement 10:                           ;# calculate the Fourier coeffs
Statement 14: intensity (-2,0) .. (2,0)
&             to 'test1.plot'           ;# 2-D intensity profiles
Statement 0 : end;
```

4.2 Output File

The following is the line printer output produced when SPLAT is given the above input file.

```
2D optical imaging with aberrations
apodization and modified illumination -- V4.2
                                3/23/93 -- DN/DL/KT
Trial 1: Print level     = 3
Trial 2: Lambda          = 0.4360 microns
Trial 3: N.A.            = 0.2800
Trial 4: Defocus         = 0.0000 microns [ 0.0000 Rayleigh Units ]
Trial 5: Sigma        = 0.7000
Trial 5: Sigma In     = 0.0000
Trial 6: Field size = 2.0000 x 2.0000 @ 1.0000 scale =1.00
#      6: x size requires 8 harmonics.
#      6: y size requires 8 harmonics.
#      8: Add triangle ( 0.000, 0.000)x( 0.500, 0.500) @-1.000 < 0.000>
Trial 8: Triangle = ( 0.000, 0.000) ( 0.000, 0.500) ( 0.500, 0.000)
Trial 10: Calculate image Fourier Transform
#      10: Symmetry : T( f1, g1, f2, g2) =
#      10:            T(-f1,-g1,-f2,-g2)
#      10:            T( f1,-g1, f2,-g2)
#      10:            T(-f1, g1,-f2, g2)
#      10:            T( f2, g2, f1, g1)
#      10: TCC computation time = 0.45000 sec.
#      10: Imaged with 8 by 8 harmonics
#      10: TCC calls: 361 zeros: 520
Trial 14: 3-decimal plot line : ( -2.000, 0.000)..( 2.000, 0.000) 50
       points saved in "test1.plot"
Trial 0 : End of session
          User Time (CPU) = 1.68000 sec, System Time = 0.28000 sec.
```

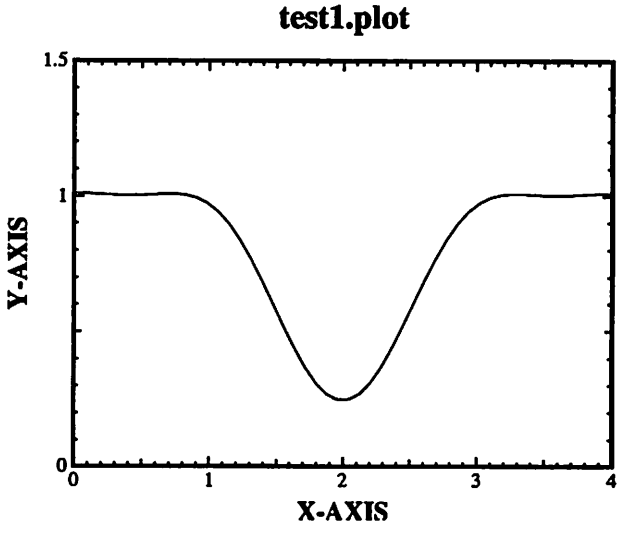
4.3 Plot

The easiest way of analyzing SPLAT output is by looking at contour or slice plots and by noting the differences between the plots when certain parameters are changed.

SPLAT output can be displayed using the contour plotting program, CONTOUR, which is intended for use on X-window graphics systems or with HP239X and HP2647/8 terminals. CONTOUR also produces plots in POSTSCRIPT; the POSTSCRIPT plots can be sent to an Apple Laserwriter for nice-looking plots.

Cross sections of the image can also be displayed with two dimensional packages such as DRAWPLOT, which displays 2D plots on X10/X11 windows, SUNVIEW displays, or HP2648 terminals. The program will also produce POSTSCRIPT plots, such as the one used in this tutorial, that can be sent to a printer.

The DRAWPLOT plot for this example is shown here.



5.0 Command Descriptions

This section gives a list of the input statements recognized by SPLAT. Arguments shown in brackets are optional (the brackets are not necessary when using the program). Strings (such as filenames) must be inside single or double quotes. Note that the word STATEMENT at the beginning of the input statement is not necessary.

[STATEMENT] 1 *printlevel*

STATEMENT 1 sets the level of diagnostic output generated by the program. PRINTLEVEL=1 produces no diagnostic output, while PRINTLEVEL=2 causes the program to echo the input lines to the output. PRINTLEVEL=3, on the other hand, is primarily for interactive/diagnostic use and causes the program to report when statement execution is completed by displaying the input parameters. The default value of PRINTLEVEL is 1.

[STATEMENT] 2 *wavelength*

This statement sets the wavelength, in micrometers, of the light used to illuminate the mask. Currently, the program is set up to accept only a single wavelength. The default is WAVELENGTH = 0.436 μm.

[STATEMENT] 3 *numerical.aperture*

The imaging system is projection-type, with an imaging lens numerical aperture equal to NUMERICAL.APERTURE. Default NUMERICAL.APERTURE is 0.28.

[STATEMENT] 4 *defocus [mode]*

The image can be calculated at a plane other than the plane of best focus. The distance from the plane of best focus, DEFOCUS, is measured in micrometers and defaults to 0.0 μm. Positive DEFOCUS is defined as being below the Gaussian plane, while negative DEFOCUS is above it.MODE is an integer flag used to specify the type of defocus. MODE = 1 specifies that true defocus ($\sqrt{1-\rho^2}$) will be used and is the default mode. MODE = 0 specifies that approximate defocus ($1-\rho^2/2$) will be used.

[STATEMENT] 5 *sigma [sigma_in]*

STATEMENT 5 sets the partial coherence factor, SIGMA, of the imaging system. Only values of partial coherence that lie between 0.0 (full coherence) and 1.0 (partial coherence) are accepted by the program. SIGMA defaults to 0.7, a value common to most projection printers. SIGMA_IN is used for annular illumination schemes. It indicates the radius of the illumination cone which will be blocked out. The actual illumination occurs only in the annulus from SIGMA_IN to SIGMA. SIGMA_IN defaults to 0.0.

[STATEMENT] 6 *xlength ylength [transmittance [scale]]*

The working area is the size of the mask which is to be imaged and is specified by XLENGTH and YLENGTH in micrometers. This specified area is actually only the first quadrant in Cartesian coordinates. The total/unfolded area is bounded by the coordinates (-XLENGTH,-YLENGTH), (-XLENGTH,YLENGTH), (XLENGTH,YLENGTH), (XLENGTH,-YLENGTH), and a periodic extension in both the X and Y directions is assumed. The optional TRANSMITTANCE is the initial transmittance of the mask (usually either 1 for transparent or 0 for opaque). The image calculation time is approximately proportional to the square of the imaged area, and to avoid long computation times, field sizes with total areas greater than 16 micrometers squared should be avoided. One way to do this is to take advantage of symmetry - see STATEMENT 7 for details. The default for TRANSMITTANCE is 0, while XLENGTH and YLENGTH both default to 2.0 micrometers. SCALE is a positive real number that can be used to scale the mask. This scale factor will also apply to any STATEMENTs 7, 8, 27, and 28 that come after a STATEMENT 6.

[STATEMENT] 7 *xcoord ycoord xlen ylen transmittance [phase]*

This statement is used to add a rectangular feature to the mask. The rectangle specified is mirrored across the x and y axes. Rectangles may overlap other

rectangles - it is the user's responsibility to make sure no part of the mask has transmittance greater than 1 or less than 0.

XCOORD = x coordinate of lower left corner of rectangle

YCOORD = y coordinate of lower left corner of rectangle

XLEN = length of rectangle in x direction

YLEN = length of rectangle in y direction

TRANSMITTANCE= transmittance of the rectangle relative to the current transmittance of the mask where it is to be placed.

PHASE = phase angle (normally 0 degrees). This option is used for phase shifted masks.

As an example, to specify a 2 μm x 2 μm transparent square in a 5 μm x 5 μm opaque mask, the following statements could be used:

STATEMENT 6 : 5 um x 5 um at 0 transmittance;

STATEMENT 7 : (2.0,2.0) 2.0 um x 2.0 um at 1 transmittance;

However, symmetry can and should be utilized to reduce the computation time by locating the center of the rectangle at the origin of the coordinate system in the following manner:

STATEMENT 6 : 2 um x 2 um at 0 transmittance;

STATEMENT 7 : (0.0,0.0) 1.0 um x 1.0 um at 1 transmittance;

In the last two input statements above, a 1 μm x 1 μm transparent square is defined in a 2 μm x 2 μm opaque mask. Because the program automatically makes even periodic extensions in the x and y directions, the region specified by the user is actually only the first quadrant in the cartesian coordinate system; the other three quadrants are the mirror images of the first, reflected across the x and y-axes respectively. Therefore, the 1 μm x 1 μm square defined above is actually only part of a 2 μm x 2 μm square, with the center of symmetry of that larger square located at the origin. In a similar manner, an opaque square on a transparent mask can be defined using the statements below, where the transmittance is 1 for the background, and -1 for the square.

STATEMENT 6 : 2 um x 2 um at 1 transmittance;

STATEMENT 7 : (0.0,0.0) 1.0 um x 1.0 um at -1 transmittance;

[STATEMENT] 8 *xcoord ycoord xlen ylen transmittance [phase]*; or

[STATEMENT] 8 *x1 y1 x2 y2 x3 y3 transmittance [phase]*;

Add a triangular aperture to the mask. The triangle specified is mirrored across the x and y axes. Triangles may overlap other triangles and rectangles - it is the user's responsibility to make sure no part of the mask has transmittance greater than 1 or less than 0. The triangle can be specified in two ways - as a right-angled triangle defined by the corner (right angle) point, base and height or as a general triangle defined by three points. Again, even periodic

extensions in the x and y directions are assumed. Please note that if only six arguments are provided, the program assumes that the triangle being defined is a right- angled triangle.

XCOORD = x coordinate of corner of right-angled triangle

YCOORD = y coordinate of corner of right-angled triangle

XLEN = base length of right-angled triangle (may be negative to flip triangle)

YLEN = height of right-angled triangle (may be negative to flip triangle)

X1 = x coordinate of point defining general triangle

Y1 = y coordinate of point defining general triangle

X2 = x coordinate of point defining general triangle

Y2 = y coordinate of point defining general triangle

X3 = x coordinate of point defining general triangle

Y3 = y coordinate of point defining general triangle

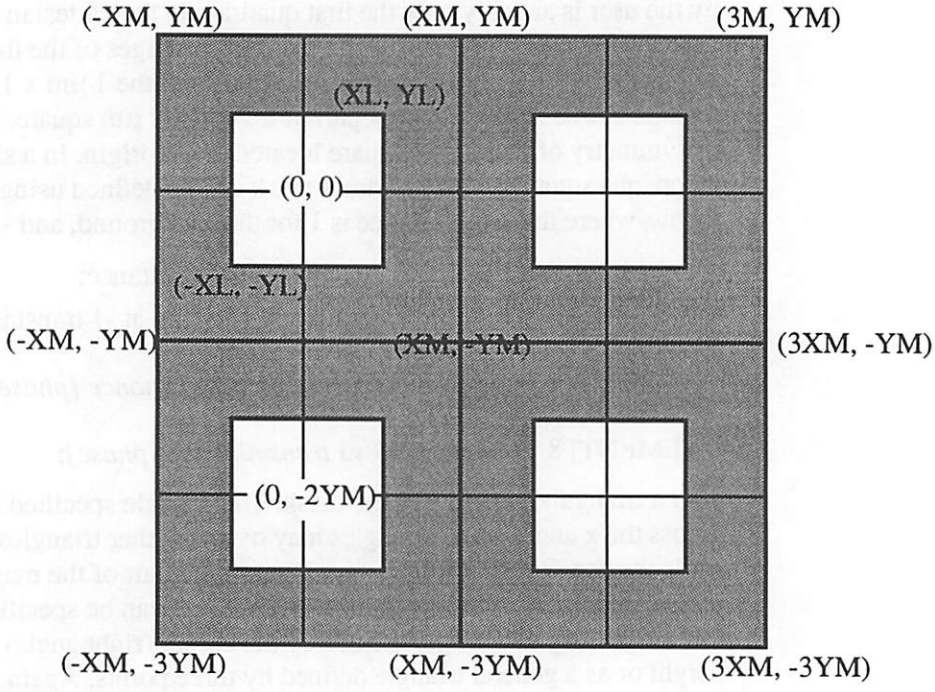
TRANSMITTANCE= transmittance of the triangle relative to the current transmittance of the mask where it is to be placed.

PHASE = phase angle (normally 0 degrees). This option is used for phase shifted masks.

Input Statements :

Statement 6 : XM x YM @0;

Statement 7 : (0,0) (XL,YL) @1;



[STATEMENT] 9 *nx ny [mode [xllc yllc xlen ylen [difflag]]] ['filename']*;

STATEMENT 9 is used to calculate the transmittance profile of the mask. The transmittance at any point is the sum of the working area(mask) transmittance (see STATEMENT 6) and the transmittances of all rectangles or triangles defined by STATEMENTs 7 and 8 that cover that point. The output file consists of a list of transmittances at points on a grid defined by NX and NY. The default is to calculate for the entire working area, but the user can specify a smaller (or larger) area. MODE defines the type of output obtained from this statement. MODE = 0 produces output to a specified file, MODE = 1 sends a crude contour plot to the terminal, while MODE = 2 does both.

NX = number of divisions along x axis (default = 20)

NY = number of divisions along y axis (default = 20)

MODE = type of output desired

XLLC = x coordinate of lower left corner of rectangle to be plotted

YLLC = y coordinate of lower left corner of rectangle to be plotted

XLEN = length of rectangle in x direction

YLEN = length of rectangle in y direction

DIFFLAG = 1 to limit spatial frequencies (normally not used)

FILENAME= name of file into which to store the calculated intensities.

[STATEMENT] 10 *[mode [force]] ['filename']*;

STATEMENT 10 orders the program to calculate the transmission cross-coefficients (TCCs) as well as the Fourier coefficients of the image intensity profile. This statement can take a long time to execute. MODE = 1 stores the TCCs in *binary* form in the file FILENAME. FORCE is an integer flag which forces the program to choose which of the 3 integration routines should be used for TCC calculation. FORCE = 0 is the default, FORCE = 1 uses a 1-dimensional integration routine, while FORCE = 2 uses a 2-dimensional integration scheme. When aberrations other than approximate defocus are used or when magnification is set, FORCE = 2 is used automatically.

[STATEMENT] 11 *[xllc yllc xlen ylen [nx ny]] ['filename']*;

This statement calculates the image intensity within a rectangle specified by XLLC, YLLC, XLEN, YLEN. The intensities are calculated at each point of a rectilinear NX by NY equally spaced grid within that rectangle. (Note : Maximum NX*NY is 10000) This output file may be used with a 3-dimensional plotting package to provide either a 3- dimensional intensity profile or an intensity contour plot. The parameters are :

XLLC = x coordinate of lower left corner of rectangle to be plotted

YLLC = y coordinate of lower left corner of rectangle to be plotted

XLEN = length of rectangle in x direction

YLEN = length of rectangle in y direction

NX = number of grid-points in the x direction (default is 50)

NY = number of grid-points in the y direction (default is 50)

FILENAME= name of file into which to store the calculated intensities.

[STATEMENT] 12 *'filename'*

Save the Fourier coefficients that describe mask transmittance and image intensity. This input line will produce a relatively large (approx. 30 kbytes) binary output file, which contains the Fourier coefficients as well as the imaging system parameters - numerical aperture, wavelength, coherence factor, mask size. The coefficients can be reloaded with STATEMENT 13 to generate more plot files of the same mask pattern.

[STATEMENT] 13 *'filename'*

STATEMENT 13 is used to load the Fourier mask and image coefficients previously saved with STATEMENT 12. Therefore, for a given set of imaging system parameters (N.A., wavelength, sigma, mask size and pattern), the intensity profile need only be calculated once with STATEMENT 10. STATEMENT 12 saves the data computed by STATEMENT 10 for later use, and STATEMENT 13 reloads this data. Note that the TCCs saved via STATEMENT 10 can be used for any mask pattern, as long as the field size remains constant. On the other hand, STATEMENT 12-13 saves the Fourier coefficients only, so these can be used only for one particular mask pattern.

[STATEMENT] 14 *xi yi xf yf [npts [mode]] ['filename'];*

Calculate the image intensity profile along the line joining points (XI,YI) and (XF,YF). The number of points along that profile and is optionally specified by NPTS, which defaults to 50. The intensity profile is output in the same format as a SAMPLE f77punch7 file. The 'x' values in the file represent the distance along the line from the point (XI,YI), while the 'y' values represent the light intensity normalized to 1.00. Again, MODE is an integer flag, normally 0, used to specify the format of the intensity profile. MODE = 1 outputs the intensities with 5 decimal spaces (as compared to 3 with MODE = 0). This option is particularly useful for analysis of small patterns such as defects, where the intensity is very low.

XI = x coordinate of initial plot point

YI = y coordinate of initial plot point

XF = x coordinate of final plot point

YF = y coordinate of final plot point

NPTS = number of points (default is 50, maximum is 500)

MODE = accuracy flag (defaults to 0)

[STATEMENT] 15 empty

[STATEMENT] 16 empty

[STATEMENT] 17 empty

[STATEMENT] 18 empty

[STATEMENT] 19 *'filename'*

STATEMENT 19 loads the transmission cross-coefficients(TCC) saved from STATEMENT 10. These TCCs can be reused for different mask patterns because the TCCs are only dependent on the mask size and not upon the patterns contained within it. Using these TCCs a new image can be recomputed very speedily (in approx. 38 cpu seconds).

[STATEMENT] 20 *xpos ypos*

This statement is used primarily to determine how the object is situated (perpendicular, parallel, etc.) relative to the lens axis. This statement does not work with CodeV pupil maps (STATEMENT 34) or with Zernike polynomials (STATEMENT 36). The object coordinates input through this statement will be normalized such that $XPOS^2 + YPOS^2 = 1$. For example, (XPOS,YPOS) = (1,0) means that the portion of the mask currently being simulated is located on the X-axis of the field. If the mask consists of vertical lines, then it is possible to say that the pattern is perpendicular to the lens axis. In contrast, if (XPOS,YPOS) = (0,1), the vertical pattern would be parallel to the lens axis. This distinction has been found to be important for aberrations such as coma and astigmatism which are orientation-dependent.

[STATEMENT] 21 *spherical_aberration*

[STATEMENT] 22 *coma*

[STATEMENT] 23 *astigmatism*

[STATEMENT] 24 *curvature*

[STATEMENT] 25 *distortion*

Statements 21-25 can be used to specify any combination of primary lens aberrations. The numbers specified (COMA, ASTIGMATISM, etc.) are used as multipliers to a predefined power series (Zernike polynomials) and as such, can be either positive or negative real values. It is easier to think of these multipliers as a measure of the deviations, measured in fractions of a wavelength, from the ideal spherical wavefront. In general, the deviation is related to the multiplier as $\Delta\lambda$ = multiplier / wavelength. For example, if λ =

0.5 μm and COMA = 0.1, this would mean that the wavefront has a maximum deviation of 0.2λ from the ideal spherical wavefront.

[STATEMENT] 26 empty

[STATEMENT] 27 *xcoord ycoord xlen ylen transmittance [phase]*

This statement is used to add a *non-symmetrical* rectangular feature to the mask. Unlike STATEMENT 7, the rectangle defined here is *not* mirrored across the x and y axes. Furthermore, the XCOORD and YCOORD coordinates of the lower left corner of the rectangle may be specified anywhere inside the four quadrants of the mask. Rectangles may overlap other rectangles - it is the user's responsibility to make sure no part of the mask has transmittance greater than 1 or less than 0.

XCOORD = x coordinate of lower left corner of rectangle

YCOORD = y coordinate of lower left corner of rectangle

XLEN = length of rectangle in x direction

YLEN = length of rectangle in y direction

TRANSMITTANCE= transmittance of the rectangle relative to the current transmittance of the mask where it is to be placed.

PHASE = phase angle (normally 0 degrees). This option is used for phase shifted masks.

As an example, to specify a 2 μm x 2 μm transparent square at the center of a 5 μm x 5 μm opaque mask, the following statements could be used :

STATEMENT 6 : 2.5 um x 2.5 um at 0 transmittance;

STATEMENT 27 : (-1.0,-1.0) 2.0 um x 2.0 um at 1 transmittance;

[STATEMENT] 28 *xcoord ycoord xlen ylen transmittance [phase]*; or

[STATEMENT] 28 *x1 y1 x2 y2 x3 y3 transmittance [phase]*;

Add a *nonsymmetrical* triangular aperture to the mask. Unlike STATEMENT 8, the triangle defined here is *not* mirrored across the x and y axes. Triangles may overlap other triangles and rectangles - it is the user's responsibility to make sure no part of the mask has transmittance greater than 1 or less than 0. The triangle can be specified in two ways - as a right-angled triangle defined by the corner (right angle) point, base and height; or as a general triangle defined by three points. Again, even periodic extensions in the x and y directions are assumed. Please note that if only six arguments are provided, the program assumes that the triangle being defined is a right-angled triangle.

XCOORD = x coordinate of corner of right-angled triangle

YCOORD = y coordinate of corner of right-angled triangle

XLEN = base length of right-angled triangle (may be negative to flip triangle)

YLEN = height of right-angled triangle (may be negative to flip triangle)

X1 = x coordinate of point defining general triangle

Y1 = y coordinate of point defining general triangle

X2 = x coordinate of point defining general triangle

Y2 = y coordinate of point defining general triangle

X3 = x coordinate of point defining general triangle

Y3 = y coordinate of point defining general triangle

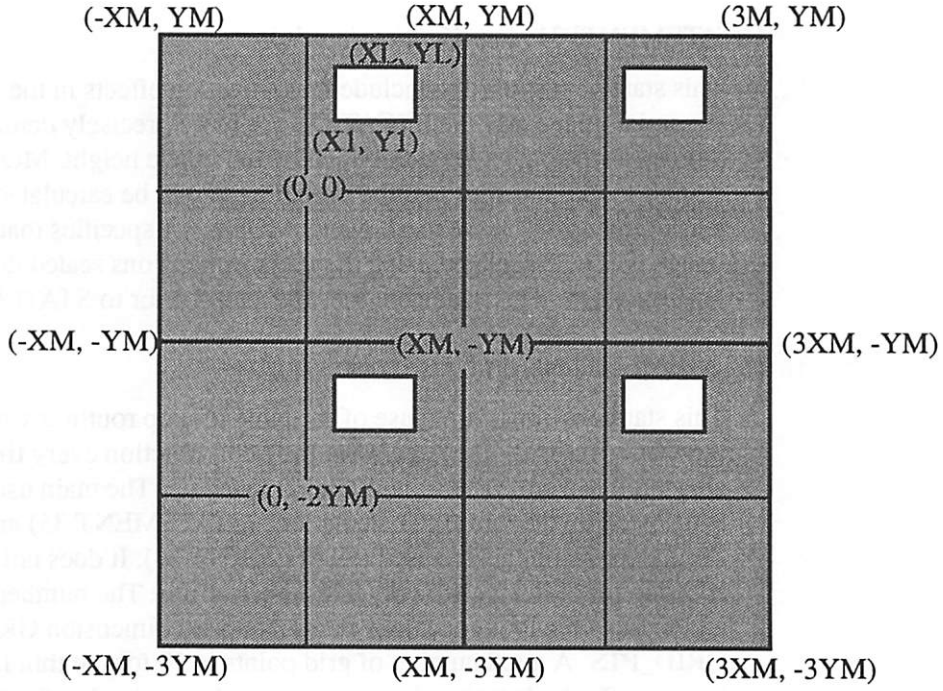
TRANSMITTANCE= transmittance of the triangle relative to the current transmittance of the mask where it is to be placed.

PHASE = phase angle (normally 0 degrees). This option is used for phase shifted masks.

Input Statements :

Statement 6 : XM x YM @0;

Statement 27 : (X1,Y1) (XL,YL) @1;



[STATEMENT] 29 *filter param1 param2*

STATEMENT 29 is used for apodization or in-lens filtering at the lens pupil. Only the Hitachi filter has been implemented so the filter must be 1. The two parameters for the Hitachi filter are β and θ. The Hitachi filter works by placing two distinct images along the optic axis. β is the distance of these images

from the original focal plane. θ is the phase difference between the images (in radians).

[STATEMENT] 30 *spot_radius [inner_spot_radius] Jx Jy*

STATEMENT 30 allows entry of a single illumination spot. The SPOT_RADIUS is the radius of the illumination spot while (Jx,Jy) indicate its coordinates in the entrance pupil of the system. The INNER_SPOT_RADIUS is used to create an off-axis annulus of illumination. The radius of the entrance pupil is normalized to 1. Multiple illumination spots are specified by using up to 10 of these statements. For example, the following commands define a quadrupole illumination scheme for a system with a partial coherence of 0.695. The system partial coherence is computed automatically as the smallest sigma which will overlap all illumination spots.

STATEMENT 30 : Spot_radius = 0.2 at (0.35, 0.35);

STATEMENT 30 : Spot_radius = 0.2 at (-0.35, 0.35);

STATEMENT 30 : Spot_radius = 0.2 at (0.35,-0.35);

STATEMENT 30 : Spot_radius = 0.2 at (-0.35,-0.35);

[STATEMENT] 31 *magnification [mode]*

This statement is used to include magnification effects in the calculation of the aerial image. MAGNIFICATION (or more precisely demagnification) is defined as the object height divided by the image height. MODE = 0, the default mode, specifies that the mask image will be calculated using the mask dimensions at the mask plane. MODE = 1 specifies that the mask image will be calculated using the mask dimensions scaled down by MAGNIFICATION. This statement must be called prior to STATEMENT 6.

[STATEMENT] 32 *[grid_pts [symmetry]]*

This statment turns on the use of the table lookup routines when calculating the TCCs. Instead of recalculating the pupil function every time, the values are stored in a table and subsequently looked up. The main use of this feature is to speed up the thin-film calculations (STATEMENT 35) and when many Zernike polynomials are used (STATEMENT 36). It does not provide any speedup for systems with only a few aberrations. The number of columns and rows of the lookup table is initialized with dimension GRID_PTS X GRID_PTS. A good number of grid points is 50 for the thin film cases--the results differ by less than 1% from those when not using the table. The maximum number of grid points is 400 and the default is 50. The SYMMETRY parameter refers to the use of symmetry in the calculation of the TCCs (STATEMENT 10) which normally reduces the computation time. When SYMMETRY = 0, symmetry in the TCC calculation is turned off. This will speed up the simulation time when the mask area is very small since the determination of the symmetry dominates the simulation time. Explicitly dis-

abling the symmetry is only permitted in conjunction with STATEMENT 35. By default SYMMETRY = 1 which is the state of normal operation.

[STATEMENT] 33 *mode [format_mode [output_scale [grid_scale]]]*

['filename']

This statement is used to print out the contents of the pupil function. The following MODEs are supported:

MODE	Output
0	Magnitude of the pupil function (computed)
1	OPD of the pupil function in wavelengths (computed)
2	Imaginary parts of the complex table values
3	Real parts of the complex table values
4	Fringe table values loaded using STATEMENT 34 (CodeV Data)
5	Zernike Polynomial values in wavelengths (computed)
6	Imaginary parts of complex table values (interpolated, *only for testing the table lookup function*)
7	Real parts of complex table values (interpolated, *only for testing the table lookup function*)

The FORMAT_MODE parameter defaults to a value of 0 which specifies that a PLOTMTV output file will be written. A value of 1 will create a PDRAW output file.

The OUTPUT_SCALE parameter is used to scale the physical size of output as desired. The GRID_SCALE parameter is used to limit the number of triangles in the output plot. The number of existing grid points along the length of the grid is divided by this factor. If the output file name is not specified, 'out.3D.plot' will be used. When the output file is created in the PDRAW format, the number of triangles may not be correctly specified in the header when the default grid size is changed. The correct number of triangles will be in the last line of the file. Manually replace this number in the appropriate place at the beginning of the file or run the Unix script described in the Miscellaneous Utilities section.

[STATEMENT] 34 *[scale] 'CODEV_wavefront_data_file'*

Statement 34 reads in a wavefront map generated from CODEV and initializes the pupil function. Currently, the CODEV input file can not exceed a grid size of 65, and odd square grid sizes should be used (e.g. 33 X 33). In this way, the data will fall exactly on the grid points in the internal table of SPLAT. The CODEV input file should be not have data for a particular row

on two separate lines. Data in this format was probably generated by doing a copy and paste of the screen instead of using CODEV to save the data to a file directly. The input file may have the header information normally generated and other extraneous lines as long as the lines do not begin with numbers. In addition blank spaces are not allowed between the data rows for each area of the pupil, and each data set of the pupil must be separated by at least one blank line. (Refer to the demo examples). The SCALE parameter specifies what fraction of a wavelength the values in the data file represent. The default is 0.01 μm.

A few notes on using Splat to Simulate Projection X-Ray Lithography:

A few unresolved issues remain. When sigma becomes too small, errors will be reported during the calculation of the TCCs for certain source positions. When this occurs, increase sigma until no errors are reported. However, a comparison of the intensity plots from the error-free run and from the error flagged result will likely show little or no differences.

[STATEMENT] 35 *resist_layer 'thin-film_data_file'*

Statement 35 allows the image to be calculated within the photoresist. The resist is split up into many thin layers as specified in the 'THIN-FILM_-DATA_FILE'. The format of the thin-film input file is given below. The correct number of parameters must be listed on each line and any number of spaces (but no tabs) may be used to separate them.

Thin-film Data File Input Format

substrate_n subsrtate_k medium_n medium_k wavelength

resist_n A B C dose thickness layers

layer1_n layer1_k thickness

layer2_n layer2_k thickness

...

layer12_n layer12_k thickness

The first four parameters are the real and imaginary parts of refractive index of the substrate and medium (Air) respectively. The fifth parameter is the wavelength in the initial medium. The next line specifies the resist properties: the real part of the index of refraction; the resist A, B, C parameters, used to calculate the imaginary part of the index of refraction; dose; thickness; and number of layers. Any lines following the resist parameters specify the real and imaginary parts of the index of refraction of layers under the photoresist and their thicknesses. Up to 12 layers may be specified. Refer to Example 15 for an actual input file.

[STATEMENT] 36 *[norm_mode [mult_mode]]*

'Zernike_polynomial_multiplier_file'

This statement causes the aberration function to be represented by a sum of Zernike polynomials. By default NORM_MODE is 0 which sets the normalization such that the jth multiplier represents the rms value of the jth aberration. Otherwise NORM_MODE is 1 which disables any normalization. Thus any normalization must be explicitly included in the multiplier file. The multipliers in front of each Zernike polynomial are provided in the 'ZERNIKE_POLYNOMIAL_MULTIPLIER_FILE'. By default MULT_MODE is 0 which treats the multipliers as fractions of a wave. When MULT_MODE is 1, the multipliers are specified in units of micrometers. All multipliers must be specified up to the last non-zero multiplier. Thereafter, all other multipliers need not be specified and will be set to zero. The input format is as follows:

[0] multiplier_term_0

[1] multiplier_term_1

. . .

[35] multiplier_term_35

The first parameter on each line is optional. It is provided to keep track of which multiplier corresponds to which Zernike polynomial term. For details on the implementation please refer to the actual code (file 'zp.f'). A list of the first 64 Zernike polynomials is given in Appendix A. STATEMENT 32 may also be used with the Zernike polynomials. This may decrease the computation time when many terms are used. The aberration function can become quite steep at the edges of depending on the specified multipliers. When this occurs SPLAT has difficulty in the integration routine for the calculation of the TCCs. To see a plot of the pupil function used, invoke STATEMENT 33 with MODE=5. When many terms are used STATEMENT 32 may be used to speed up the calculation.

[STATEMENT] 37 *'coefficients_file'*

Instead of calculating the Fourier transform of the ideal mask, statement 37 specifies that the mask Fourier coefficients be read from a file named 'COEFFICIENTS_FILE'. Such a file can be generated by the simulation program TEMPEST which models mask topography effects.

[STATEMENT] 38 *inner_aperture_radius1 [inner_aperture_radius2]*

This statement is used to specify the inner radii of an aperture inside the imaging system. The aperture is assumed to be ring-shaped, with a normalized outer radius of 1.0. The region from INNER_APERTURE_RADIUS1 down to INNER_APERTURE_RADIUS2 will be blocked. By default INNER_APERTURE_RADIUS2 is 0.0.

Statements 39 - 46 are used to define an arbitrary illumination source. Statements 40 - 43 are used to add geometric primitives to the discretized source. The intensity values of each are added together to form the source. Negative intensities are clipped to zero but intensities greater than 1 are allowed. The open field intensity is always normalized to 1 even when non-uniform intense illumination is used.

[STATEMENT] 39 *[source_grid_points [integration_mode]]*

This statement is used to specify the number of source grid points from edge to edge and should be odd so that the grid is perfectly centered. By default SOURCE_GRID_POINTS is set to 201 and its maximum value is 401. The INTEGRATION_MODE is by default 0 which is adaptive integration. When INTEGRATION_MODE is 1, the adaptive integration is disabled and trapezoidal integration is used.

[STATEMENT] 40 *x_center y_center radius intensity_value*

This statement adds a circle to the illumination source. The circle coordinates are located at the nearest grid points. For example:

40: (.2, .2) .3 1.0

defines an off-centered spot while

40: (.2, .2) .5 1.0

40: (.2, .2) .3 -1.0

defines an off-centered annulus.

[STATEMENT] 41 *x1 x2 width height intensity_value*

This statement adds a rectangle to the illumination source. The bottom left corner of the rectangle is located at (x1, y1).

[STATEMENT] 42 *x1 y1 x2 y2 x3 y3 intensity_value*

This statement adds a triangle to the illumination source. The triangle has vertices located at (x1, y1), (x2, y2), and (x3, y3).

[STATEMENT] 43 *x1 y1 x2 y2 ... xn yn intensity_value*

This statement adds a polygon to the illumination source given a list of vertices and the intensity value. The line segments may intersect each other. Any enclosed region will be assigned the given intensity value.

[STATEMENT] 44 *x1 y1 x2 y2 ... xn yn intensity_value*

This statement adds a polygon to the illumination source given a list of vertices and the intensity value. The line segments may intersect each other. Any enclosed region will be assigned the given intensity value.

[STATEMENT] 45 *rotation_angle*

This statement rotates the source in the clockwise direction. Negative angles rotate the source in the counter clockwise direction.

[STATEMENT] 46 *[format_mode] ['output_file_name']*

This statement saves the source in a PDRAW or PLOTMTV format. The user can specify the output file name or use the default name 'source.map'.

FORMAT_MODE	Output
0 (default)	PLOTMTV format (3D view)
1	PDRAW format
2	PLOTMTV (contour format)

[STATEMENT] 47 *non-uniform_illumination_file*

This loads the source with intensity values stored in NON-UNIFORM_ILLUMINATION_FILE. This file should define a square matrix in the CONTOUR program data format. The number of rows should be odd so that center point will exactly fall on the center of a grid. Any non-zero intensity values defined using the primitives will be assigned the values contained in the NON-UNIFORM_ILLUMINATION_FILE.

[STATEMENT] 48

empty

[STATEMENT] 49 *open_field_intensity*

This statement renormalizes the open field intensity to OPEN_FIELD_INTENSITY. When any amount of OTA (see STATEMENT 50) is used the open field (clear mask) intensity will not be 1. This statement may be used for other cases in which renormalization is necessary.

[STATEMENT] 50 *0 a [b [c [d [e]]]]* (mode=0)

50 1 a [b] (mode=1 or 2)

This statement specifies the optical transmission with apopdization (OTA), or the amplitude of the pupil function. The MODE parameter follows the trial statement number.

MODE=0 sets the radially symmetric terms:

magnitude = $1 - [a + b\rho + c\rho^2 + d\rho^3 + e\rho^4]$

MODE=1 sets the first order non-symmetric terms

magnitude = 1 - [$a\rho^2\cos(\phi) + b\rho^2\cos(\phi)$]

MODE=2 sets the first order non-symmetric terms

magnitude = 1 - [$a\rho^2\cos(^2\phi) + b\rho^2\cos(2\phi)$]

6.0 Examples

The following examples demonstrate several of the different options available with SPLAT. The input file is presented first and is followed by the line-printer output that results from that input file. The DRAWPLOT or CONTOUR plot is also included. Recall that all characters on the input line are ignored except for numbers, quoted strings, semicolons, the ampersand (&) in the first column, and the pound(#) sign. Furthermore, the words "Statement" and "Trial" are optional.

6.1 Example 1 - Square and line

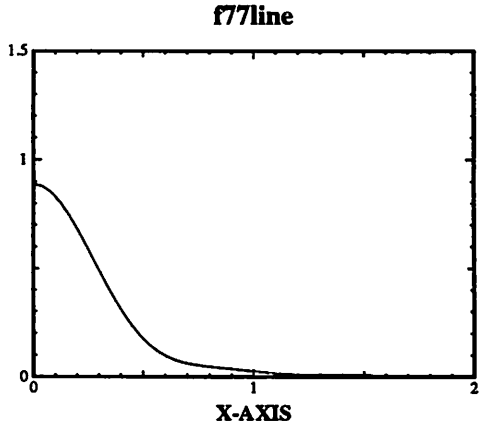
This example shows 0.4um square in close proximity to a 0.8um line with 1 um defocus.

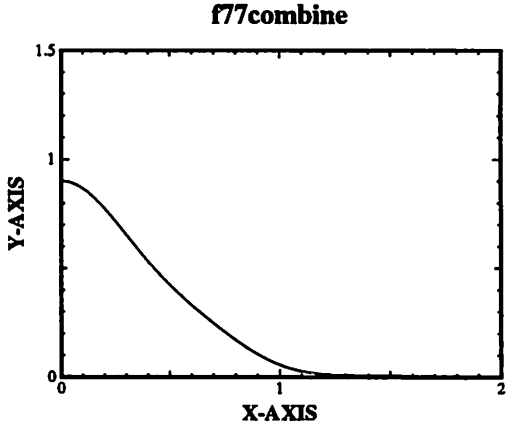
6.1.1 Input File

```
#
# Projection Lithography using SPLAT -- Example 1
#
Trial 1 : Printlevel 3                    ;# set print level
Trial 2 : lambda    = 0.5 um              ;# wavelength
Trial 3 : NA        = 0.5                 ;# numerical aperture
Trial 4 : Defocus   = 1.0 um              ;# defocus
Trial 5 : Sigma     = 0.7                 ;# coherence factor
Trial 6 : mask      = 2um x 2um
&         at 0 transmittance              ;# define the working area
Trial 7 : cutout = (0.0, 0.0)
&         0.4 x 2.0 at 1;
Trial 7 : cutout = (0.5, 0.0)
&         0.4 x 0.2 at 1                  ;# define the mask openings
Trial 10:                                 ;# calculate the Fourier coeffs
Trial 14: intensity (0,2) .. (2,2)
&         to 'f77line';
Trial 14: intensity (0,0) .. (2,0)
&         to 'f77combined'                ;# intensity profiles
Trial 0 : end;
```

6.1.2 Output File

```
2D optical imaging with aberrations apodization and modified illumi-
nation -- V4.2 3/23/93 -- DN/DL/KT
Trial 1: Print level = 3
Trial 2: Lambda      = 0.5000 microns
Trial 3: N.A.        = 0.5000
Trial 4: Defocus     = 1.0000 microns [ 1.0000 Rayleigh Units ]
Trial 5: Sigma       = 0.7000
Trial 5: Sigma In    = 0.0000
Trial 6: Field size = 2.0000 x 2.0000 @ 0.0000 scale =1.00
#     6: x size requires 12 harmonics.
#     6: y size requires 12 harmonics.
Trial 7: Cutout =( 0.0000, 0.0000)x( 0.4000, 2.0000) @ 1.0000 <0.0000>
Trial 7: Cutout =( 0.5000, 0.0000)x( 0.4000, 0.2000) @ 1.0000 <0.0000>
Trial 10: Calculate image Fourier Transform
#     10: Symmetry : T( f1, g1, f2, g2) =
#     10:             T(-f1,-g1,-f2,-g2)
#     10:             T( f1,-g1, f2,-g2)
#     10:             T(-f1, g1,-f2, g2)
#     10:       conjg[T( f2, g2, f1, g1)]
#     10: TCC computation time = 21.07000 sec.
#     10: Imaged with 12 by 12 harmonics
#     10: TCC calls: 1874 zeros: 1823
Trial 14: 3-decimal plot line : ( 0.000, 2.000)..( 2.000, 2.000)
          50 points saved in "f77line "
Trial 14: 3-decimal plot line : ( 0.000, 0.000)..( 2.000, 0.000)
          50 points saved in "f77combine"
Trial 0 : End of session
User Time (CPU) = 23.46000 sec, System Time = 0.38000 sec.
```





6.2 Example 2 - Calculate cross-coefficients

This example calculates the image for a checker-board pattern and stores the calculated transmission cross-coefficients in the file named "test2.tcc". A slight amount of coma is introduced.

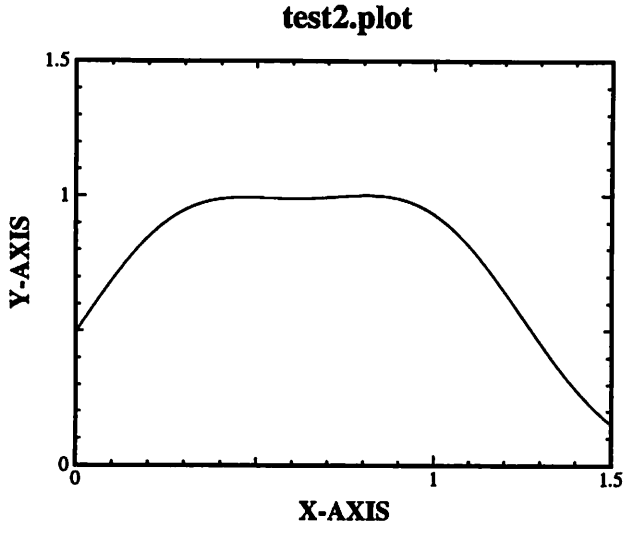
6.2.1 Input File

```
#
# 1.50 um. mask with 1.50 um. squares. -- Example 2
#
stmt 1: print = 3
stmt 2: lambda = 0.5 um
stmt 3: na = 0.5
stmt 4: defoc = 0.0 um
stmt 5: s = 0.7
stmt 6: mask = 1.5 x 1.5 @0
stmt 7: cutout = (0.00,0.00) 0.75 x 0.75 @1
stmt 7: cutout = (0.75,0.75) 0.75 x 0.75 @1
stmt 20: xpos = 1 ypos = 0
stmt 22: coma = 0.1
stmt 10: mode = 1 save in 'test2.tcc'
stmt 14: cutline from (-0.75,0.4) to (0.75,0.4)
& with 100 points in 'test2.plot'
end 0
```

6.2.2 Output File

```
2D optical imaging with aberrations apodization
and modified illumination -- V4.2 3/23/93 -- DN/DL/KT
Trial 1: Print level= 3
Trial 2: Lambda = 0.5000 microns
Trial 3: N.A. = 0.5000
Trial 4: Defocus = 0.0000 microns [ 0.0000 Rayleigh Units ]
Trial 5: Sigma = 0.7000
Trial 5: Sigma In = 0.0000
Trial 6: Field size = 1.5000 x 1.5000 @ 0.0000 scale =1.00
#     6: x size requires 10 harmonics.
#     6: y size requires 10 harmonics.
Trial 7: Cutout =( 0.0000, 0.0000)x( 0.7500, 0.7500) @ 1.0000 <0.0000>
Trial 7: Cutout =( 0.7500, 0.7500)x( 0.7500, 0.7500) @ 1.0000 <0.0000>
Trial 20: Relative x-coord (object) = 1.0000
Trial 20: Relative y-coord (object) = 0.0000
Trial 22: a031(coma) = 0.1000
Trial 22: b031(dist) = 0.0000
```

```
Trial 22: Maximum Coma OPD = 0.2000 Wavelengths
Trial 10: Calculate image Fourier Transform
Trial 10: Imaging System Parameters saved
# 10: Lambda = 0.5000
# 10: N.A. = 0.5000
# 10: Defocus = 0.0000
# 10: Sigma = 0.7000
# 10: Cone = 0.0000 0.0000
# 10: Cone = 0.0000 0.0000
# 10: Cone = 0.0000 0.0000
# 10: Cone = 0.0000 0.0000
# 10: Cone = 0.0000 0.0000
# 10: Cone = 0.0000 0.0000
# 10: Cone = 0.0000 0.0000
# 10: Cone = 0.0000 0.0000
# 10: Cone = 0.0000 0.0000
# 10: Cone = 0.0000 0.0000
Trial 10: Mask Characteristics saved
# 10: xlength = 1.5000
# 10: ylength = 1.5000
# 10: Transmittance = 0.0000
# 10: Object Coordinates : ( 1.0000 0.0000)
Trial 10: 1 Primary Lens Aberrations saved
# 10: Com = 0.1000 Dis = 0.0000
# 10: Symmetry : T( f1, g1, f2, g2) =
# 10: conjg[T(-f1,-g1,-f2,-g2)]
# 10: T( f1,-g1, f2,-g2)
# 10: conjg[T(-f1, g1,-f2, g2)]
# 10: conjg[T( f2, g2, f1, g1)]
# 10: 4905 / 14641 TCCs saved in "test2.tcc "
# 10: TCC computation time = 8.63000 sec.
# 10: Imaged with 10 by 10 harmonics
# 10: TCC calls: 662 zeros: 1259
Trial 14: 3-decimal plot line : ( -0.750, 0.400)..( 0.750, 0.400)
100 points saved in "test2.plot"
Program execution terminated.
User Time (CPU) = 10.82000 sec, System Time = 0.75000 sec.
```



6.3 Example 3 - Intensity profiles of isolated square

Reading the previously computed TCC's from the file created in Example 2, this example calculates the intensity profiles for a different pattern, an isolated square. Note that the image parameters (lambda, na, sigma) should not be changed. Also, the mask size must remain constant. If any of these parameters are changed, the results obtained with STATEMENT 19 will not be valid.

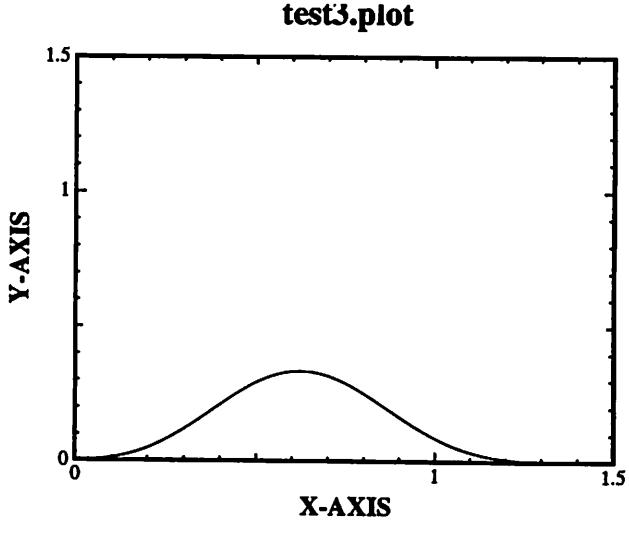
6.3.1 Input File

```
#
# An isolated 0.5 lambda/NA square. -- Example 3
#
stmt 1: print = 3
stmt 6: mask = 1.5 x 1.5 @0
stmt 7: cutout = (0.00,0.00) 0.25 x 0.25 @1
stmt 19: read from 'test2.tcc'
stmt 14: cutline from (-0.75,0.0) to (0.75,0.0)
&     with 100 points in 'test3.plot'
end 0
```

6.3.2 Output File

```
2D optical imaging with aberrations
apodization and modified illumination -- V4.2
                                3/23/93 -- DN/DL/KT
Trial 1: Print level= 3
```

```
Trial 6: Field size = 1.5000 x 1.5000 @ 0.0000 scale =1.00
#      6: x size requires 6 harmonics.
#      6: y size requires 6 harmonics.
Trial 7: Cutout =( 0.0000, 0.0000)x( 0.2500, 0.2500) @ 1.0000 <0.0000>
Trial 19: Imaging System Parameters Loaded
#      19: Lambda = 0.5000
#      19: N.A. = 0.5000
#      19: Defocus = 0.0000
#      19: Sigma = 0.7000
Trial 19: Mask Characteristics Loaded
#      19: xlength = 1.5000
#      19: ylength = 1.5000
#      19: Transmittance = 0.0000
#      19: Object Coordinates : ( 1.0000 0.0000)
Trial 19: 1 Primary Lens Aberrations Loaded
#      19: Com = 0.1000 Dis = 0.0000
Trial 19: 4905 TCCs read in from "test2.tcc ".
#      19: Imaged with 5 by 5 harmonics
Trial 19: Calculate image Fourier Transform
Trial 14: 3-decimal plot line : ( -0.750, 0.000)..( 0.750, 0.000)
      100 points saved in "test3.plot"
Program execution terminated.
User Time (CPU) = 2.18000 sec, System Time = 0.30000 sec.
```



6.4 Example 4 - Statements 9 and 11

This example is a demonstration of Statements 9 and 11. Statement 9 shows what the mask looks like after a discrete Fourier transformation followed by an inverse Fourier transformation.

6.4.1 Input File

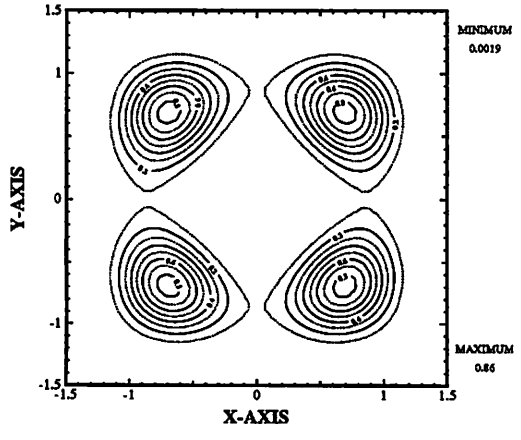
```
#
# Triangular pattern -- Example 4
#
stmt 1: print = 3
stmt 2: lambda = 0.5 um
stmt 3: na = 0.5
stmt 4: defoc = 0.0 um
stmt 5: s = 0.3 stmt 6: mask = 1.5 x 1.5 @0
stmt 8: cutout = (1.00,1.00) -1.00 x -1.00 @1
stmt 9: 20 20 mode = 1 box : (-1.5,-1.5) 3 x 3;
stmt 10:
stmt 11: -1.5 -1.5 3.0 3.0 'test4.ctr'
stmt 14: cutline from (-1.5,-1.5) to (1.5,1.5)
&        with 100 points in 'test4.plot'
end 0
```

6.4.2 Output File

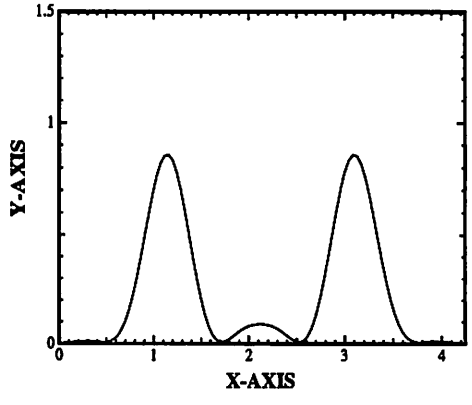
```
2D optical imaging with
aberrations apodization and modified illumination -- V4.2
                                        3/23/93 -- DN/DL/KT

Trial 1: Print level = 3
Trial 2: Lambda      = 0.5000 microns
Trial 3: N.A.        = 0.5000
Trial 4: Defocus     = 0.0000 microns [ 0.0000 Rayleigh Units ]
Trial 5: Sigma       = 0.3000
Trial 5: Sigma In    = 0.0000
Trial 6: Field size  = 1.5000 x 1.5000 @ 0.0000 scale =1.00
#     6: x size requires 6 harmonics.
#     6: y size requires 6 harmonics.
#     8: Add triangle ( 1.000, 1.000)x(-1.000,-1.000) @ 1.000 <0.000>
Trial 8: Triangle = ( 0.000, 1.000) ( 1.000, 1.000) ( 1.000, 0.000)
Trial 9: Transmittance : 20x20 pts ( -1.500, -1.500)x( 1.500, 1.500)
         Contour map of the mask transmittance
         Left bottom corner : ( -1.500 -1.500)
         Right top corner : ( 1.500 1.500)
         x-interval : 0.158 microns
         y-interval : 0.158 microns
         00000000000000000000
```

```
                0000000000000000000
                0000000000000000000
                0000             0000
                000 111110011111 000
                000 1111 00 1111 000
                000 111 0000 111 000
                000 11 000000 11 000
                000 1 00000000 1 000
                000 000000000000 000
                000 000000000000 000
                000 1 00000000 1 000
                000 11 000000 11 000
                000 111 0000 111 000
                000 1111 00 1111 000
                000 111110011111 000
                0000             0000
                0000000000000000000
                0000000000000000000
                0000000000000000000
Trial 10: Calculate image Fourier Transform
#     10: Symmetry : T( f1, g1, f2, g2) =
#     10:             T(-f1,-g1,-f2,-g2)
#     10:             T( f1,-g1, f2,-g2)
#     10:             T(-f1, g1,-f2, g2)
#     10:             T( f2, g2, f1, g1)
#     10: TCC computation time = 0.18001 sec.
#     10: Imaged with 6 by 6 harmonics
#     10: TCC calls: 263 zeros: 74
Trial 11: Image intensity contour data stored in "test4.ctr "
Trial 14: 3-decimal plot line : ( -1.500, -1.500)..( 1.500, 1.500)
          100 points saved in "test4.plot"
Program execution terminated.
User Time (CPU) = 138.82001 sec, System Time = 0.76000 sec.
```

test4.ctr



test4.plot



6.5 Example 5 - Phase-shifter outriggers

This is a 0.6 λ/NA x 0.6 λ/NA square contact with 0.2 λ/NA phase-shifter outriggers.

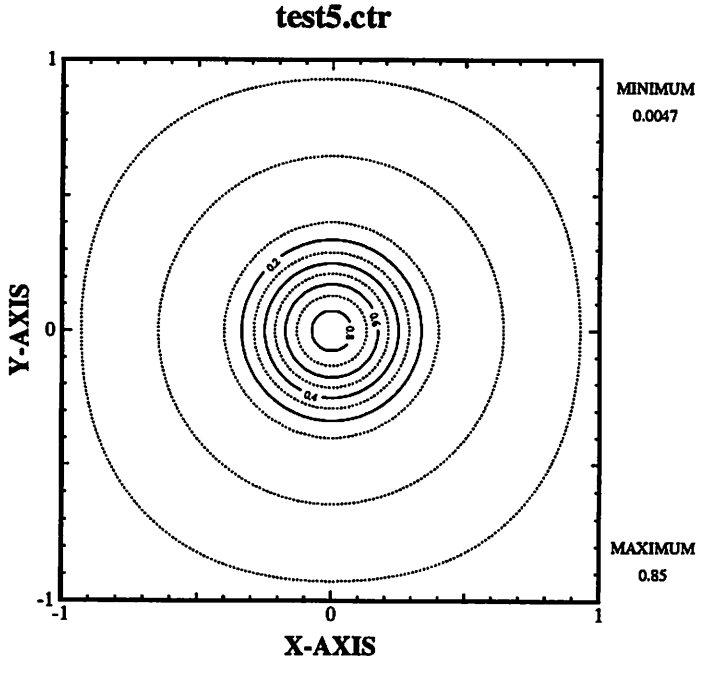
6.5.1 Input File

```
#
# Projection Lithography using SPLAT -- Example 5
#
1 : Printlevel 3                              ;# set print level
2 : lambda = 0.5 um                           ;# wavelength
3 : NA = 0.5                                  ;# numerical aperture
4 : Defocus= 0.0 um                           ;# defocus
5 : Sigma = 0.5                               ;# coherence factor
6 : mask = 2 um x 2 um
& at 0 transmittance                          ;# define the working area
7 : cutout=(0.0, 0.0) 0.3 x 0.3 at 1 (0)      ;# define the contact
7 : cutout=(0.6, 0.0) 0.2 x 0.6 at 1 (180);# define vertical outrigger
7 : cutout=(0.0, 0.6) 0.6 x 0.2 at 1 (180);# define horizontal outrigger
10:                                           ;# calculate the Fourier coeffs
11: (-1,-1) 2 x 2 to 'test5.ctr'              ;# contour plot
0 : end;
```

6.5.2 Output File

```
2D optical imaging with aberrations
apodization and modified illumination -- V4.2
                                3/23/93 -- DN/DL/KT
Trial 1: Print level = 3
Trial 2: Lambda      = 0.5000 microns
Trial 3: N.A.        = 0.5000
Trial 4: Defocus     = 0.0000 microns [ 0.0000 Rayleigh Units ]
Trial 5: Sigma       = 0.5000
Trial 5: Sigma In    = 0.0000
Trial 6: Field size  = 2.0000 x 2.0000 @ 0.0000 scale =1.00
#     6: x size requires 12 harmonics.
#     6: y size requires 12 harmonics.
Trial 7: Cutout =( 0.0000, 0.0000)x( 0.3000, 0.3000) @ 1.0000 <0.0000>
Trial 7: Cutout =( 0.6000, 0.0000)x( 0.2000, 0.6000) @ 1.0000 <180.0000>
Trial 7: Cutout =( 0.0000, 0.6000)x( 0.6000, 0.2000) @ 1.0000 <180.0000>
Trial 10: Calculate image Fourier Transform
#     10: Symmetry : T( f1, g1, f2, g2) =
#     10:            T(-f1,-g1,-f2,-g2)
#     10:            T( f1,-g1, f2,-g2)
```

```
#     10:              T(-f1, g1,-f2, g2)
#     10:              T( f2, g2, f1, g1)
#     10: TCC computation time = 1.79000 sec.
#     10: Imaged with 12 by 12 harmonics
#     10: TCC calls: 1313 zeros: 2384
Trial 11: Image intensity contour data stored in "test5.ctr "
Trial 0 : End of session
User Time (CPU) = 24.99000 sec, System Time = 0.28000 sec.
```



6.6 Example 6 - Rectangular mask with border

This example simulates a hole pattern consisting of central rectangular, surrounded by four bars w=0.3, d=0.05, l=0.6. The input file results in negative intensity values when run on certain machines, such as the DEC 3100. This could be a compiler error.

6.6.1 Input File

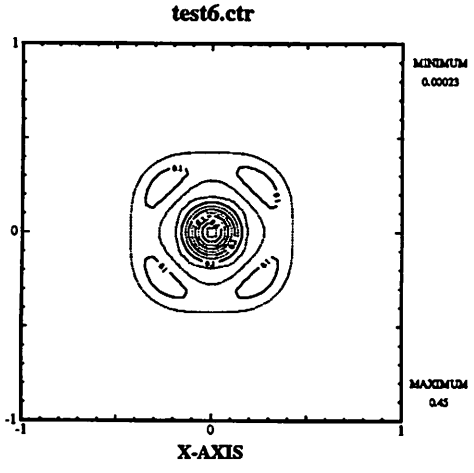
```
#
# Example 6 - Phase Mask
#
1 printlevel = 2;
2 wavelength = 0.2484;
```

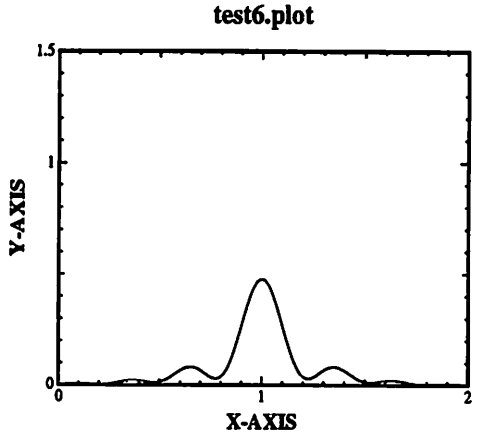
```
3 NA = 0.5;
5 sigma = 0.3;
6 mask = 2 x 2 at 0 transmittance;
7 (0.0,0.0) 0.15 x 0.15 at 1 transmittance;
7 (.2, 0) 0.1 x 0.3 at 1 transmittance 180 degree phase shift;
7 (0,0.2) 0.2 x 0.1 at 1 transmittance 180 degree phase shift;
#
4 Defocus = 0.5;
10 ;calculate fourier coefficients and TCC's
11 (-1,-1) 2 x 2 to 'test6.ctr';
14 (-1,0) ...(1,0) 500 to 'test6.plot';
0 End;
```

6.6.2 Output File

```
2D optical imaging with aberrations
apodization and modified illumination -- V4.2
                                    3/23/93 -- DN/DL/KT
 Trial 1: Print level= 2
2 wavelength = 0.2484;
 Trial 2: Lambda       = 0.2484 microns
3 NA = 0.5;
 Trial 3: N.A.         = 0.5000
5 sigma                = 0.3;
 Trial 5: Sigma        = 0.3000
 Trial 5: Sigma In     = 0.0000
6 mask = 2 x 2 at 0 transmittance;
 Trial 6: Field size = 2.0000 x 2.0000 @ 0.0000 scale =1.00
7 (0.0,0.0) 0.15 x 0.15 at 1 transmittance;
 Trial 7: Cutout =( 0.0000, 0.0000)x( 0.1500, 0.1500) @ 1.0000 < 0.0000>
7 (.2, 0) 0.1 x 0.3 at 1 transmittance 180 degree phase shift;
 Trial 7: Cutout =( 0.2000, 0.0000)x( 0.1000, 0.3000) @ 1.0000 <180.0000>
7 (0,0.2) 0.2 x 0.1 at 1 transmittance 180 degree phase shift;
 Trial 7: Cutout =( 0.0000, 0.2000)x( 0.2000, 0.1000) @ 1.0000 <180.0000>
#
4 Defocus = 0.5;
 Trial 4: Defocus = 0.5000 microns [ 1.0064 Rayleigh Units ]
10 ;calculate fourier coefficients and TCC's
 Trial 10: Calculate image Fourier Transform
#       10: Symmetry : T( f1, g1, f2, g2) =
#       10:              T(-f1,-g1,-f2,-g2)
#       10:              T( f1,-g1, f2,-g2)
#       10:              T(-f1, g1,-f2, g2)
#       10:        conjg[T( f2, g2, f1, g1)]
```

```
#       10: TCC computation time = 144.89999 sec.
11 (-1,-1) 2 x 2 to 'test6.ctr';
 Trial 11: Image intensity contour data stored in "test6.ctr "
14 (-1,0) ...(1,0) 500 to 'test6.plot';
 Trial 14: 3-decimal plot line : ( -1.000, 0.000)..( 1.000, 0.000)
500 points saved in "test6.plot"

0 End;
 Trial 0 : End of session
User Time (CPU) = 211.30000 sec, System Time = 0.82000 sec.
```





6.7 Example 7 - Statements 27 and 28

This input file demonstrates the use of STATEMENTs 27 and 28.This example was made by Kenny Toh to illustrate the use of triangular shapes.

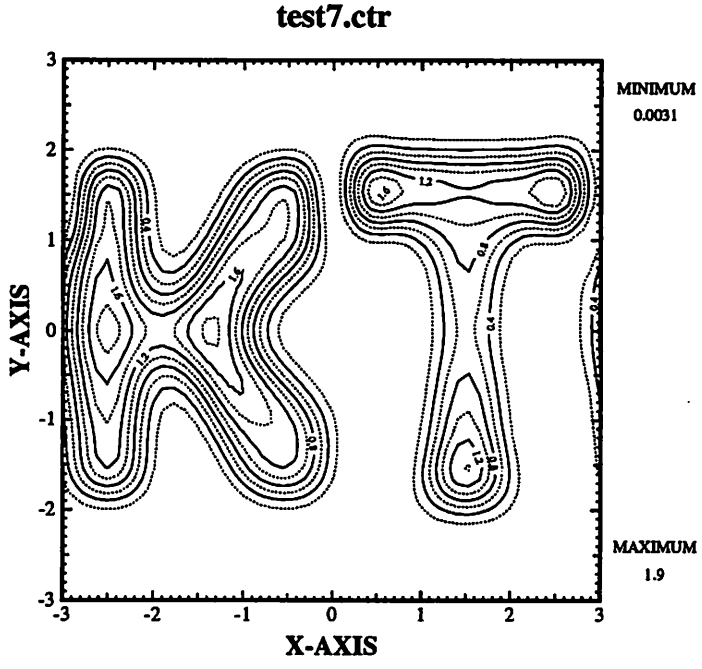
6.7.1 Input File

```
#
# Test 7
#
1:3;
2:0.5;              # wavelength
3:0.5;              # numerical aperture
4:0;                # defocus
5:0.5;              # coherence
6:3 3 0;            # mask size (1st quadrant)
27:(0.0 1.0) x (3.0 1.0) @1 180;          # Upper T
```

```
27:(1.0 1.0) x (1.0 -3.0) @1 180;          # Lower T
27:(-3.0 -2.0) x (3.0 4.0) @1 0;           # Set up the K
28:(-2.0 2.0) x (1.0 -2.0) @-1 0;          # The upper K
28:(-2.0 -2.0) x (1.0 2.0) @-1 0;          # The lower K
28:(-1.0 0.0) (0.0 1.0) (0.0 -1.0) @-1 0; # The valley of the K
10;                                        # calculate...
11:-3 -3 6 6 "test7.ctr";                  # contour plot
0;
```

6.7.2 Output File

```
2D optical imaging with aberrations
apodization and modified illumination -- V4.2
                                3/23/93 -- DN/DL/KT
Trial 1: Print level = 3
Trial 2: Lambda      = 0.5000 microns
Trial 3: N.A.        = 0.5000
Trial 4: Defocus     = 0.0000 microns [ 0.0000 Rayleigh Units ]
Trial 5: Sigma       = 0.5000
Trial 5: Sigma In    = 0.0000
Trial 6: Field size  = 3.0000 x 3.0000 @ 0.0000 scale =1.00
#      6: x size requires 18 harmonics.
#      6: y size requires 18 harmonics.
Trial 27: Cutout =( 0.0000, 1.0000)x( 3.0000, 1.0000) @ 1.0000 <180.0000>
Trial 27: Cutout =( 1.0000,-2.0000)x( 1.0000, 3.0000) @ 1.0000 <180.0000>
Trial 27: Cutout =(-3.0000,-2.0000)x( 3.0000, 4.0000) @ 1.0000 <0.0000>
#      28: Add triangle (-2.000, 2.000)x( 1.000,-2.000) @-1.000 <0.000>
Trial 28: Triangle = (-2.000, 2.000) (-2.000, 0.000) (-1.000, 2.000)
#      28: Add triangle (-2.000,-2.000)x( 1.000, 2.000) @-1.000 <0.000>
Trial 28: Triangle = (-2.000,-2.000) (-2.000, 0.000) (-1.000,-2.000)
#      28: Add triangle ( 0.000, 0.000)x(-1.000,-1.000) @-1.000 <0.000>
#      28: Add triangle ( 0.000, 0.000)x(-1.000, 1.000) @-1.000 <0.000>
Trial 28: Triangle = (-1.000, 0.000) ( 0.000, 1.000) ( 0.000,-1.000)
Trial 10: Calculate image Fourier Transform
#      10: Symmetry : T( f1, g1, f2, g2) =
#      10:              T(-f1,-g1,-f2,-g2)
#      10:              T( f1,-g1, f2,-g2)
#      10:              T(-f1, g1,-f2, g2)
#      10:              T( f2, g2, f1, g1)
#      10: TCC computation time = 8.25000 sec.
#      10: Imaged with 18 by 18 harmonics
#      10: TCC calls: 6441 zeros: 10120
Trial 11: Image intensity contour data stored in "test7.ctr"
Trial 0 : End of session
User Time (CPU) = 64.17000 sec, System Time = 0.37000 sec.
```



6.8 Example 8 - Circular contours

This is another case that at one time caused errors in SPLAT. If the contours are not circular, then the calculation is wrong!

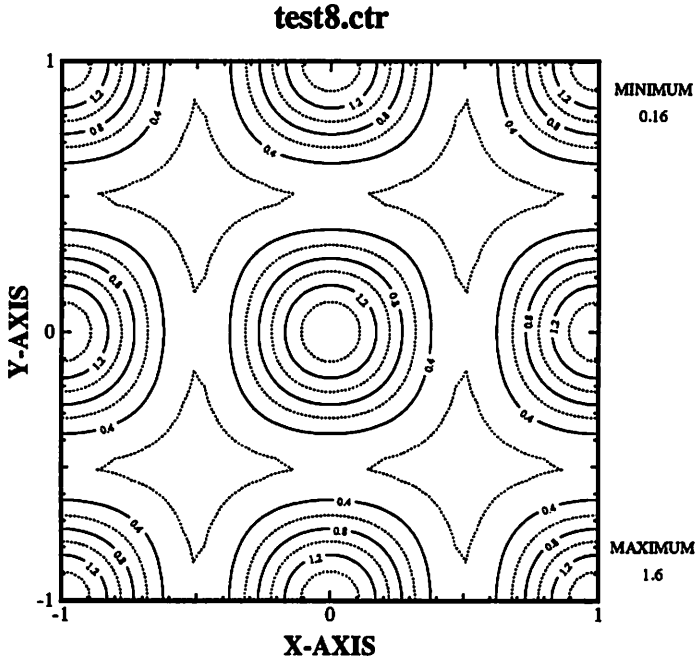
6.8.1 Input File

```
#
# Test 8
#
1:3;
2:0.5;
3:0.5;
4:1;
5:0.5;
6:1 1 0;
7:0.0 0.0 0.5 0.5 1 0;
7:0.5 0.5 0.5 0.5 1 0;
7:0.0 0.5 0.5 0.5 1 180;
7:0.5 0.0 0.5 0.5 1 180;
10:;
11:-1 -1 2 2 "test8.ctr";
```

```
0;
```

6.8.2 Output File

```
2D optical imaging with aberrations
apodization and modified illumination -- V4.2
                                3/23/93 -- DN/DL/KT
Trial 1: Print level = 3
Trial 2: Lambda      = 0.5000 microns
Trial 3: N.A.        = 0.5000
Trial 4: Defocus     = 1.0000 microns [ 1.0000 Rayleigh Units ]
Trial 5: Sigma       = 0.5000
Trial 5: Sigma In    = 0.0000
Trial 6: Field size  = 1.0000 x 1.0000 @ 0.0000 scale =1.00
#     6: x size requires 6 harmonics.
#     6: y size requires 6 harmonics.
Trial 7: Cutout =( 0.0000, 0.0000)x( 0.5000, 0.5000) @ 1.0000 <0.0000>
Trial 7: Cutout =( 0.5000, 0.5000)x( 0.5000, 0.5000) @ 1.0000 <0.0000>
Trial 7: Cutout =( 0.0000, 0.5000)x( 0.5000, 0.5000) @ 1.0000 <180.0000>
Trial 7: Cutout =( 0.5000, 0.0000)x( 0.5000, 0.5000) @ 1.0000 <180.0000>
Trial 10: Calculate image Fourier Transform
#     10: Symmetry : T( f1, g1, f2, g2) =
#     10:             T(-f1,-g1,-f2,-g2)
#     10:             T( f1,-g1, f2,-g2)
#     10:             T(-f1, g1,-f2, g2)
#     10:       conjg[T( f2, g2, f1, g1)]
#     10: TCC computation time = 1.02000 sec.
#     10: Imaged with 6 by 6 harmonics
#     10: TCC calls: 80 zeros: 257
Trial 11: Image intensity contour data stored in "test8.ctr"
Trial 0 : End of session
User Time (CPU) = 10.56000 sec, System Time = 0.27000 sec.
```



6.9 Example 9 - Apodization

This is a simulation of a 0.6μm x 0.6 μm contact hole. In lens filtering is used to increase the depth of focus.

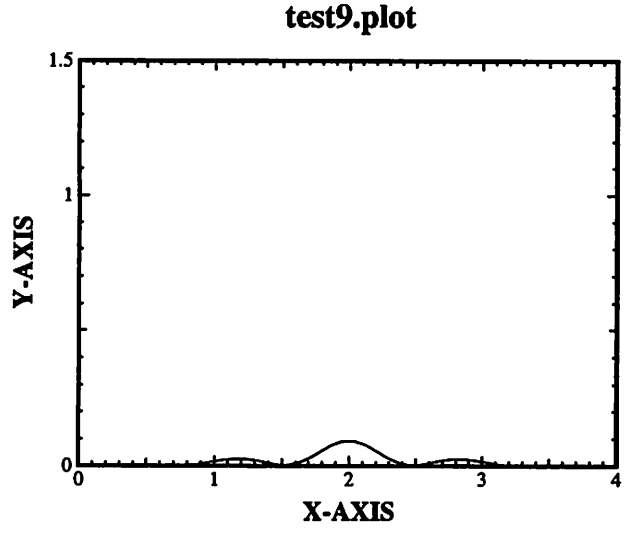
6.9.1 Input File

```
#
# Projection Lithography using SPLAT -- Test9
#
Statement 1 : Printlevel 3              ;# set print level
Statement 2 : lambda = 0.5 um           ;# wavelength
Statement 3 : NA = 0.5                  ;# numerical aperture
Statement 4 : Defocus= 0.0 um           ;# defocus
Statement 5 : Sigma = 0.5               ;# coherence factor
Statement 6 : mask = 2um x 2um
&             at 0 transmittance        ;# define working area
Statement 7 : cutout = (0.0, 0.0)
&             0.3 x 0.3 at 1            ;# define mask openings
Statement 29: Filter 1.0
&             with parameters 0.65 4.54;# define in-lens filter
```

```
Statement 10: 0 2                          ;# calculate the Fourier coeffs
Statement 14: intensity (-2,0) .. (2,0)
&             to 'test9.plot'              ;#2-D intensity profiles
Statement 0 : end;
```

6.9.2 Output File

```
2D optical imaging with aberrations
apodization and modified illumination -- V4.2
                                    3/23/93 -- DN/DL/KT
Trial 1: Print level= 3
Trial 2: Lambda       = 0.5000 microns
Trial 3: N.A.         = 0.5000
Trial 4: Defocus      = 0.0000 microns [ 0.0000 Rayleigh Units ]
Trial 5: Sigma        = 0.5000
Trial 5: Sigma In     = 0.0000
Trial 6: Field size   = 2.0000 x 2.0000 @ 0.0000 scale =1.00
#      6: x size requires 12 harmonics.
#      6: y size requires 12 harmonics.
Trial 7: Cutout =( 0.0000, 0.0000)x( 0.3000, 0.3000) @ 1.0000 <0.0000>
Trial 29: Spatial Filter of cos(2*pi* 0.65 *r**2 - 4.54/2)
Trial 10: Calculate image Fourier Transform
#      10: Force=2 -- 2D Numerical Integration
#      10: Symmetry : T( f1, g1, f2, g2) =
#      10:            T(-f1,-g1,-f2,-g2)
#      10:            T( f1,-g1, f2,-g2)
#      10:            T(-f1, g1,-f2, g2)
#      10:            T( f2, g2, f1, g1)
#      10: TCC computation time = 25.79000 sec.
#      10: Imaged with 12 by 12 harmonics
#      10: TCC calls: 1264 zeros: 2433
Trial 14: 3-decimal plot line : ( -2.000, 0.000)..( 2.000, 0.000)
          50 points saved in "test9.plot"
Trial 0 : End of session
User Time (CPU) = 27.30000 sec, System Time = 0.35000 sec.
```



6.10 Example 10 - Annular source

This example models 0.6 μm elbows with an annular source.

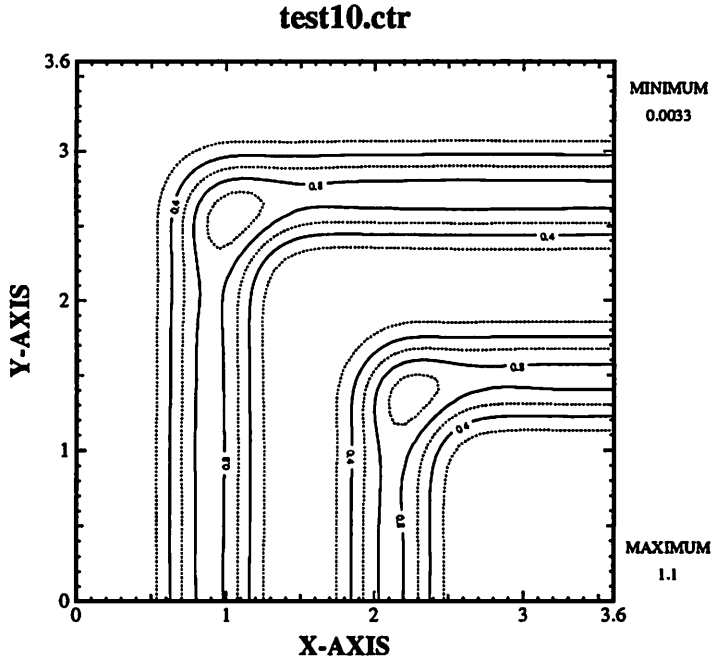
6.10.1 Input File

```
#
# Projection Lithography using SPLAT -- Test 10
#
Statement 1 : Printlevel 3;              # set print level
Statement 2 : lambda = 0.5 um;           # wavelength
Statement 3 : NA = 0.5;                  # numerical aperture
Statement 4 : Defocus= 0.0 um;           # defocus
Statement 5 : Sigma = 0.5 Sigmain = 0.3; # set annular illuminator
Statement 6 : mask = 3.6um 3.6um
&             at 0 transmittance;
Statement 7 : cutout = (0.6 0.0)
&             0.6 x 2.4 at 1;
Statement 7 : cutout = (0.6 2.4)
&             3.0 x 0.6 at 1;
Statement 7 : cutout = (1.8 0.0)
&             0.6 x 1.2 at 1;
Statement 7 : cutout = (1.8 1.2)
&             1.8 x 0.6 at 1;            # define mask openings
Statement 10 : 0 2;                      # calculate Fourier coeffs
Statement 11 : (0 0) 3.6 3.6 "test10.ctr"; # contour
```

```
Statement 0 : End;
```

6.10.2 Output File

```
2D optical imaging with aberrations
apodization and modified illumination -- V4.2
                                3/23/93 -- DN/DL/KT
Trial 1: Print level     = 3
Trial 2: Lambda          = 0.5000 microns
Trial 3: N.A.            = 0.5000
Trial 4: Defocus         = 0.0000 microns [ 0.0000 Rayleigh Units ]
Trial 5: Sigma           = 0.5000
Trial 5: Sigma In        = 0.3000
Trial 6: Field size = 3.6000 x 3.6000 @ 0.0000 scale =1.00
#      6: x size requires 20 harmonics.
#      6: y size requires 20 harmonics.
Trial 7: Cutout =( 0.6000, 0.0000)x( 0.6000, 2.4000) @ 1.0000 <0.0000>
Trial 7: Cutout =( 0.6000, 2.4000)x( 3.0000, 0.6000) @ 1.0000 <0.0000>
Trial 7: Cutout =( 1.8000, 0.0000)x( 0.6000, 1.2000) @ 1.0000 <0.0000>
Trial 7: Cutout =( 1.8000, 1.2000)x( 1.8000, 0.6000) @ 1.0000 <0.0000>
Trial 10: Calculate image Fourier Transform
#      10: Force=2 -- 2D Numerical Integration
#      10: Symmetry : T( f1, g1, f2, g2) =
#      10:              T(-f1,-g1,-f2,-g2)
#      10:              T( f1,-g1, f2,-g2)
#      10:              T(-f1, g1,-f2, g2)
#      10:              T( f2, g2, f1, g1)
#      10: TCC computation time = 192.34000 sec.
#      10: Imaged with 20 by 20 harmonics
#      10: TCC calls: 22259 zeros: 27023
Trial 11: Image intensity contour data stored in "test10.ctr"
Trial 0 : End of session
User Time (CPU) = 247.53999 sec, System Time = 0.77000 sec.
```



6.11 Example 11 - Offaxis illumination

This example show 0.6um elbows with offaxis illumination.

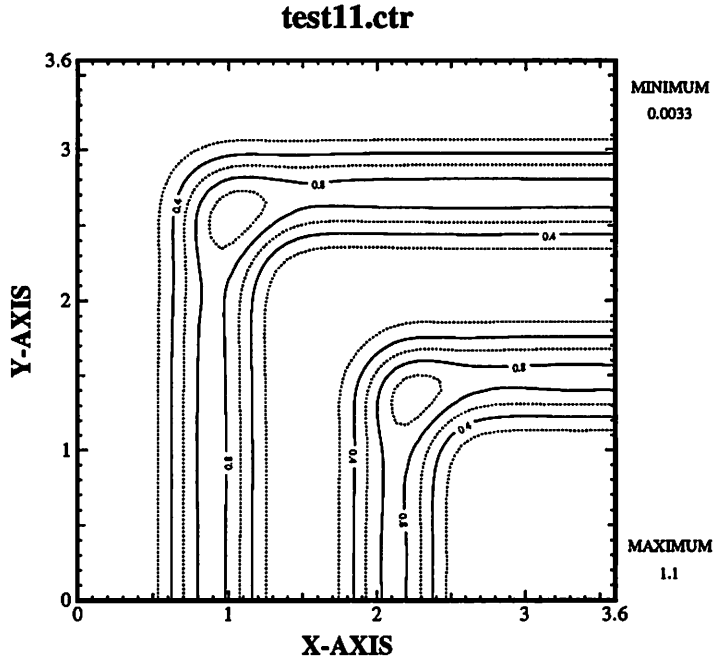
6.11.1 Input File

```
#
# Projection Lithography using SPLAT - Test 11
#
Statement 1 : Printlevel 3              ; # set print level
Statement 2 : lambda = 0.5 um           ; # wavelength
Statement 3 : NA = 0.5;                  # numerical aperture
Statement 4 : Defocus= 0.0 um;           # defocus
Statement 5 : Sigma = 0.5                # set partial coherence
Statement 30: Spot_radius = 0.2
&             at (0.0,0.4);              # use off-axis source
Statement 6 : mask = 3.6um 3.6um
&             at 0 transmittance;
Statement 7 : cutout = (0.6 0.0)
&             0.6 x 2.4 at 1;
Statement 7 : cutout = (0.6 2.4)
```

```
&              3.0 x 0.6 at 1;
Statement 7 : cutout = (1.8 0.0)
&              0.6 x 1.2 at 1;
Statement 7 : cutout = (1.8 1.2)
&              1.8 x 0.6 at 1;          # define mask openings
Statement 10 : 0 2;                     # calculate Fourier coeffs
Statement 11 : (0 0) 3.6 3.6 "test11.ctr"; # contour
Statement 0 : End;
```

6.11.2 Output File

```
2D optical imaging with aberrations
apodization and modified illumination -- V4.2
                                3/23/93 -- DN/DL/KT
Trial 1: Print level        = 3
Trial 2: Lambda             = 0.5000 microns
Trial 3: N.A.             = 0.5000
Trial 4: Defocus          = 0.0000 microns [ 0.0000 Rayleigh Units ]
Trial 5: Sigma            = 0.5000
Trial 5: Sigma In         = 0.0000
Trial 30: Spot Radius     = 0.2000
Trial 30: Coordinates of Spot = 0.0000 0.4000
Trial 6: Field size = 3.6000 x 3.6000 @ 0.0000 scale =1.00
#      6: x size requires 20 harmonics.
#      6: y size requires 20 harmonics.
Trial 7: Cutout =( 0.6000, 0.0000)x( 0.6000, 2.4000) @ 1.0000 <0.0000>
Trial 7: Cutout =( 0.6000, 2.4000)x( 3.0000, 0.6000) @ 1.0000 <0.0000>
Trial 7: Cutout =( 1.8000, 0.0000)x( 0.6000, 1.2000) @ 1.0000 <0.0000>
Trial 7: Cutout =( 1.8000, 1.2000)x( 1.8000, 0.6000) @ 1.0000 <0.0000>
Trial 10: Calculate image Fourier Transform
#      10: Force=2 -- 2D Numerical Integration
#      10: Symmetry : T( f1, g1, f2, g2) =
#      10: T(-f1, g1,-f2, g2)
#      10: T( f2, g2, f1, g1)
#      10: TCC computation time = 132.28000 sec.
#      10: Imaged with 20 by 20 harmonics
#      10: TCC calls: 12826 zeros: 36335
Trial 11: Image intensity contour data stored in "test11.ctr"
Trial 0 : End of session
User Time (CPU) = 187.71001 sec, System Time = 0.53000 sec.
```



6.12 Example 12 - Quadropole illuminator

Here are 0.6um elbows again. This time a quadrupole illuminator is used.

6.12.1 Input File

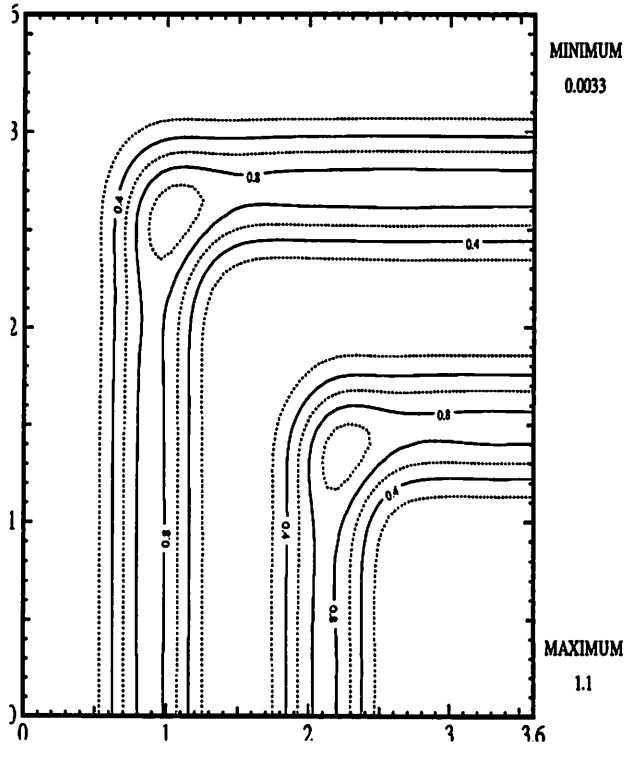
```
#
# Projection Lithography using SPLAT - Test 12
#
Statement 1 : Printlevel 3;              # set print level
Statement 2 : lambda = 0.5 um;           # wavelength
Statement 3 : NA = 0.5;                  # numerical aperture
Statement 4 : Defocus= 0.0 um;           # defocus
Statement 30: Spot_radius = 0.2
&             at ( 0.35, 0.35);
Statement 30: Spot_radius = 0.2
&             at (-0.35, 0.35);
Statement 30: Spot_radius = 0.2
&             at ( 0.35,-0.35);
Statement 30: Spot_radius = 0.2
&             at (-0.35,-0.35);          # set quadrupole illuminator
Statement 6 : mask = 3.6um 3.6um
```

```
&             at 0 transmittance;
Statement 7 : cutout = (0.6 0.0)
&             0.6 x 2.4 at 1;
Statement 7 : cutout = (0.6 2.4)
&             3.0 x 0.6 at 1;
Statement 7 : cutout = (1.8 0.0)
&             0.6 x 1.2 at 1;
Statement 7 : cutout = (1.8 1.2)
&             1.8 x 0.6 at 1;          # define mask openings
Statement 10 : 0 2;                    # calculate Fourier coeffs
Statement 11 : (0 0) 3.6 3.6 "test12.ctr"; # contour
Statement 0 : End;
```

6.12.2 Output File

```
2D optical imaging with aberrations
apodization and modified illumination -- V4.2
                              3/23/93 -- DN/DL/KT

 Trial  1:  Print level=  3
 Trial  2:  Lambda     =  0.5000 microns
 Trial  3:  N.A.       =  0.5000
 Trial  4:  Defocus    =  0.0000 microns   [ 0.0000 Rayleigh Units ]
 Trial 30:  Spot Radius            =  0.2000
 Trial 30:  Coordinates of Spot    =  0.3500 0.3500
 Trial 30:  Spot Radius            =  0.2000
 Trial 30:  Coordinates of Spot    = -0.3500 0.3500
 Trial 30:  Spot Radius            =  0.2000
 Trial 30:  Coordinates of Spot    =  0.3500-0.3500
 Trial 30:  Spot Radius            =  0.2000
 Trial 30:  Coordinates of Spot    = -0.3500-0.3500
 Trial  6:  Mask size  = 3.6000 x 3.6000 @ 0.0000 layout scale =1.00 (um/div)
 #      6:  x size requires  24 harmonics.
 #      6:  y size requires  24 harmonics.
 Trial  7:  Mask Cutout =(0.6000, 0.0000)x(0.6000, 2.4000) @ 1.0000 <0.0000>
 Trial  7:  Mask Cutout =(0.6000, 2.4000)x(3.0000, 0.6000) @ 1.0000 <0.0000>
 Trial  7:  Mask Cutout =(1.8000, 0.0000)x(0.6000, 1.2000) @ 1.0000 <0.0000>
 Trial  7:  Mask Cutout =(1.8000, 1.2000)x(1.8000, 0.6000) @ 1.0000 <0.0000>
 Trial 10:  Calculate image Fourier Transform
 #     10:     Force=2 -- 2D Numerical Integration
 #     10:  Setting System Partial Coherence = 0.69497
 #     10:  Symmetry :  T( f1, g1, f2, g2) =
 #     10:              T(-f1,-g1,-f2,-g2)
 #     10:              T( f1,-g1, f2,-g2)
```

```
#     10:              T(-f1, g1,-f2, g2)
#     10:              T( f2, g2, f1, g1)
#     10:  TCC computation time  = 408.37000 sec.
#     10:  Imaged with  24 by  24 harmonics
#     10:  TCC calls:   26820  zeros:  170368
Trial 11:  Image intensity contour data stored in "test12.ctr"
Trial 0 :  End of session
User Time (CPU) =  506.85999 sec,  System Time =   1.21000 sec.
```



6.13 Example 13 - Magnification and Obliquity Factors

The following example demonstrates the use of magnification with STATEMENT 31. The input file was sent to SPLAT two times, changing STATEMENT 31 from

stmt 31: magnification = 1.0 ;

to

stmt 31: magnification = 5.0 ;

Although only the input and output files for the first trial are shown here, the output plots have been overlaid to shown the effects of magnification.

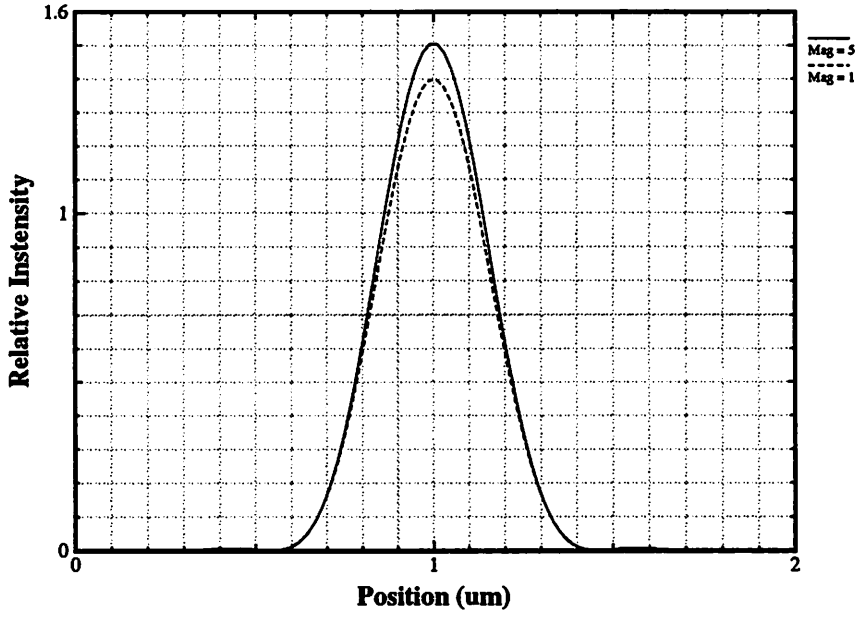
6.13.1 Input File

```
#
# Projection Lithography using SPLAT -- Test 13
#
stmt 1: print = 3
stmt 2: lambda = 0.365 um
stmt 3: na = 0.6
stmt 31: magnification = 1.0 ;
stmt 5: sigma = 0.2
stmt 6: mask = 1.0 x 1.0 trans = 0 ;
stmt 7: cutout = (0.00, 0.00) 0.25 x 0.25 trans = 1 ;
stmt 10: 0 2;
stmt 14: cutline from (0.0, -1.0) to (0.0, 1.0)
&     with 100 points mode = 1 in 'fig_m1.plot'
end 0
```

6.13.2 Output File

```
 2D optical imaging with aberrations
 apodization and modified illumination -- V4.2
                  3/23/93 -- DN/DL/KT

 Trial 1: Print level= 3
 Trial 2: Lambda = 0.3650 microns
 Trial 3: N.A. = 0.6000
 Trial 31: Optical System Demagnification = 1.000
 #     31: The mask image will be calculated using the mask dimensions
at
 #     31: the mask plane as specified in the input.
 Trial 5: Sigma = 0.2000
 Trial 5: Sigma In = 0.0000
 Trial 6: Mask size = 1.0000 x 1.0000 @ 0.0000 layout scale =1.00 (um/div)
 #     6: x size requires 6 harmonics.
 #     6: y size requires 6 harmonics.
 Trial 7: Mask Cutout =( 0.0000, 0.0000)x( 0.2500, 0.2500) @ 1.0000 < 0.0000>
 Trial 10: Calculate image Fourier Transform
 #     10: Force=2 -- 2D Numerical Integration
 #     10: Symmetry : T( f1, g1, f2, g2) =
 #     10: T(-f1,-g1,-f2,-g2)
 #     10: T( f1,-g1, f2,-g2)
 #     10: T(-f1, g1,-f2, g2)
 #     10: T( f2, g2, f1, g1)
 #     10: TCC computation time = 2.83000 sec.
```

```
#     10: Imaged with 6 by 6 harmonics
#     10: TCC calls: 283 zeros: 54
Trial 14: 5-decimal plot line : ( 0.000, -1.000)..( 0.000, 1.000)
      100 points saved in "fig_m1.plo"
Program execution terminated.
User Time (CPU) = 4.92000 sec, System Time = 0.59000 sec.
```



6.14 Example 14 - Defocus

This example demonstrates the difference between true defocus and approximate defocus. STATEMENT 4 was:

stmt 4: defocus = 1.0 ;

in the first file to simulate true defocus since that is the default for MODE. In the second file, STATEMENT 4 was changed to:

stmt 4: defocus = 1.0 mode = 0 ;

which demonstrate approximate defocus, MODE = 0. Only the input and output files for this second trial are shown, but, again, the plot shows the overlay of the two resulting curves.

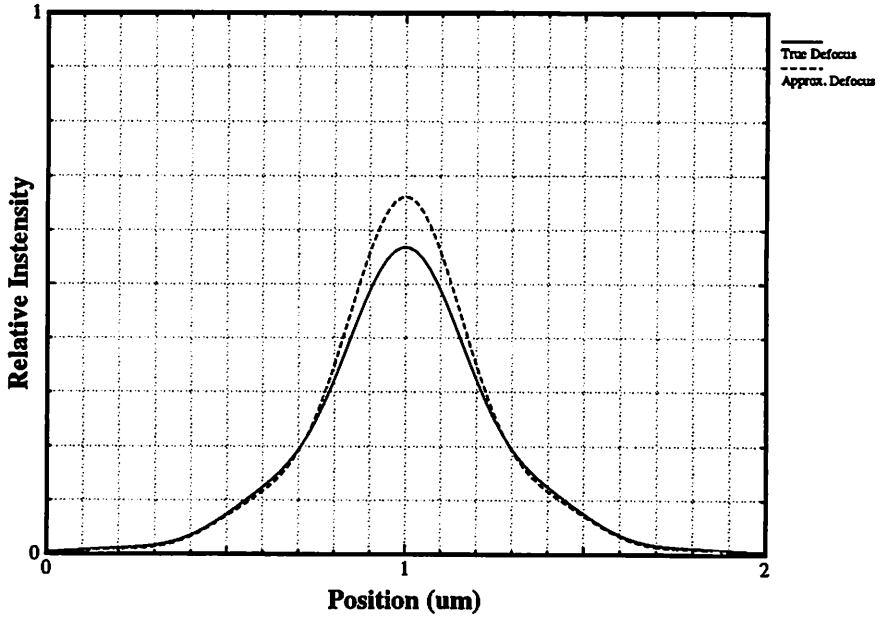
6.14.1 Input File

```
#
# Projection Lithography using SPLAT -- Test 14
#
stmt 1: print = 3
stmt 2: lambda = 0.365 um
stmt 3: na = 0.6
stmt 4: defocus= 1.0 mode = 0
stmt 5: sigma = 0.2
stmt 6: mask = 1.0 x 1.0 trans = 0 ;
stmt 7: cutout = (0.00, 0.00) 0.25 x 0.25 trans = 1 ;
stmt 10: 0 2;
stmt 14: cutline from (0.0, -1.0) to (0.0, 1.0)
&       with 100 points mode = 1 in 'fig_z1.app.plot'
end 0
```

6.14.2 Output File

```
2D optical imaging with aberrations
apodization and modified illumination -- V4.2
                    3/23/93 -- DN/DL/KT

Trial 1: Print level= 3
Trial 2: Lambda = 0.3650 microns
Trial 3: N.A. = 0.6000
#     4: Approximate Defocus Calculation
Trial 4: Defocus = 1.0000 microns [ 1.9726 Rayleigh Units ]
Trial 5: Sigma = 0.2000
Trial 5: Sigma In = 0.0000
Trial 6: Mask size = 1.0000 x 1.0000 @ 0.0000 layout scale =1.00 (um/div)
#     6: x size requires 6 harmonics.
#     6: y size requires 6 harmonics.
Trial 7: Mask Cutout =( 0.0000, 0.0000)x( 0.2500, 0.2500) @ 1.0000 < 0.0000>
Trial 10: Calculate image Fourier Transform
#     10: Force=2 -- 2D Numerical Integration
#     10: Symmetry : T( f1, g1, f2, g2) =
#     10: T(-f1,-g1,-f2,-g2)
#     10: T( f1,-g1, f2,-g2)
#     10: T(-f1, g1,-f2, g2)
#     10: conjg[T( f2, g2, f1, g1)]
#     10: TCC computation time = 4.49000 sec.
#     10: Imaged with 6 by 6 harmonics
#     10: TCC calls: 283 zeros: 54
```

```
Trial 14: 5-decimal plot line : ( 0.000, -1.000)..( 0.000, 1.000)
      100 points saved in "fig_z1.app"
Program execution terminated.
User Time (CPU) = 6.72000 sec, System Time = 0.26000 sec.
```



6.15 Example 15 - Imaging within Thin-Films

The following example shows a simulation which includes thin-film interference effects and the use of the lookup table.

6.15.1 Input Files

```
# 0.40 um contact
stmt 1: print = 3
stmt 2: lambda = 0.365 um
stmt 3: na = 0.700
stmt 4: defocus = 0.1 mode = 1 #true defocus
stmt 5: sigma = 0.500
stmt 6: mask = 0.40 x 0.40 trans = 0 ;
stmt 7: cutout = (0.00, 0.00) 0.200 x 0.20 trans = 1 ;
stmt 35: resist layer =59 'case1.dat'
stmt 31: mag = 5 mode = 0 ;
stmt 32: 50; # use table lookup [50 x 50] Grid
```

```
stmt 10: 0 2 ;
stmt 11: (-0.40 -0.40) 0.8 0.8 'lyr_59.con.z+0.5.ctr';
stmt 14: cutline from (-0.40, 0.0) to (0.40, 0.0)
& with 60 points mode = 1 in 'tf.59.con.z+0.5.plot'
end 0
```

Thin-film Input file: "case1.dat"

```
4.7300 -0.1400 1.00 0.00
1.6800 0.7400 0.200 0.01200 10.0 1.000 93
```

6.15.2 Output File

```
2D optical imaging with aberrations -- V5.0
                    11/10/93 -- DL/DN/KT

 Trial 1: Print level= 3
 Trial 2: Lambda = .3650 microns
 Trial 3: N.A. = .7000
 Trial 4: Defocus = .1000 microns [ .2685 Rayleigh Units ]
 # 4: True Defocus Calculation
 Trial 5: Sigma = .5000
 # 5: Sigma In = .0000
 Trial 6: Mask size = .4000 x .4000 @ .0000 layout scale =1.00 (um/
div)
 # 6: x size requires 4 harmonics.
 # 6: y size requires 4 harmonics.
 Trial 7: Mask Cutout =( .0000, .0000)x( .2000, .2000)
& @ 1.0000 < .0000>
 Trial 35: High NA
 # 35: Substrate (N,K) = ( 4.7300, -.1400)
 # 35: "Air" (N,K) = ( 1.0000, .0000)
 # 35: Resist: nreal = 1.680000
 # 35: A = .740000
 # 35: B = .200000
 # 35: C = .012000
 # 35: dose = 240.000000 mJ/cm^2
 # 35: thickness = 1.000000 microns
 # 35: layers = 93
 # 35: dz = .010753 microns
 # 35: Current Resist Layer = 59 of 93
Trial 31: Optical System Demagnification = 5.000
 # 31: The mask image will be calculated using the mask dimensions at
 # 31: the mask plane as specified in the input.
```

```
Trial 32: Initializing table ...
 # 32: [ 50 X 50 ]
 # 32: Table initialzed
 Trial 10: Calculate image Fourier Transform
 # 10: Force=2 -- 2D Numerical Integration
 # 10: Symmetry : T( f1, g1, f2, g2) =
 # 10: T(-f1,-g1,-f2,-g2)
 # 10: T( f1,-g1, f2,-g2)
 # 10: T(-f1, g1,-f2, g2)
 # 10: conjg[T( f2, g2, f1, g1)]
 # 10: TCC computation time = .00000 sec.
 # 10: Imaged with 4 by 4 harmonics
 # 10: TCC calls: 46 zeros: 51
 Trial 11: Image intensity contour data stored in "lyr_59.con"
 Trial 14: 5-decimal plot line : ( -.400, .000)..( .400, .000)
 60 points saved in "tf.59.con."
 Trial 0 : End of session
 User Time (CPU) = .00000 sec, System Time = .00000 sec.
```

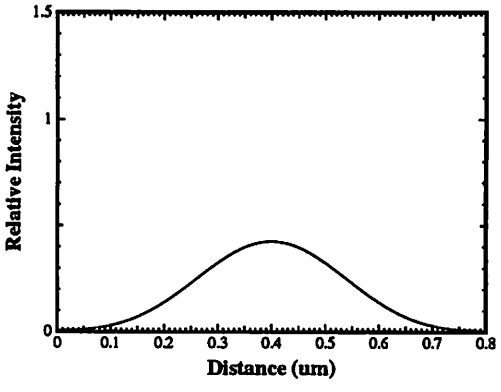
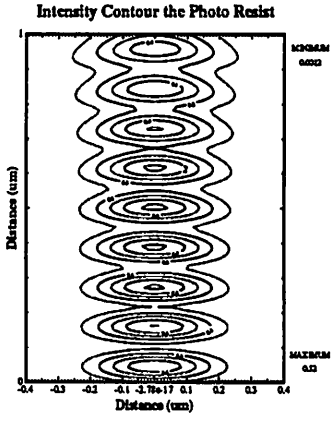


Image in 59th photoresist layer



2D intensity contour of a vertical slice through the photoresist. This image was made by combining the 93 1D intensity plots.

6.16 Example 16 - Projection X-Ray Lithography

The following example demonstrates the use of STATEMENT 33 which reads in a composite wavefront map using the output from CODE-V. Currently, no other format is readable, though the code can be extended.

6.16.1 Input Files

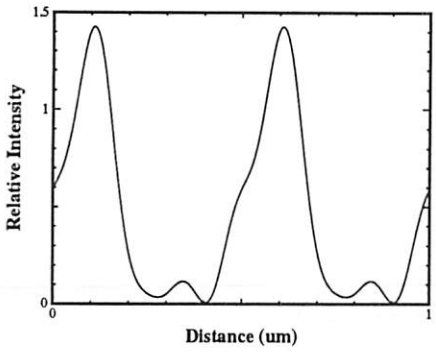
```
# Splat Projection X-Ray Lithography Example
```

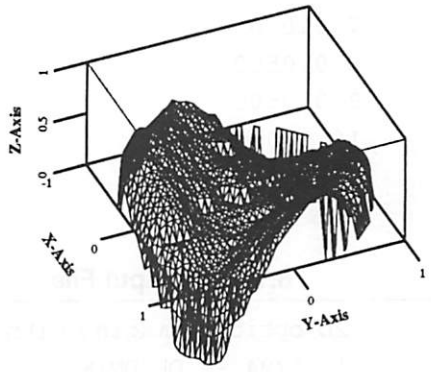
```
#
# 0.25 um lines and spaces
#
1 : print level = 3;
2 : 0.013 ;
3 : na = 0.0835;
30: 0.002 (-.2968, -.2968)
6 : mask = 0.50 x 0.250 trans = 0;
7 : cutout = (0.0, 0.0) 0.125 x 0.250 trans =1;
7 : cutout = (0.375, 0.0) 0.125 x 0.250 trans =1;
33 : mode = 3 'pupil.map.plot';
34 : scale = 0.01 of a wavelength 'tejmul.dat';
10 : 0 2;
14 : cutline from (-0.50, 0) to (0.50, 0.0)
& with 200 points mode=1 in 'ls_SPOT.plot';
end 0;
```

6.16.2 Output File

```
2D optical imaging with aberrations--V5.0
                    11/10/93--DL/DN/KT

Trial1: Printlevel=3
Trial2: Lambda=.0130 microns
Trial3: N.A.=.0835
Trial30: Spot1 Radius=.0020
Trial30: Coordinates of Spot1 =-.2968-.2968
Trial6: Masksize=.5000x.2500@.0000 layoutscale = 1.00(um/div)
#6: x size requires 18 harmonics.
#6: y size requires 8 harmonics.
Trial7: MaskCutout=(.0000,.0000)x(.1250,.2500)@1.0000<.0000>
Trial7: MaskCutout=(.3750,.0000)x(.1250,.2500)@1.0000<.0000>
Trial34: Loading Fringe Tablefrom "tejmul.dat"
Trial 33: Saving Pupil in "pupil.map.plot " (mode = 3)
Trial10: Calculate image Fourier Transform
#10: Force=2--2D Numerical Integration
#10: Setting System Partial Coherence=.42174
#10: TCC computation time=.00000sec.
#10: Imaged with 18 by 8 harmonics
#10: TCCcalls: 4624 zeros:24617
Trial14: 5-decimal plot line:(-.500,.000)..(.500,.000)
200 points saved in "ls_SPOT.pl"
Trial0: End of session
```

```
User Time(CPU)=.00000sec, System Time=.00000sec.
```





Composite Pupil Map (CodeV output)

6.17 Example 17 - Zernike Polynomial

The following example demonstrates the use of STATEMENT 36 which reads in a file containing Zernike polynomial multipliers to describe the wavefront aberration.

6.17.1 Input Files

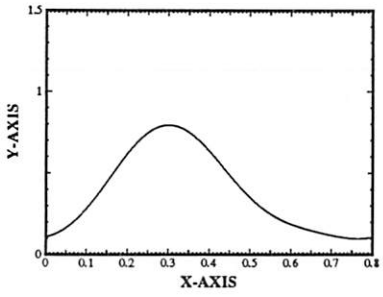
```
# Zernike Polynomial example
# 0.40 um square contact
stmt 1: print = 3
stmt 2: lambda = 0.365 um
stmt 3: na = 0.700
stmt 5: sigma = 0.500
stmt 6: mask = 0.40 x 0.40 trans = 0 ;
stmt 7: cutout = (0.00, 0.00) 0.20 x 0.20 trans = 1 ;
stmt 36: Zernike Polynomial Multipliers in 'test.zp.mults' ;
stmt 33: mode = 5 format=1 'test.zp.pupil.map'; #Save pupil map
stmt 10: 0 2 ;
stmt 11: (-0.40, 0.40) 0.80 0.80 'test.zp.ctr;
stmt 14: cutline from (-0.40, 0.0) to (0.40, 0.0)
&     with 100 points mode = 1 in 'test.zp.plot';
stmt 0;

Zernike polynomial multiplier input file "test.zp.mults"
1 0.1000
2 0.000
```

```
3 0.000
4 0.000
5 0.000
6 0.000
7 0.000
8 0.0500
9 0.0500
10 0.0500
11 0.0500
```

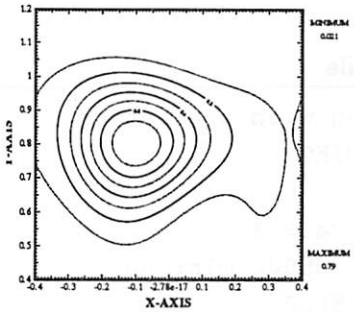
6.17.2 Output File

```
2D optical imaging with aberrations -- V5.0
10/1/94 -- DL/DN/KT

Trial 1: Print level= 3
Trial 2: Lambda = 0.3650 microns
Trial 3: N.A. = 0.7000
Trial 5: Sigma = 0.5000
# 5: Sigma In = 0.0000
Trial 6: Mask size = 0.4000 x 0.4000 @ 0.0000 layout scale =1.00 (um/div)
# 6: x size requires 4 harmonics.
# 6: y size requires 4 harmonics.
Trial 7: Mask Cutout =( 0.0000, 0.0000)x( 0.2000, 0.2000) @ 1.0000 <0.0000>
Trial 36: Zernike Polynomials
# 36: Multipliers Specified as Fractions of Wave
# 36: Loaded Multipliers from "test.zp.mults "
# 36: OPD at (1,0) = 0.6113 Wavelengths
# 36: OPD at (0,1) =-0.1877 Wavelengths
# 36: OPD at (0.707,0.707) = 0.2532 Wavelengths
# 36: OPD at (0,0) = 0.1118 Wavelengths
Trial 33: Saving Pupil in "test.zp.pupil.m" (mode = 5)
Trial 10: Calculate image Fourier Transform
# 10: Force=2 -- 2D Numerical Integration
# 10: Symmetry : T( f1, g1, f2, g2) =
# 10: conjg[T( f2, g2, f1, g1)]
# 10: TCC computation time = 13.07000 sec.
# 10: Imaged with 4 by 4 harmonics
# 10: TCC calls: 153 zeros: 200
Trial 11: Image intensity contour data stored in "test.zp.ct"
Trial 14: 5-decimal plot line : ( -0.400, 0.000)..( 0.400, 0.000)
100 points saved in "test.zp.pl"
Program execution terminated.
```

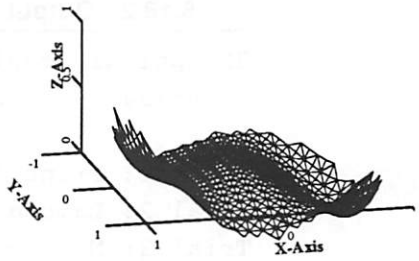
```
User Time (CPU) = 24.47000 sec, System Time = 0.92000 sec.
```



test.zp.plot



test.zp.ctr



test.zp.pupil.map

6.18 Example 18 - Arbitrary Source

The following example demonstrates using and defining an arbitrarily shaped source with uniform intensity.

6.18.1 Input Files

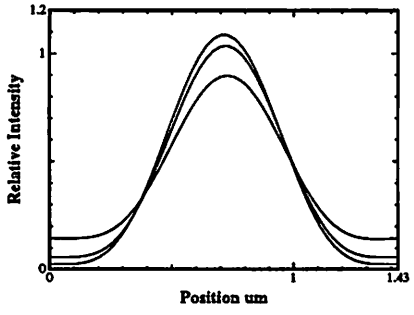
```
# Arbitrary source example
# Wiper shaped source from an X-Ray Lithography System
#
# .71429 um lines
#
Statement 1 : Printlevel 3                ;# set print level
Statement 2 : lambda = 0.5 um             ;# wavelength
Statement 3 : NA = 0.5                    ;# numerical aperture
Statement 4 : Defocus= 0.0 um             ;# defocus
Statement 6 : mask = 0.71429 x 0.71429
& at 0 transmittance                      ;# define working area
Statement 7 : cutout = (0.0, 0.0)
& (0.71429, 0.35714) at 1                 ;# define mask openings
39: 201 ;# set the source grid points
# define the source
stmt 43: ( .1 .025) (-.1 .525) (-.1 .525) ( .1 .025) 1.0 ;
stmt 43: ( .225 -.525) ( .025 -.025) ( .025 -.025) ( .225 -.525) 1.0 ;
stmt 43: (-.025 -.525) (-.225 -.025) (-.225 -.025) (-.025 -.525) 1.0 ;
stmt 43: (-.250 -.25 ) (-.450 .25 ) (-.450 .25 ) (-.250 -.25 ) 1.0 ;
stmt 43: ( .250 .25 ) ( .450 -.25 ) ( .450 -.25 ) ( .250 .25 ) 1.0 ;
#
```

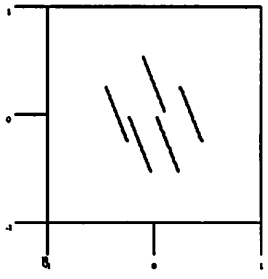
```
stmt 46: format=1 "smap" ;# generate a plot of the source
Statement 10: 0 2 ;# calculate the Fourier coeffs
Statement 14: intensity (0, -0.71429) .. (0, 0.71429)
& to 'wiper.plot' ;#2-D intensity profiles
stmt 0;
```

6.18.2 Output File

```
 2D optical imaging with aberrations -- V5.0
 10/05/94 -- DL/DN/KT

 Trial 1: Print level= 3
 Trial 2: Lambda = .5000 microns
 Trial 3: N.A. = .5000
 Trial 4: Defocus = .0000 microns [ .0000 Rayleigh Units ]
 Trial 6: Mask size = .7143 x .7143 @ .0000 layout scale =1.00 (um/
div)
 # 6: x size requires 4 harmonics.
 # 6: y size requires 4 harmonics.
 Trial 7: Mask Cutout =( .0000, .0000)x( .7143, .3571) @ 1.0000 <
.0000>
 Trial 39: No. of Source Grid Points = 201
 # 39: Using Adaptive Integration
 Trial 43: Adding Polygon to Source @ 1.0000
 # 43: Segment from ( .1000, .0300) to ( -.1000, .5300)
 # 43: Segment from ( -.1000, .5300) to ( -.1000, .5300)
 # 43: Segment from ( -.1000, .5300) to ( .1000, .0300)
 # 43: Segment from ( .1000, .0300) to ( .1000, .0300)
 Trial 43: Adding Polygon to Source @ 1.0000
 # 43: Segment from ( .2300, -.5300) to ( .0300, -.0300)
 # 43: Segment from ( .0300, -.0300) to ( .0300, -.0300)
 # 43: Segment from ( .0300, -.0300) to ( .2300, -.5300)
 # 43: Segment from ( .2300, -.5300) to ( .2300, -.5300)
 Trial 43: Adding Polygon to Source @ 1.0000
 # 43: Segment from ( -.0300, -.5300) to ( -.2300, -.0300)
 # 43: Segment from ( -.2300, -.0300) to ( -.2300, -.0300)
 # 43: Segment from ( -.2300, -.0300) to ( -.0300, -.5300)
 # 43: Segment from ( -.0300, -.5300) to ( -.0300, -.5300)
 Trial 43: Adding Polygon to Source @ 1.0000
 # 43: Segment from ( -.2500, -.2500) to ( -.4500, .2500)
 # 43: Segment from ( -.4500, .2500) to ( -.4500, .2500)
 # 43: Segment from ( -.4500, .2500) to ( -.2500, -.2500)
 # 43: Segment from ( -.2500, -.2500) to ( -.2500, -.2500)
 Trial 43: Adding Polygon to Source @ 1.0000
```

```
# 43: Segment from ( .2500, .2500) to ( .4500, -.2500)
# 43: Segment from ( .4500, -.2500) to ( .4500, -.2500)
# 43: Segment from ( .4500, -.2500) to ( .2500, .2500)
# 43: Segment from ( .2500, .2500) to ( .2500, .2500)
Trial 46: Saving Source Map in "smap "
# 46: (Pdraw format)
Trial 10: Calculate image Fourier Transform
# 10: Weighted Illumination Source Area = .02550
# 10: Setting System Partial Coherence = .58775
# 10: Force=2 -- 2D Numerical Integration
# 10: Symmetry : T( f1, g1, f2, g2) =
# 10: T( f2, g2, f1, g1)
# 10: Imaged with 4 by 4 harmonics
# 10: TCC calls: 88 zeros: 265
Trial 14: 3-decimal plot line : ( .000, -.714)..( .000, .714)
50 points saved in "wiper.plot"
Trial 0 : End of session
```





'smap'-- Plot of the illumination source

'wiper.plot' (0 defocus) combined with two other plots at 0.5 and 1.0 R.U. defocus levels. Notice the plots become asymmetric with focus. This is due to the interaction of defocus with the asymmetric source.

6.19 Example 19 - Arbitrary Source with Non-Uniform Intensity

The following example demonstrates the use of STATEMENT 47 which assigns the illumination source a non-uniform intensity distribution contained in a CONTOUR format square matrix.

6.19.1 Input Files

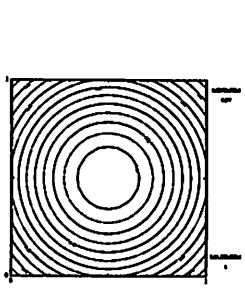
```
# Example using Statement 47 which loads in a non-uniform
# illumination file to define the intensity of the source.
#
Statement 1 : Printlevel 3              ;# set print level
```

```
Statement 2 : lambda = 0.5 um          ;# wavelength
Statement 3 : NA = 0.5                 ;# numerical aperture
Statement 4 : Defocus= 0.0 um          ;# defocus
Statement 6 : mask = 2um x 2um
& at 0 transmittance                   ;# define working area
Statement 7 : cutout = (0.0, 0.0)
& 0.5 x 0.5 at 1                       ;# define mask openings
39: 21 ;# define the grid size
# define a circular source
40: 0 0 .50 1.0                        ;# define a circular source
47: 1 0 'illum_radial.dat'             ;# grid size must match 39:
46: 1 'smap'                           ;# plot the non-uniform source
10: ;
14: intensity (-2.0, 0) .. (2.0, 0)
& to 'non-uniform.illum.plot' ;#2-D intensity profiles
0;
```

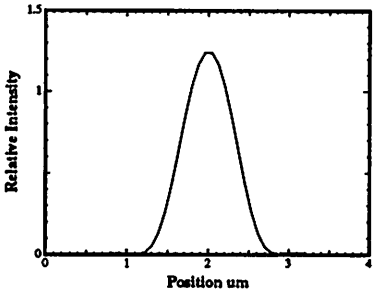
6.19.2 Output File

```
 2D optical imaging with aberrations -- V5.0
 10/05/94 -- DL/DN/KT

 Trial 1: Print level= 3
 Trial 2: Lambda = .5000 microns
 Trial 3: N.A. = .5000
 Trial 4: Defocus = .0000 microns [ .0000 Rayleigh Units ]
 Trial 6: Mask size = 2.0000 x 2.0000 @ .0000 layout scale =1.00 (um/
div)
 # 6: x size requires 12 harmonics.
 # 6: y size requires 12 harmonics.
 Trial 7: Mask Cutout =( .0000, .0000)x( .5000, .5000) @ 1.0000 <
.0000>
 Trial 39: No. of Source Grid Points = 21
 # 39: Using Adaptive Integration
 Trial 40: Adding Source Circle
 # 40: ( .0000 .0000) r = .5000 @ 1.0000
 Trial 47: Loading Non-Uniform Illumination from "illum_radial.da"
 Trial 46: Saving Source Map in "smap "
 # 46: (Pdraw format)
 Trial 10: Calculate image Fourier Transform
 # 10: Weighted Illumination Source Area = .89907
 # 10: Setting System Partial Coherence = .63852
 # 10: Symmetry : T( f1, g1, f2, g2) =
```

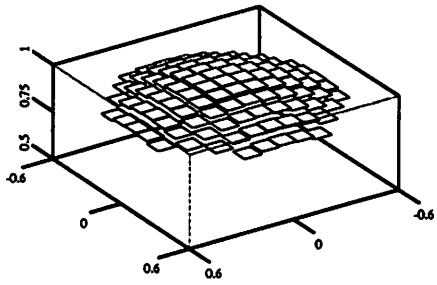
```
# 10: T(-f1,-g1,-f2,-g2)
# 10: T( f1,-g1, f2,-g2)
# 10: T(-f1, g1,-f2, g2)
# 10: T( f2, g2, f1, g1)
# 10: Imaged with 12 by 12 harmonics
# 10: TCC calls: 1724 zeros: 1973
Trial 14: 3-decimal plot line : ( -2.000, .000)..( 2.000, .000)
50 points saved in "non-unifor"
Trial 0 : End of session
```



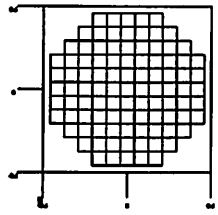
'illum_radial.dat'



'non-uniform_illum.plot'



'smap'



'smap' (top view)

6.20 Example 20 - Optical Transmission with Apodization

The following example demonstrates the use of STATEMENTS 49 and 50. STATEMENT 50 adds optical transmission with apodization (OTA) to the pupil (modifies the magnitude of the pupil). Because the clear field value is no longer 1, STATEMENT 49 is used to renormalize the output intensities to the new clear field value which is determined on a separate simulation.

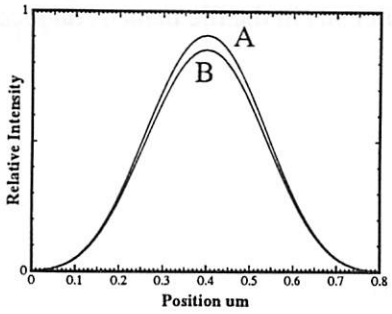
6.20.1 Input Files

```
# 0.40 um contact
# OTA = 1 - 0.25 rho^2
# Intensity values are renormalized to open field
# (stmt 49) which was computed from a separate
# SPLAT run.
#
stmt 1: print = 3
stmt 2: lambda = 0.365 um
stmt 3: na = 0.700
stmt 5: sigma = 0.500
stmt 6: mask = 0.80 x 0.80 trans = 0
stmt 7: cutout = (0.00, 0.00) 0.200 x 0.20 trans = 1
```

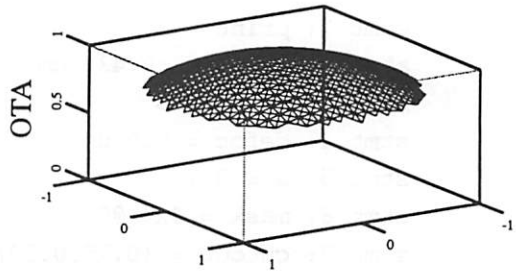
```
stmt 49: open field intensity = .938909
stmt 50: mode=0 a=0.0 b=0.0 c=0.25
stmt 33: mode 0
stmt 10: 0 2
stmt 11: (-0.40 -0.40) 0.8 0.8 'out.ctr'
stmt 14: cutline from (-0.40, 0.0) to (0.40, 0.0)
& 101 0 in 'ota.plot'
end 0
```

6.20.2 Output File

```
 2D optical imaging with aberrations -- V5.0
 10/05/94 -- DL/DN/KT

 Trial 1: Print level= 3
 Trial 2: Lambda = .3650 microns
 Trial 3: N.A. = .7000
 Trial 5: Sigma = .5000
 # 5: Sigma In = .0000
 Trial 6: Mask size = .8000 x .8000 @ .0000 layout scale =1.00 (um/
div)
 # 6: x size requires 8 harmonics.
 # 6: y size requires 8 harmonics.
 Trial 7: Mask Cutout =( .0000, .0000)x( .2000, .2000) @ 1.0000 <
.0000>
 Trial 49: Renormalizing Open Field Intensity to .93891
 Trial 50: Optical Transmission with Apodization
 # 50: OTA += 1 - .0000[1] - .0000[rho] - .2500[rho^2]
 # 50: - .0000[rho^3] - .0000[rho^4]
 Trial 33: Saving Pupil in "out.3D.plot " (mode = 0)
 # 33: Output in PLOTMTV format
 Trial 10: Calculate image Fourier Transform
 # 10: Force=2 -- 2D Numerical Integration
 # 10: Symmetry : T( f1, g1, f2, g2) =
 # 10: T(-f1,-g1,-f2,-g2)
 # 10: T( f1,-g1, f2,-g2)
 # 10: T(-f1, g1,-f2, g2)
 # 10: T( f2, g2, f1, g1)
 # 10: Imaged with 8 by 8 harmonics
 # 10: TCC calls: 508 zeros: 373
 Trial 11: Image intensity contour data stored in "out.ctr "
 Trial 14: 3-decimal plot line : ( -.400, .000)..( .400, .000)
 101 points saved in "ota.plot "
 Trial 0 : End of session
```



'ota.plot' A) normalized B) not normalized to open field intensity



'out.3D.plot' (magnitude of the pupil)

6.21 Example20 - Linking Splat to Sample

As mentioned earlier, SPLAT can be used to generate intensity profiles along a cutline through the user-specified mask. These profiles, which are calculated by STATEMENT 14 (see Trial specifications), are saved in plot-data files which are similar in format to the SAMPLE 'f77punch7' files. Note : the SPLAT-SAMPLE link is only available on SAMPLE Version 1.7 or later.

The procedure for linking SPLAT to SAMPLE is as follows :

1. First, generate the intensity profiles using TRIAL 14 of SPLAT. Each profile generated will be saved in a file whose name is specified by the user. For example, the trial statement

```
Trial 14 : (0,0)..(2,2) 'plotfile';
```

will produce an intensity profile between the specified coordinates, and that profile will be stored in a file named 'plotfile'.

2. The intensity plot file generated by SPLAT has to be renamed '2ddat', or copied into a file with that name. This is because \fBSAMPLE\fP expects its intensity input to be in an '2ddat' file.
3. Replace the SAMPLE trial statement IMAGERUN with the statement READIMAGE (TRIAL 18) (no arguments) to read the intensity profile. The trial statements LAMBDA, PROJ and PARCOHDEF, which were previously used to specify the imaging parameters are now no longer needed, but can be left in anyhow. In addition, the mask-specification trial statements such as LINESPACE or PHASEMASK can also be left out. If an intensity plot is desired then the trial statement OPTIMGEXP has to precede the READIMAGE statement, as in the example below. Further process steps can then be simulated normally.

As an example, let us use the intensity plot-data file that was generated by the following two SPLAT file and contains intensity data along a diagonal through a

pair of opaque elbows. The first file calculated the image for the pair of opaque elbows and stored the calculated coefficients in the file named *samp.cof*.

```
stmt 1: print = 3
stmt 2: lambda = 0.436 um
stmt 3: na = 0.28
stmt 4: defoc = 0.0 um
stmt 5: s = 0.7
stmt 6: mask = 4x4 @0
stmt 7: cutout = (0.75,0.50) 0.75 x 2.75 @1
stmt 7: cutout = (1.50,2.50) 2.00 x 0.75 @1
stmt 7: cutout = (2.00,0.50) 0.75 x 1.50 @1
stmt 7: cutout = (2.75,1.25) 0.75 x 0.75 @1
stmt 10
stmt 12: save = 'samp.cof'
end 0
```

The second file then read the coefficients back and calculated the intensity profiles:

```
Trial 1 : 2;
Trial 13: load file 'test3.cof';
Trial 14: intensity (0,4) .. (4,0) to '2ddat';
Trial 0 :
```

The SAMPLE input data file 'samop1' (from "PHOTOLITHOGRAPHY EXAMPLES") is modified by replacing the IMAGERUN statement with the trial statement READIMAGE. The contents of the modified SAMPLE file are as follows :

```
lambda 0.4358 ;                  # lambda parameter (optional)
proj 0.28 ;                      # numerical aperture (optional)
parcohdef 0 0.7 0.0 ;            # sigma and defocus (optional)
optimgexp 1 0 1 0 0 ;            # profile coordinates for plot
readimage ;                      # read external file for image profile
resmodel ((0.4358))
       (0.551, 0.058, 0.010)
       1.68, ((-0.02))) (0.7133) ;# resist exposure parameters
layers (4.73,-0.14)
 (1.47,0.0,0.0741) ;             # layer parameters
dose 150 ;                       # dose for exposure
exposerun ;                      # run exposure machine
optdevelop 0 1 0 ;               # profile coordinates for plot
devrate 1 (5.63, 7.43, -12.6) ;  # resist development parameters
devtime 15 75, 5 ;               # development times
```

```
developrun ;                    # run development machine
descumspec 0.02, 0.04, 3 ;      # run descum
```

And finally, SAMPLE is run. SAMPLE will read the intensity data from the '2ddat' file, and following further process steps, such as development, will print out the resist profiles to the 'f77punch7' file. As a result, the effect of two-dimensional structures on processing can now be examined. The output files of this example are very lengthy and are excluded from this manual.

7.0 Miscellaneous Utilities

A number of utilities have been created to manipulate the data files in splat. These programs are used primarily to run many simulations using either the thin-film interference and/or projection X-Ray lithography options.

7.1 Thin-Film Utilities

The following UNIX scripts and or C programs have been created:

run.layers - a unix script to run many thin-film interference simulations using a single input file

Usage: *run.layers splat_input_file start_layer end_layer*

The example input file given below can be executed by typing:

> run.layers splat_input_file 1 93

This script searches for the word 'RESIST_LAYER' and replaces it with the current resist layer and then calls SPLAT. At the end of the simulation there will be 93 different 1D intensity files in the current directory. This script calls SPLAT, so SPLAT must be in your current directory or in your Unix environment PATH variable.

7.1.1 Thin-Film Batch Input File Example

```
# 0.40 um contact
stmt 1: print = 3
stmt 2: lambda = 0.365 um
stmt 3: na = 0.700
stmt 4: defocus = 0.5 mode = 1 #true defocus
stmt 5: sigma = 0.500
stmt 6: mask = 0.40 x 0.40 trans = 0 ;
```

```
stmt 7: cutout = (0.00, 0.00) 0.200 x 0.20 trans = 1 ;
stmt 35: resist layer = RESIST_LAYER 'case1.dat'
stmt 31: mag = 5 mode = 0 ;
stmt 32: 50 0; #50 Grid Points, Disable Symmetry
stmt 10: 0 2 ;
stmt 11: (-0.40 -0.40) 0.8 0.8 'tf_RESIST_LAYER.z+0.5.ctr';
stmt 14: cutline from (-0.40, 0.0) to (0.40, 0.0)
& with 60 points mode = 1 in 'tf.RESIST_LAYER.z+0.5.plot'
end 0
```

merge.layers - a unix script used to merge the 1D intensity created using run.-layers into a 2D intensity contour file

Usage: *merge.layers filename_RESIST_LAYER.plot start_layer end_layer*

To merge the 93 1D intensity plots in the above example, type:

> merge.layers tf_RESIST_LAYER.z+0.5.plot 1 93 > all.a.z+0.5.2D.ctr

merge.contours - a unix script used to merge the 2D intensity contour files created using run.layers 3D intensity file for input in SAMPLE-3D

Usage: *merge.layers filename_RESIST_LAYER.ctr start_layer end_layer*

To merge the 93 2D intensity contours in the above example type:

> merge.layers tf_RESIST_LAYER.z+0.5.ctr 1 93 > all.a.z+0.5.3D.ctr

7.2 Projection X-Ray Lithography Utilities

sweep - a unix script used to run many simulations with the source position at *num_spots* angular positions using only a single input file

Usage: *sweep splat.in num_spots*

To simulate 8 spot locations in the example given below type:

> sweep splat_input_file 8

The sources begin above the 3:00 position and are equi-angularly distributed. The radius of the starting spot location is calculated using STATEMENT 30 of the input file--This line MUST be present. To prevent output files from over-writing each other, the word SPOT is searched for and replaced with the current spot location. See the example below.

iavg - a C program to average the intensity values from 1D intensity files or 2D intensity contours

Usage: *iavg base_filename num_spots*

The *base_filename* must contain the word 'SPOT' which will be replaced by the current spot number. Following the example for *sweep* given above, *iavg* would be used in the following manner:

> iavg ls_SPOT.plot 8 > averge.plot

7.2.1 X-Ray Projection Lithography Batch File Example

```
#
# 0.25 um lines and spaces
#
1 : print level = 3;
2 : 0.013 ;
3 : na = 0.0835;
30: 0.002 (-.2968, -.2968)
6 : mask = 0.50 x 0.250 trans = 0;
7 : cutout = (0.0, 0.0) 0.125 x 0.250 trans =1;
7 : cutout = (0.375, 0.0) 0.125 x 0.250 trans =1;
34 : scale = 0.01 of a wavelength 'tejmul.dat';
10 : 0 2;
14 : cutline from (-0.50, 0) to (0.50, 0.0)
& with 200 points mode=1 in 'ls_SPOT.plot';
end 0;
```

7.3 General Utilites

count.triangles - a unix script to set the correct number of triangles in a PDRAW file generated from SPLAT using STAMENT 33.

Usage: *count.triangles pupil.map_file*

When saving the pupil function into a PDRAW file (STAMENT 33, *format_mode* = 1) the number of triangles specified in the header line may be incorrect depending on the mode and number of grid points used. The correct number of triangles is recorded on the last line of the data file. This script places the correct number of triangles into the appropriate header line automatically.

8.0 Appendix A : Zernike Polynomials

The first 64 Zernike Polynomials are listed below. For more information about Zernike polynomials refer to Born & Wolf and to the comments in the FORTRAN code (file: "zp.f").

n	m	B& W n	B& W m	Term #	Norm Const	Polynomial
0	0	*0*	*0*	**0**	1	1
1	+1	*1*	*1*	**1**	2	$\rho\cos\phi$
	-1			**2**	2	$\rho\sin\phi$
	0	*2*	*0*	**3**	$\sqrt{3}$	$2\rho^2-1$
2	+2	*2*	*2*	**4**	$\sqrt{6}$	$\rho^2\cos 2\phi$
	-2			**5**	$\sqrt{6}$	$\rho^2\sin 2\phi$
	+1	*3*	*1*	**6**	$\sqrt{8}$	$(3\rho^3-2\rho)\cos\phi$
	-1			**7**	$\sqrt{8}$	$(3\rho^3-2\rho)\sin\phi$
	0	*4*	*0*	**8**	$\sqrt{5}$	$6\rho^4-6\rho^2+1$
3	+3	*3*	*3*	**9**	$\sqrt{8}$	$\rho^3\cos 3\phi$
	-3			**10**	$\sqrt{8}$	$\rho^3\sin 3\phi$
	+2	*4*	*2*	**11**	$\sqrt{10}$	$(4\rho^4-3\rho^2)\cos 2\phi$
	-2			**12**	$\sqrt{10}$	$(4\rho^4-3\rho^2)\sin 2\phi$
	+1	*5*	*1*	**13**	$\sqrt{12}$	$(10\rho^5-12\rho^3+3\rho)\cos\phi$
	-1			**14**	$\sqrt{12}$	$(10\rho^5-12\rho^3+3\rho)\sin\phi$
	0	*6*	*0*	**15**	$\sqrt{7}$	$20\rho^6-30\rho^4+12\rho^2-1$
4	+4	*4*	*4*	**16**	$\sqrt{10}$	$\rho^4\cos 4\phi$
	-4			**17**	$\sqrt{10}$	$\rho^4\sin 4\phi$
	+3	*5*	*3*	**18**	$\sqrt{12}$	$(5\rho^5-4\rho^3)\cos 3\phi$
	-3			**19**	$\sqrt{12}$	$(5\rho^5-4\rho^3)\sin 3\phi$

n	m	B&W n	B&W m	Term #	Norm Const	Polynomial
	+2	6	2	**20**	$\sqrt{14}$	$(15\rho^6 - 20\rho^4 + 6\rho^2)\cos 2\phi$
	-2			**21**	$\sqrt{14}$	$(15\rho^6 - 20\rho^4 + 6\rho^2)\sin 2\phi$
	+1	7	1	**22**	4	$(35\rho^7 - 60\rho^5 + 30\rho^3 - 4\rho)\cos\phi$
	-1			**23**	4	$(35\rho^7 - 60\rho^5 + 30\rho^3 - 4\rho)\sin\phi$
	0	8	0	**24**	3	$70\rho^8 - 140\rho^6 + 90\rho^4 - 20\rho^2 + 1$
5	+5	5	5	**25**	$\sqrt{12}$	$\rho^5\cos 5\phi$
	-5			**26**	$\sqrt{12}$	$\rho^5\sin 5\phi$
	+4	6	4	**27**	$\sqrt{14}$	$(6\rho^6 - 5\rho^4)\cos 4\phi$
	-4			**28**	$\sqrt{14}$	$(6\rho^6 - 5\rho^4)\sin 4\phi$
	+3	7	3	**29**	4	$(21\rho^7 - 30\rho^5 + 10\rho^3)\cos 3\phi$
	-3			**30**	4	$(21\rho^7 - 30\rho^5 + 10\rho^3)\sin 3\phi$
	+2	8	2	**31**	$\sqrt{18}$	$(56\rho^8 - 105\rho^6 + 60\rho^4 - 10\rho^2)\cos 2\phi$
	-2			**32**	$\sqrt{18}$	$(56\rho^8 - 105\rho^6 + 60\rho^4 - 10\rho^2)\sin 2\phi$
	+1	9	1	**33**	$\sqrt{20}$	$(126\rho^9 - 280\rho^7 + 210\rho^5 - 60\rho^3 + 5\rho)\cos\phi$
	-1			**34**	$\sqrt{20}$	$(126\rho^9 - 280\rho^7 + 210\rho^5 - 60\rho^3 + 5\rho)\sin\phi$
	0	10	0	**35**	$\sqrt{11}$	$252\rho^{10} - 630\rho^8 + 560\rho^6 - 210\rho^4 + 30\rho^2 - 1$
6	+6	6	6	**36**	$\sqrt{14}$	$\rho^6\cos 6\phi$
	-6			**37**	$\sqrt{14}$	$\rho^6\sin 6\phi$
	+5	7	5	**38**	4	$(7\rho^7 - 6\rho^5)\cos 5\phi$
	-5			**39**	4	$(7\rho^7 - 6\rho^5)\sin 5\phi$
	+4	8	4	**40**	$\sqrt{18}$	$(28\rho^8 - 42\rho^6 + 15\rho^4)\cos 4\phi$
	-4			**41**	$\sqrt{18}$	$(28\rho^8 - 42\rho^6 + 15\rho^4)\sin 4\phi$
	+3	9	3	**42**	$\sqrt{20}$	$(84\rho^9 - 168\rho^7 + 105\rho^5 - 20\rho^3)\cos 3\phi$
	-3			**43**	$\sqrt{20}$	$(84\rho^9 - 168\rho^7 + 105\rho^5 - 20\rho^3)\sin 3\phi$
	+2	10	2	**44**	$\sqrt{22}$	$(210\rho^{10} - 504\rho^8 + 420\rho^6 - 140\rho^4 + 15\rho^2)\cos 2\phi$

n	m	B&W n	B&W m	Term #	Norm Const	Polynomial
	-2			**45**	$\sqrt{22}$	$(210\rho^{10} - 504\rho^8 + 420\rho^6 - 140\rho^4 + 15\rho^2)\sin 2\phi$
	+1	*11*	*1*	**46**	$\sqrt{24}$	$(462\rho^{11} - 1260\rho^9 + 1260\rho^7 - 560\rho^5 + 105\rho^3 - 6\rho)\cos\phi$
	-1			**47**	$\sqrt{24}$	$(462\rho^{11} - 1260\rho^9 + 1260\rho^7 - 560\rho^5 + 105\rho^3 - 6\rho)\sin\phi$
	0	*12*	*0*	**48**	$\sqrt{13}$	$924\rho^{12} - 2772\rho^{10} + 3150\rho^8 - 1680\rho^6 + 420\rho^4 - 42\rho^2 + 1$
7	+7	*7*	*7*	**49**	4	$\rho^7\cos 7\phi$
	-7			**50**	4	$\rho^7\sin 7\phi$
	+6	*8*	*6*	**51**	$\sqrt{18}$	$(8\rho^8 - 7\rho^6)\cos 6\phi$
	-6			**52**	$\sqrt{18}$	$(8\rho^8 - 7\rho^6)\sin 6\phi$
	+5	*9*	*5*	**53**	$\sqrt{20}$	$(36\rho^9 - 56\rho^7 + 21\rho^5)\cos 5\phi$
	-5			**54**	$\sqrt{20}$	$(36\rho^9 - 56\rho^7 + 21\rho^5)\sin 5\phi$
	+4	*10*	*4*	**55**	$\sqrt{22}$	$(120\rho^{10} - 252\rho^8 + 168\rho^6 - 35\rho^4)\cos 4\phi$
	-4			**56**	$\sqrt{22}$	$(120\rho^{10} - 252\rho^8 + 168\rho^6 - 35\rho^4)\sin 4\phi$
	+3	*11*	*3*	**57**	$\sqrt{24}$	$(330\rho^{11} - 840\rho^9 + 756\rho^7 - 280\rho^5 + 35\rho^3)\cos 3\phi$
	-3			**58**	$\sqrt{24}$	$(330\rho^{11} - 840\rho^9 + 756\rho^7 - 280\rho^5 + 35\rho^3)\sin 3\phi$
	+2	*12*	*2*	**59**	$\sqrt{26}$	$(792\rho^{12} - 2310\rho^{10} + 2520\rho^8 - 1260\rho^6 + 280\rho^4 - 21\rho^2)\cos 2\phi$
	-2			**60**	$\sqrt{26}$	$(792\rho^{12} - 2310\rho^{10} + 2520\rho^8 - 1260\rho^6 + 280\rho^4 - 21\rho^2)\sin 2\phi$
	+1	*13*	*1*	**61**	$\sqrt{28}$	$(1716\rho^{13} - 5544\rho^{11} + 6930\rho^9 - 4200\rho^7 + 1260\rho^5 - 168\rho^3 + 7\rho)\cos\phi$
	-1			**62**	$\sqrt{28}$	$(1716\rho)^{13} - 5544\rho^{11} + 6930\rho^9 - 4200\rho^7 + 1260\rho^5 - 168\rho^3 + 7\rho)\sin\phi$
	0	*14*	*0*	**63**	$\sqrt{15}$	$3432\rho^{14} - 12012\rho^{12} + 16632\rho^{10} - 11550\rho^8 + 4200\rho^6 - 756\rho^4 + 56\rho^2 - 1$