

**ELIPGRID-PC: A PC PROGRAM FOR CALCULATING
HOT SPOT PROBABILITIES**

J.R. Davidson

Date Published: October 1994

Prepared by
Oak Ridge National Laboratory
Health Sciences Research Division
Environmental Technology Section
Grand Junction, Colorado
managed by
Martin Marietta Energy Systems, Inc.
for the
United States Department of Energy
under contract DE-AC05-84OR21400

MASTER

rb

CONTENTS

FIGURES	v
TABLES	vii
ACKNOWLEDGEMENTS	ix
ABSTRACT	xi
1. INTRODUCTION	1
2. PREVIOUS WORK	1
3. PROGRAM ASSUMPTIONS	3
4. PROGRAM DESCRIPTIONS	6
4.1 ELIPGRID-1	6
4.2 ELIPGRID-2	6
4.3 ELIPGRID-PC	7
5. INSTALLATION EXAMPLE	10
6. SUMMARY	11
REFERENCES	12
APPENDIX A	DEMONSTRATION OF NONEQUIVALENCE OF ELIPGRID FORTRAN CODE AND SINGER AND WICKMAN'S EQUATION
APPENDIX B	TESTING PROCEDURE
APPENDIX C	TRIANGULAR GRID DISCONTINUITY
APPENDIX D	ELIPGRID-2 SOURCE CODE
APPENDIX E	ELIPGRID-PC SOURCE CODE
APPENDIX F	EGGRAPH SOURCE CODE

FIGURES

1.	Hypothetical subsurface pocket of contamination	4
2.	Grid configuration for finding hot spots	5
3.	Probability of hit vs total sample cost for a square grid	9
C.1	Probability of missing hot spot vs L/G ratio, triangular grid, 0.99 shape, and 15° angle	C-2
C.2	Probability of missing hot spot vs L/G ratio, triangular grid, 0.99 shape, and 0° angle	C-3
C.3	Probability of missing hot spot vs L/G ratio, triangular grid, 0.99 shape, and 30° angle	C-4
C.4	Probability of missing hot spot vs L/G ratio, triangular grid, 0.90 shape, and 15° angle	C-6
C.5	Probability of missing hot spot vs L/G ratio, triangular grid, 0.85 shape, and 15° angle	C-7
C.6	Probability of missing hot spot vs L/G ratio, triangular grid, 1.00 shape, and 15° angle	C-8
C.7	Probability of missing hot spot vs L/G ratio, triangular grid, 0.99 shape, and 15° angle	C-9
C.8	Probability of missing hot spot vs L/G ratio, triangular grid, 0.90 shape, and 15° angle	C-10
C.9	Probability of missing hot spot vs L/G ratio, triangular grid, 0.85 shape, and 15° angle	C-11
C.10	Probability of missing hot spot vs L/G ratio, triangular grid, 0.99 shape, and 15° angle	C-13
C.11	Probability of missing hot spot vs L/G ratio, triangular grid, 0.85 shape, and 15° angle	C-14

TABLES

B.1	Corrected sequential order of Singer's tables	B-1
B.2	Input file listing after style used in ELIPGRID	B-3
B.3	ELIPGRID-1 output file listing	B-6
B.4	ELIPGRID-2 output file listing	B-8
B.5	ELIPGRID-PC output file listing	B-11
B.6	ELIPGRID-PC SIF-style input file listing	B-13
B.7	ELIPGRID-PC SIF-style output file listing	B-16

ACKNOWLEDGEMENTS

The author would like to recognize two individuals outside of Oak Ridge National Laboratory (ORNL) and three individuals of ORNL Grand Junction for significant help in the development of *ELIPGRID-PC*:

Donald A. Singer for his helpful advice relating to *ELIPGRID*, a program he developed more than twenty years ago in a much tougher computer environment than is present today.

Richard O. Gilbert for his encouragement to resolve the problems that developed in adapting *ELIPGRID* to the personal computer.

Gloria H. Stevens, Project Manager for the ORNL Independent Verification Contract on the Grand Junction Project Office Remedial Action Project, for strong support of the development of *ELIPGRID-PC*.

Phil V. Egidi for his help with graphics in general and for tremendous help with the development of a poster session relating to *ELIPGRID-PC*

John E. Wilson for his excellent help with debugging code. His help was vital in the tedious task of tracking down subtle coding errors.

ABSTRACT

ELIPGRID-PC, a new personal computer program, has been developed to provide easy access to Singer's 1972 ELIPGRID algorithm for hot-spot detection probabilities. Three features of the program are the ability to determine: 1) the grid size required for specified conditions, 2) the smallest hot spot that can be sampled with a given probability, and 3) the approximate grid size resulting from specified conditions and sampling cost. ELIPGRID-PC also provides probability of hit versus cost data for graphing with spreadsheets or graphics software. The program has been successfully tested using Singer's published ELIPGRID results. An apparent error in the original ELIPGRID code has been uncovered and an appropriate modification incorporated into the new program.

1. INTRODUCTION

The standard approach for calculating the probability of detecting small, highly contaminated areas called hot spots is based on a punch-card-era computer program developed over 20 years ago. This program, ELIPGRID (Singer 1972), is the foundation for three programs developed by Oak Ridge National Laboratory (ORNL) for the IBM® personal computer (PC): ELIPGRID-1, a PC version of ELIPGRID; ELIPGRID-2, a modified PC version; and ELIPGRID-PC, a user-friendly PC version containing several new options not found in ELIPGRID.

ELIPGRID-1 is a direct translation of ELIPGRID to the PC and retains a coding error found in ELIPGRID's rectangular grid routine. ELIPGRID-2 is similar to ELIPGRID-1 but corrects the rectangular grid error. ELIPGRID-PC, though based on ELIPGRID's algorithms, is a new program that simplifies input file selection, data entry, and file output.

ELIPGRID-1 and ELIPGRID-2 can be viewed as transitional programs used to work out technical problems involved in adapting ELIPGRID to the PC. They are documented here to provide a record of this transition. ELIPGRID-PC, however, is intended as a full replacement for the ELIPGRID program.

2. PREVIOUS WORK

In 1969, Singer and Wickman published a mathematical procedure for determining the probability of locating elliptical geological deposits (Singer and Wickman 1969). Using this procedure, five computer programs were written to calculate values published as probability tables for various target shapes, grid types, and grid sizes. These programs were run on an IBM® System 370/67 computer.

In 1972, Singer published ELIPGRID, a FORTRAN IV program based on Singer and Wickman's mathematical procedure (Singer 1972). This program calculated the probability of success in locating elliptical targets with square, rectangular, and hexagonal (triangular) grids. The data input and code were designed for the then-standard punch-card computer.

Zirschky and Gilbert developed a nomographic procedure based on ELIPGRID to assist with the detection of highly contaminated areas at chemical- or nuclear-waste disposal sites (Zirschky and Gilbert 1984). Gilbert used these nomographs as the basis for the chapter "Locating Hot Spots" in his widely referenced book on environmental statistical methods (Gilbert 1987). These nomographs were subsequently used by the U.S. Environmental Protection Agency (EPA) to develop tables for calculating the probability of missing various hot-spot shapes using triangular and square sampling grids (U.S.EPA 1989).

Gilbert's nomographs and the EPA tables have some inherent limitations not in the original ELIPGRID program. Three limitations are:

- (1) Probabilities for only one rectangular sampling grid are given in Gilbert's nomographs; no data for rectangular grids are given in the EPA tables.
- (2) Specific orientation angles for suspected hot spots are not allowed. For example, if the probability of detecting a given target with a given grid for a specific orientation angle is desired, the tables and nomographs do not provide this information.
- (3) Data extracted from a graph are less likely to be accurate than output from a computer program given the same input information.

ELIPGRID-PC removes these limitations by: 1) allowing a large number of rectangular grids, 2) allowing orientation angles for suspected hot spots to be specified, and 3) calculating the results with a computer algorithm.

3. PROGRAM ASSUMPTIONS

The following assumptions underlie both the original ELIPGRID and ELIPGRID-PC:

1. The target (hot spot) is assumed to be circular or elliptical. See Fig. 1 for an illustration of an elliptical subsurface pocket of contamination.
2. Samples or measurements are taken on a square, rectangular, or triangular grid. Figure 2 illustrates the various grid configurations.
3. The distance between grid points is much larger than the size of the sample being measured or cored at grid points; that is, a very small portion of the area being studied can actually be measured.
4. The definition of a hot spot is clear and unambiguous.
5. There are no measurement misclassification errors; that is, no errors are made in deciding when a hot spot has been detected.

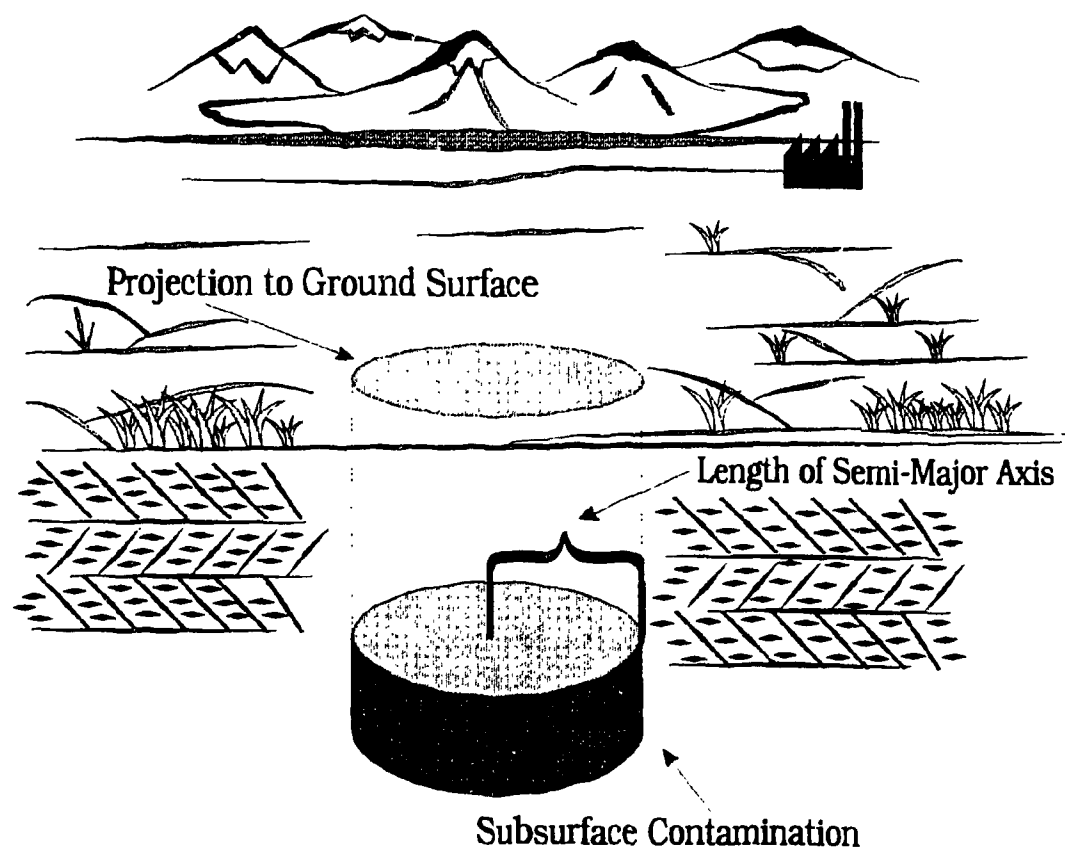


Fig. 1. Hypothetical subsurface pocket of contamination.

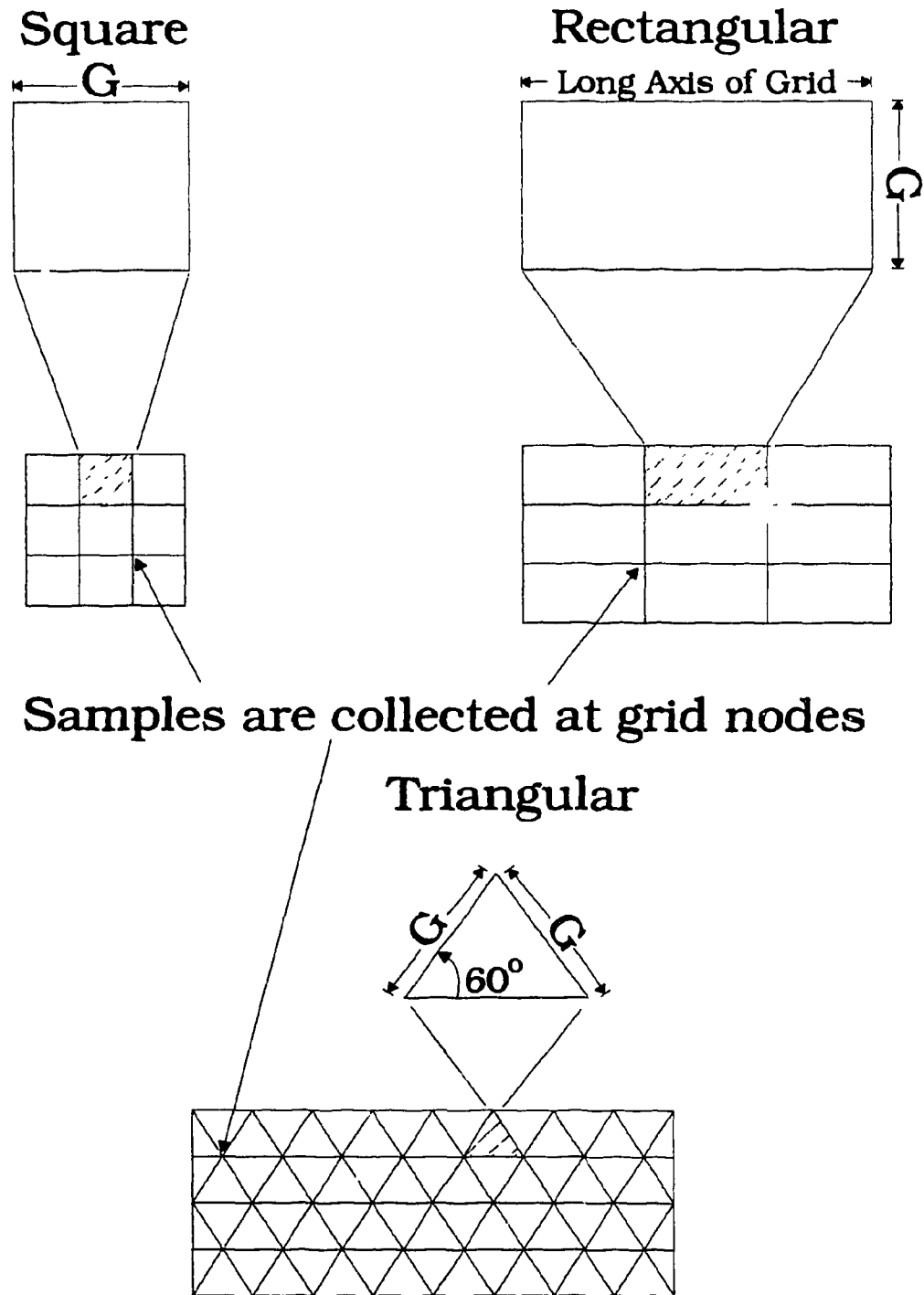


Fig. 2. Grid configuration for finding hot spots.

4. PROGRAM DESCRIPTIONS

4.1 ELIPGRID-1

ELIPGRID-1 is a PC program, written in Lahey FORTRAN, that closely conforms to the original ELIPGRID FORTRAN code structure. It was written to demonstrate that ELIPGRID code could work on a PC. The format for data input is the same as the original ELIPGRID punch-card format. The program does not provide any user-interface features other than a simple help screen and various messages relating to data input file errors.

ELIPGRID-1 contains the original algorithm used by the RECT subroutine in the published version of ELIPGRID. However, the output from ELIPGRID-1 does not match the published output for a number of rectangular grid cases (Singer 1972). These discrepancies revealed the need to modify the RECT subroutine that resulted in the ELIPGRID-2 program. See Appendix A for a demonstration of the nonequivalence of the original ELIPGRID FORTRAN code and Singer and Wickman's mathematical development.

Hardware requirements for the program include an IBM® PC (or compatible) with an Intel® 385™, i486™, or Pentium™ central processing unit, with a minimum of 512 kilobytes (KB) free random access memory (RAM) recommended. Additionally, a math co-processor is required and a fixed hard disk drive is recommended.

4.2 ELIPGRID-2

ELIPGRID-2 is essentially the same program as ELIPGRID-1, with the key difference being the modified RECT subroutine. With this modification in place, ELIPGRID-2 is able to reproduce the results of the published data (Singer 1972). See Appendix B for

comparisons of ELIPGRID-1, ELIPGRID-2, and ELIPGRID-PC output against Singer's published data. The source code for ELIPGRID-2 is found in Appendix D.

The hardware requirements for ELIPGRID-2 are the same as those for ELIPGRID-1.

4.3 ELIPGRID-PC

ELIPGRID-PC is a new program incorporating the corrected version of the ELIPGRID algorithm found in ELIPGRID-2. Although the algorithm was recoded into CA-Clipper® for ELIPGRID-PC, no changes were made to the underlying mathematical algorithm (Appendix C documents an algorithmic substitution required for a small portion of ELIPGRID's triangular grid computations). The source code for ELIPGRID-PC is found in Appendix E. Source code for a simple graphics program to display and print the output from the "Write Cost-Based Graph Data" option is found in Appendix F.

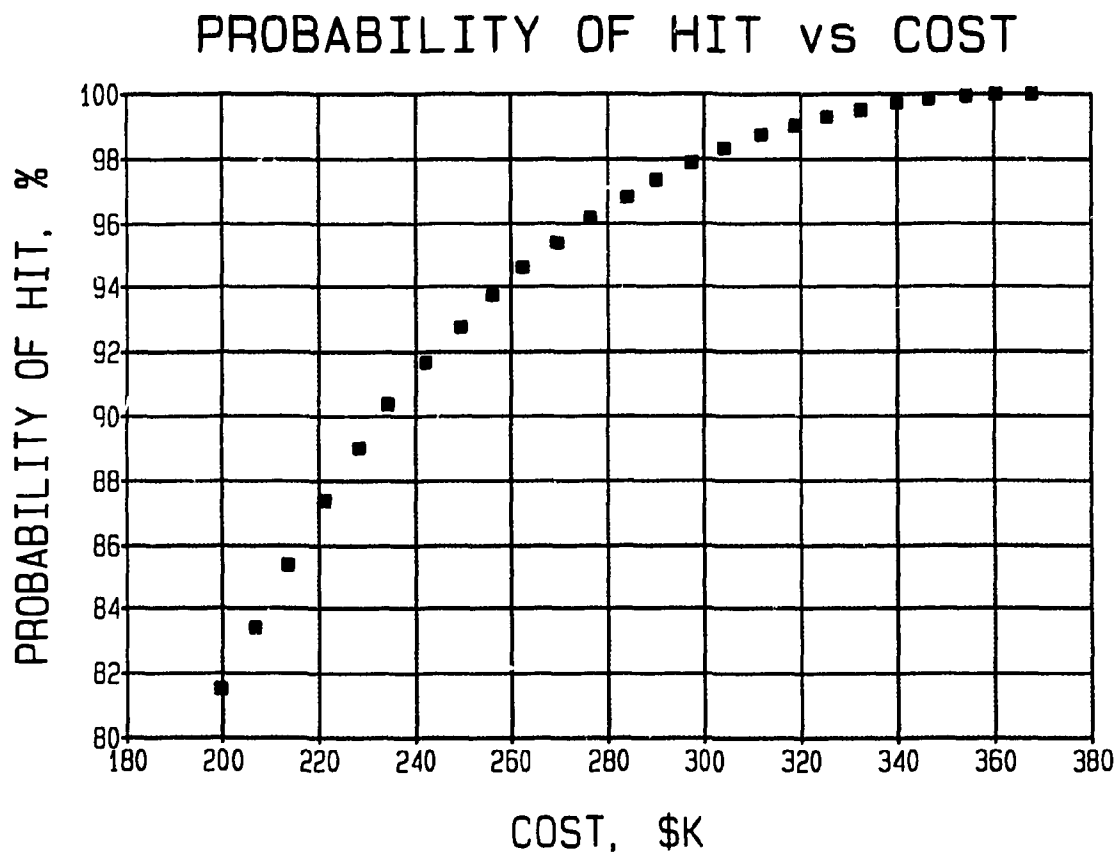
ELIPGRID-PC provides some output features not directly available in ELIPGRID:

- ELIPGRID-PC calculates a grid size, given the desired probability of detecting a specified hot spot.
- ELIPGRID-PC calculates an approximate grid size, given desired cost and hot-spot specifications. Note that this is an approximate grid size since the underlying EPA formula for determining the number of samples for a given area is itself approximate (U.S.EPA 1989).
- ELIPGRID-PC calculates the smallest hot spot that can be detected with a given probability and grid size.

- **ELIPGRID-PC** provides the capability for graphing with spreadsheets or graphics software the probability of detection versus cost. Figure 3 is an example of this for a square grid.

Designed to be user-friendly, **ELIPGRID-PC** includes the following features:

- A simplified input format (SIF) file option. SIF files provide an easier-to-use input file structure than the **ELIPGRID** format input files.
- Screen input and output in either meters or feet.
- Conversion from acres to m^2 or to ft^2 using the F10 key. The program also calculates the length of the hot spot semi-major axis from the area of the hot spot.
- Change of the basic unit of length from meters to feet as a command-line option using the letter F. Command-line option M will force a monochrome screen, and command-line option H provides usage information.
- Input and output files located on any drive and subdirectory.
- Temporary exit to DOS. DOS commands or other programs may then be executed.



Input File: C:\CLIPPER2\EDITOR\EGPC\Graph.Dat
 Print Date: 08/26/94
 Print Time: 1:17:54 pm
 Grid type chosen.....: Square
 Shape of the elliptical hot spot: 0.80
 Length of semi-major axis.....: 3.15 m
 Angle of orientation to grid.....: 0.0°
 Total area to sample.....: 8093.0 m²
 Individual sample cost.....\$: 700.00

Fig. 3. Probability of hit vs total sample cost for a square grid.

5. INSTALLATION EXAMPLE

This example is for the ELIPGRID-PC program being copied to a C: fixed hard drive. ELIPGRID-2 would be installed in a similar manner.

- (1) Make a new subdirectory on the hard drive, for example:

```
C:\>MD ELIPGRID
```

- (2) Change to the new subdirectory:

```
C:\>CD ELIPGRID
```

- (3) Copy all files from the source floppy disk in A: or B: drive:

```
C:\ELIPGRID>COPY A:*. *   or   COPY B:*. *
```

- (4) Run the ELIPGRID program. EGPC is the executable file name used in these examples:

Using defaults of a color screen and meters for basic unit of length,

```
C:\ELIPGRID>EGPC
```

With the basic unit set to feet,

```
C:\ELIPGRID>EGPC F
```

To display a command-line parameter help screen,

```
C:\ELIPGRID>EGPC H .
```

6. SUMMARY

Singer and Wickman's (1969) ELIPGRID algorithm for calculating hot-spot sampling probabilities has been successfully made available to the PC environment. ELIPGRID-PC provides the algorithm in CA-Clipper®-compatible format. The program additionally calculates the grid size required for specified conditions, the smallest hot spot that can be sampled with a given probability, and the approximate grid size resulting from specified conditions and sampling cost. ELIPGRID-PC also provides probability of detection versus cost data for graphing with spreadsheets or graphics software.

ELIPGRID-PC has been successfully tested using Singer's published ELIPGRID results and includes corrections to the rectangular and triangular grid routines of the original *ELIPGRID*.

REFERENCES

- Gilbert, R. O. 1987. *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York.
- Singer, D. A. 1972. ELIPGRID, a FORTRAN IV program for calculating the probability of success in locating elliptical targets with square, rectangular, and hexagonal grids. *Ceocom Programs*, 4:1-16.
- Singer, D. A. 1994. Letter to J. R. Davidson, Oak Ridge National Laboratory, Grand Junction, Colorado, from U.S. Geological Survey, Menlo Park, California, March 11, 1994.
- Singer, D. A., and F. E. Wickman. 1969. *Probability Tables for Locating Elliptical Targets with Square, Rectangular and Hexagonal Point Nets*. Pennsylvania State University, University Park, Pennsylvania.
- U S.EPA. 1989. *Methods for Evaluating the Attainment of Cleanup Standards. Volume 1. Soils and Solid Media*. EPA/230/02-89/042. U. S. Environmental Protection Agency.
- Zirschky, J., and R. O. Gilbert. 1984. Detecting hot spots at hazardous-waste sites. *Chemical Engineering*, 91:97-100.

APPENDIX A

DEMONSTRATION OF NONEQUIVALENCE OF ELIPGRID FORTRAN CODE AND SINGER AND WICKMAN'S EQUATION

DEMONSTRATION OF NONEQUIVALENCE OF ELIPGRID FORTRAN CODE AND SINGER AND WICKMAN'S EQUATION

For this demonstration, assume that the ELIPGRID FORTRAN code equation is equivalent to the mathematical equation it is based on. By deriving a logical contradiction, it is demonstrated that the ELIPGRID code is not equivalent to the mathematical equation.

The RECT subroutine in ELIPGRID transforms a rectangular grid and elliptical target to a square grid and transformed elliptical target using an affine transformation described in detail by Singer and Wickman (1969). The angle of the transformed elliptical target to the transformed grid is found from Singer and Wickman's Eq. (35),

$$\tan 2\gamma = \frac{2q(1 - k) \tan \alpha}{1 - k^2 q^2 - (q^2 - k^2) \tan^2 \alpha}, \quad (1)$$

where

- $\gamma =$ the angle of the transformed target to the transformed grid, if $\tan 2\gamma \geq 0$; if $\tan 2\gamma < 0$ the angle is $90^\circ - |\gamma|$;
- $\alpha =$ the angle of the original target to the original grid;
- $q =$ the shape of the rectangular grid (long side divided by short side);
- $k =$ the shape of the original elliptical target (semi-minor axis of the target divided by semi-major axis).

The relevant ELIPGRID FORTRAN code is found on lines RECT 175 and RECT 185 (Singer 1972):

$$REVANG = (ATAN (2.0 * Q * (1.0 - SQK) * TAN (ALPHA) / ((AQ - SQK) * TAN (ALPHA)**2 * TIS - 1.0)) / 2.0) * 57.295779,$$

where

$REVANG = \gamma$ in Eq. (1);

$AQ = Q^2$ and Q is the shape of the rectangular grid (long side divided by the short side), q in Eq. (1);

$SQK = Shape^2$ and $Shape$ is the semi-minor axis of the target divided by the semi-major axis, k in Eq. (1);

$TIS = AQ * SQK$ or $k^2 q^2$ in Eq. (1);

$ALPHA =$ the angle of orientation of the original target to the original grid, α in Eq. (1);

$57.295779 =$ the conversion factor for radians to degrees.

Replace ELIPGRID's symbols with the equivalent symbols used in Eq. (1), remove the radians-to-degrees conversion factor, and solve for $\tan 2\gamma$, so that the ELIPGRID code in mathematical form becomes

$$\tan 2\gamma = \frac{2q(1 - k^2) \tan \alpha}{(q^2 - k^2) \tan^2 \alpha + k^2 q^2 - 1} . \quad (2)$$

A comparison of Eq. (2) with Eq. (1) reveals two differences. First, the numerator of the ELIPGRID equation has the term $(1 - k^2)$, while Eq. (1) has the term $(1 - k)$. Second, the denominators have their terms in different orders with different signs. It is possible that Eq. (2) is an equivalent form of Eq. (1). Assume they are equivalent and derive a logical contradiction.

First, to simplify both equations, make the following substitutions:

$$A = (1 - k),$$

$$B = 1 - k^2 q^2,$$

$$C = (q^2 - k^2) \tan^2 \alpha .$$

Equation (1) becomes

$$\tan 2\gamma = \frac{2qA \tan \alpha}{B - C} .$$

Equation (2) becomes

$$\tan 2\gamma = \frac{2q(1 - k^2) \tan \alpha}{C - B} .$$

However, $(1 - k^2) = (1 + k)(1 - k) = (1 + k)A$, so Eq. (2) can be written as

$$\tan 2\gamma = \frac{2q(1 + k)A \tan \alpha}{C - B} .$$

Now assume both equations are equivalent, as they should be if the ELIPGRID code matches Eq. (1):

$$\frac{2qA \tan \alpha}{B - C} = \frac{2q(1 + k)A \tan \alpha}{C - B} .$$

The term $2qA \tan \alpha$ is common to both fractions. Divide both sides by this term if it is not equal to zero. For $q > 0$, $\alpha > 0$, and $A > 0$, this term will be greater than zero. Note that $A = 1 - k$. Therefore, if $k < 1$, then $A > 0$. Dividing both sides by $(2qA \tan \alpha)$ results in

$$\frac{1}{B - C} = \frac{(1 + k)}{C - B} .$$

Cross-multiply to yield

$$C - B = (1 + k)(B - C) .$$

Perform the indicated multiplication on the right:

$$C - B = B + Bk - C - Ck .$$

Add $(B - C)$ to both sides to give

$$0 = 2B + Bk - 2C - Ck .$$

Factor the right hand side to yield

$$0 = B(2 + k) - C(2 + k) .$$

If $k > 0$, as it must be for any actual elliptical target, then $2 + k > 0$. Therefore, divide both sides by $(2 + k)$ to give

$$0 = B - C .$$

Swapping both sides by the symmetry axiom of algebra yields

$$B - C = 0 .$$

Replacing B and C by their original values gives

$$1 - k^2 q^2 - (q^2 - k^2) \tan^2 \alpha = 0 .$$

However, the left hand side above is the *denominator* in Eq. (1) and cannot be 0 for the large number of legitimate cases where $q > 0$, $\alpha > 0$, and $0 < k < 1$. Therefore, the premise of equivalence is false.

An example case would be $q = 2$, $\alpha = 45^\circ$, and $k = 0.7$. This is the case of a rectangular grid with shape 2, target orientation angle of 45° , and target elliptical shape of 0.7. Many other such cases could be found.

Dr. Donald A. Singer, author of the ELIPGRID program, agrees with the above conclusion. His replies to the following questions appear below (Singer 1994).

- 1) It appears that the term $(1 - k)$ in the numerator of Eq. (1) [Eq. (35), Singer and Wickman 1969] should be $(1 - k^2)$. This term is given as $(1.0 - SQK)$ in the code on line RECT 175 (Singer 1972). SQK is $Shape^2$, and $Shape$ is the ELIPGRID code variable for k . Question 1: could the blank space after k in the Singer and Wickman paper imply a superscript 2 was left out by accident?
- 2) The denominator of Eq. (1) differs from the code in the order in which the terms are listed and in their signs. Question 2: could the code denominator be in error?

Question 1. "I agree with you that the blank space after k in Equation 35 in Singer and Wickman (1969) is clearly meant for a missing superscript 2. This is consistent with all other equations in Singer and Wickman and with published computer code in Singer (1972)."

Question 2. "Yes, the computer code for Equation 35 that you have used in HOTSPOT and ELIPGRID2 is consistent with the output in Singer (1972) and (based on spot checks) with the tables in Singer and Wickman."

APPENDIX B

TESTING PROCEDURE

TESTING PROCEDURE

B.1 Singer's Tables

The goal of the ELIPGRID-2 and ELIPGRID-PC programs is to duplicate for a PC the ELIPGRID program. Once the PC program was written, Singer's Table 1, input data, and Table 2, output data (Singer 1972), were used as the basis for testing the program.

A comparison of the 100 input cases in Table 1 with the 100 output cases in Table 2 reveals some obvious problems in the sequential arrangement of the data in the tables. Apparently, the original computer printouts were accidentally disordered in the paste-up process. Fortunately, a careful comparison of Table 1 to Table 2 provides a key to the correct sequential order.

Table B.1 Corrected sequential order of Singer's tables

	Table 1	Table 2
Rows	1-33	1-33
Rows	34-51	65-82
Rows	52-82	34-64
Rows	83-100	83-100

Dr. Donald A. Singer, author of the paper containing the tables in question, agrees with this conclusion. "Yes, the entries of the output table in Singer (1972) are in a different order than in the input table. Fortunately, the output table provides the input values so that the match can be made correctly as you have done" (Singer 1994).

All 100 of Singer's cases were tested using the corrected sequential order listed in Table B.1.

B.2 ELIPGRID-1 RESULTS

Table B.2 is a listing of the input file, TEST100.IN, used to duplicate Singer's Table 1. Table B.3 is a listing of the output file, TEST100.EG1, produced from running ELIPGRID-1 on TEST100.IN.

All ELIPGRID-1 square and triangular grid output values matched Singers's Table 2 output values. A match is defined as two output values that differ by no more than ± 0.0001 . This corresponds to a difference in the probability of not hitting the target of $\pm 0.01\%$.

The ELIPGRID-1 output from 10 rectangular grid cases, out of 30 total, did not match the published output in Singer's Table 2. The inability of the program to match rectangular grid cases led to a review of the RECT subroutine in the published ELIPGRID code. See Appendix A for a demonstration of the nonequivalence of the published ELIPGRID code and Singer and Wickman's original mathematical development. Since the original ELIPGRID code could not reproduce the published results, a modified RECT subroutine was developed. ELIPGRID-2 is essentially ELIPGRID-1 with the exception of the modified RECT subroutine.

B.3 ELIPGRID-2 RESULTS

File TEST100.IN was used for input data to ELIPGRID-2. Table B.4 is a listing of the output file, TEST100.EG2, produced from running ELIPGRID-2 on TEST100.IN. All square, triangular, and rectangular grid output values from ELIPGRID-2 matched Singers's Table 2 output values.

B-3

Table B.2. Input file listing after style used in ELIPGRID

File: C:\CLIPPER2\EDITOR\HOTSPOT\VALID100\TEST100.IN Print Date: 08/10/94

Page: 1

Test100.in input test file for ELIPGRD1, 2, M, and HOTSPOT, 02/03/94.

1000.0	0.38	22.0	800.0	1	0#261
1250.0	0.30	6.0	800.0	1	0#187
1250.0	0.50	38.0	800.0	1	0#190
300.0	0.25	24.0	800.0	1	0#147
625.0	0.50	35.0	800.0	1	0#10
875.0	0.31	7.0	800.0	1	0#19
625.0	0.20	18.0	800.0	1	0#26
125.0	0.50	24.0	800.0	1	0#30
1625.0	0.15	11.0	800.0	1	0#49
1250.0	0.50	0.0	800.0	1	0#104
1000.0	0.38	22.0	1000.0	1	0#261
1250.0	0.30	6.0	1000.0	1	0#187
1250.0	0.50	38.0	1000.0	1	0#190
300.0	0.25	24.0	1000.0	1	0#147
625.0	0.50	35.0	1000.0	1	0#10
875.0	0.31	7.0	1000.0	1	0#19
625.0	0.20	18.0	1000.0	1	0#26
125.0	0.50	24.0	1000.0	1	0#30
1625.0	0.15	11.0	1000.0	1	0#49
1250.0	0.50	0.0	1000.0	1	0#104
1000.0	0.38	22.0	1500.0	1	0#261
1250.0	0.30	6.0	1500.0	1	0#187
1250.0	0.50	38.0	1500.0	1	0#190
300.0	0.25	24.0	1500.0	1	0#147
625.0	0.50	35.0	1500.0	1	0#10
875.0	0.31	7.0	1500.0	1	0#19
625.0	0.20	18.0	1500.0	1	0#26
125.0	0.50	24.0	1500.0	1	0#30
1625.0	0.15	11.0	1500.0	1	0#49
1250.0	0.50	0.0	1500.0	1	0#104
1000.0	0.38	22.0	859.66	2	0#261
1250.0	0.30	6.0	859.66	2	0#187
1250.0	0.50	22.0	859.66	2	0#190
625.0	0.50	35.0	565.69	3	0#10
2.0					
875.0	0.31	7.0	565.69	3	0#19
2.0					
625.0	0.20	18.0	565.69	3	0#26
2.0					
125.0	0.50	24.0	565.69	3	0#30
2.0					
1625.0	0.15	11.0	565.69	3	0#49
2.0					
1250.0	0.50	0.0	565.69	3	0#104
2.0					
1000.0	0.38	22.0	707.11	3	0#261
2.0					
1250.0	0.30	6.0	707.11	3	0#187
2.0					
1250.0	0.50	38.0	707.11	3	0#190
2.0					
300.0	0.25	66.0	707.11	3	0#147
2.0					
625.0	0.50	35.0	707.11	3	0#10
2.0					
875.0	0.31	7.0	707.11	3	0#19
2.0					
625.0	0.20	18.0	707.11	3	0#26
2.0					
125.0	0.50	24.0	707.11	3	0#30
2.0					
1625.0	0.15	11.0	707.11	3	0#49
2.0					

B-4

Table B.2. (cont.)

File: C:\CLIPPER2\EDITOR\HOTSPOT\VALID100\TEST100.IN

Print Date: 03/10/94

Page: 2

1250.0	0.50	0.0	707.11	3	0#104
2.0					
1000.0	0.38	22.0	1060.66	3	0#261
2.0					
1250.0	0.30	6.0	1060.66	3	0#187
2.0					
300.0	0.25	6.0	859.66	2	0#147
625.0	0.50	25.0	859.66	2	0#10
875.0	0.31	7.0	859.66	2	0#19
625.0	0.20	18.0	859.66	2	0#26
125.0	0.50	24.0	859.66	2	0#30
1625.0	0.15	11.0	859.66	2	0#49
1250.0	0.50	0.0	859.66	2	0#104
1000.0	0.38	22.0	1074.57	2	0#261
1250.0	0.30	6.0	1074.57	2	0#187
1250.0	0.50	22.0	1074.57	2	0#190
300.0	0.25	6.0	1074.57	2	0#147
625.0	0.50	25.0	1074.57	2	0#10
875.0	0.31	7.0	1074.57	2	0#19
625.0	0.2	18.0	1074.57	2	0#26
125.0	0.50	24.0	1074.57	2	0#30
1625.0	0.15	11.0	1074.57	2	0#49
1250.0	0.50	0.0	1074.57	2	0#104
1000.0	0.38	22.0	1611.86	2	0#261
1250.0	0.30	6.0	1611.86	2	0#187
1250.0	0.50	22.0	1611.86	2	0#190
300.0	0.25	6.0	1611.86	2	0#147
625.0	0.50	25.0	1611.86	2	0#10
875.0	0.31	7.0	1611.86	2	0#19
625.0	0.20	18.0	1611.86	2	0#26
125.0	0.50	24.0	1611.86	2	0#30
1625.0	0.15	11.0	1611.86	2	0#49
1250.0	0.50	0.0	1611.86	2	0#104
1000.0	0.38	22.0	565.69	3	0#261
2.0					
1250.0	0.30	6.0	565.69	3	0#187
2.0					
1250.0	0.50	38.0	565.69	3	0#190
2.0					
300.0	0.25	66.0	565.69	3	0#147
2.0					
1250.0	0.50	38.0	1060.66	3	0#190
2.0					
300.0	0.25	66.0	1060.66	3	0#147
2.0					
625.0	0.50	35.0	1060.66	3	0#10
2.0					
875.0	0.31	7.0	1060.66	3	0#19
2.0					
625.0	0.20	18.0	1060.66	3	0#26
2.0					
125.0	0.50	24.0	1060.66	3	0#30
2.0					
1625.0	0.15	11.0	1060.66	3	0#49
2.0					
1250.0	0.50	0.0	1060.66	3	0#104
2.0					
1000.0	0.38	22.0	1000.0	1	1#261
1250.0	0.30	6.0	1000.0	1	1#187
1250.0	0.50	38.0	1000.0	1	1#190
300.0	0.25	24.0	1000.0	1	1#147
625.0	0.50	35.0	1000.0	1	1#10
875.0	0.31	7.0	1000.0	1	1#19
625.0	0.20	18.0	1000.0	1	1#26

B-5

Table B.2. (cont.)

File: C:\CLIP\PER2\EDITOR\HOTSPOT\VALID100\TEST100.IN					Print Date: 08/10/94	Page: 3
125.0	0.50	24.0	1000.0	1	1#30	
1625.0	0.15	11.0	1000.0	1	1#49	
1250.0	0.50	0.0	1000.0	1	1#104	
9.9	9.9	9.9	9.9	9	9 EOF	

B-6

Table B. 3. ELIPGRID-1 output file listing

ELIPGRD1 Output File

Data from: Test100.in input test file for ELIPGRD1, 2, M, and NOTSPOT, 02/03/94.

TARGET	GRID TYPE	SEMINAJOR AXIS IN RELATIVE UNITS	GRIDSPACE IN ORIG UNITS	SHAPE	ANGLE	PROB(1)	PROB(>1)	PROB(0)	
#261	SQUARE	1.25	800.00	0.38	22.0	9.0000	9.0000	0.0000	****
#187	SQUARE	1.56	800.00	0.30	6.0	0.1241	0.8448	0.0311	
#190	SQUARE	1.56	800.00	0.50	38.0	9.0000	9.0000	0.0000	****
#147	SQUARE	0.38	800.00	0.25	24.0	0.1104	0.0000	0.8896	
#10	SQUARE	0.78	800.00	0.50	35.0	0.8600	0.0494	0.0906	
#19	SQUARE	1.09	800.00	0.31	7.0	0.2283	0.4658	0.3059	
#26	SQUARE	0.78	800.00	0.20	18.0	0.3835	0.0000	0.6165	
#30	SQUARE	0.16	800.00	0.50	24.0	0.0383	0.0000	0.9617	
#49	SQUARE	2.03	800.00	0.15	11.0	0.1775	0.7755	0.0470	
#104	SQUARE	1.56	800.00	0.50	0.1	9.0000	9.0000	0.0000	****
#261	SQUARE	1.00	1000.00	0.38	22.0	0.6825	0.2557	0.0619	
#187	SQUARE	1.25	1000.00	0.30	6.0	0.1672	0.5990	0.2337	
#190	SQUARE	1.25	1000.00	0.50	38.0	9.0000	9.0000	0.0000	****
#147	SQUARE	0.30	1000.00	0.25	24.0	0.0707	0.0000	0.9293	
#10	SQUARE	0.63	1000.00	0.50	35.0	0.6136	0.0000	0.3864	
#19	SQUARE	0.88	1000.00	0.31	7.0	0.3362	0.2047	0.4591	
#26	SQUARE	0.63	1000.00	0.20	18.0	0.2454	0.0000	0.7546	
#30	SQUARE	0.13	1000.00	0.50	24.0	0.0245	0.0000	0.9755	
#49	SQUARE	1.63	1000.00	0.15	11.0	0.2576	0.4925	0.2499	
#104	SQUARE	1.25	1000.00	0.50	0.1	9.0000	9.0000	0.0000	****
#261	SQUARE	0.67	1500.00	0.38	22.0	0.5306	0.0000	0.4694	
#187	SQUARE	0.83	1500.00	0.30	6.0	0.3241	0.1652	0.5107	
#190	SQUARE	0.83	1500.00	0.50	38.0	0.8560	0.1174	0.0266	
#147	SQUARE	0.20	1500.00	0.25	24.0	0.0314	0.0000	0.9686	
#10	SQUARE	0.42	1500.00	0.50	35.0	0.2727	0.0000	0.7273	
#19	SQUARE	0.58	1500.00	0.31	7.0	0.3119	0.0097	0.6783	
#26	SQUARE	0.42	1500.00	0.20	18.0	0.1091	0.0000	0.8909	
#30	SQUARE	0.08	1500.00	0.50	24.0	0.0109	0.0000	0.9891	
#49	SQUARE	1.08	1500.00	0.15	11.0	0.3856	0.0837	0.5307	
#104	SQUARE	0.83	1500.00	0.50	0.1	0.4696	0.3106	0.2198	
#261	HEXAGONAL	1.16	859.66	0.38	22.0	9.0000	9.0000	0.0000	****
#187	HEXAGONAL	1.45	859.66	0.30	6.0	9.0000	9.0000	0.0000	****
#190	HEXAGONAL	1.45	859.66	0.50	22.0	9.0000	9.0000	0.0000	****
#10	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	1.10	565.69	0.50	35.0	0.8367	0.0610	0.1023
#19	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	1.55	565.69	0.31	7.0	0.6256	0.2697	0.1047
#26	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	1.10	565.69	0.20	18.0	0.3835	0.0000	0.6165
#30	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	0.22	565.69	0.50	24.0	0.0383	0.0000	0.9617
#49	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	2.87	565.69	0.15	11.0	9.0000	9.0000	0.0000
#104	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	2.21	565.69	0.50	0.0	9.0000	9.0000	0.0000
#261	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	1.11	707.11	0.38	22.0	0.7698	0.2120	0.0182
#187	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	1.77	707.11	0.30	6.0	0.5132	0.4797	0.0071
#190	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	1.77	707.11	0.50	38.0	9.0000	9.0000	0.0000
#147	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	0.42	707.11	0.25	66.0	0.0707	0.0000	0.9293
#10	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	0.88	707.11	0.50	35.0	0.6136	0.0000	0.3864
#19	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	1.24	707.11	0.31	7.0	0.6095	0.0681	0.3224
#26	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	0.88	707.11	0.20	18.0	0.2454	0.0000	0.7546
#30	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	0.18	707.11	0.50	24.0	0.0245	0.0000	0.9755
#49	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	2.30	707.11	0.15	11.0	0.6090	0.3177	0.0733
#104	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	1.77	707.11	0.50	0.0	9.0000	9.0000	0.0000
#261	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	0.94	1060.66	0.38	22.0	0.5306	0.0000	0.4694
#187	RECTANGULAR	X AXIS OF GRID= 2.0 Y AXIS	1.18	1060.66	0.30	6.0	0.5723	0.0411	0.3866
#147	HEXAGONAL		0.35	859.66	0.25	6.0	0.1104	0.0000	0.8896
#10	HEXAGONAL		0.73	859.66	0.50	25.0	0.8187	0.0700	0.1113

B-7

Table B.3. (cont.)

#19	HEXAGONAL	1.02	859.66	0.31	7.0	0.3162	0.4244	0.2594	
#26	HEXAGONAL	0.73	859.66	0.20	18.0	0.3835	0.0000	0.6165	
#30	HEXAGONAL	0.15	859.66	0.50	24.0	0.0383	0.0000	0.9617	
#49	HEXAGONAL	1.89	859.66	0.15	11.0	0.1836	0.8141	0.0023	
#104	HEXAGONAL	1.45	859.66	0.50	0.1	9.0000	9.0000	0.0000	****
#261	HEXAGONAL	0.93	1074.57	0.38	22.0	0.7795	0.2072	0.0134	
#187	HEXAGONAL	1.16	1074.57	0.30	6.0	0.2162	0.6030	0.1807	
#190	HEXAGONAL	1.16	1074.57	0.50	22.0	9.0000	9.0000	0.0000	****
#147	HEXAGONAL	0.28	1074.57	0.25	6.0	0.0707	0.0000	0.9293	
#10	HEXAGONAL	0.58	1074.57	0.50	25.0	0.6136	0.0000	0.3864	
#19	HEXAGONAL	0.81	1074.57	0.31	7.0	0.4030	0.1713	0.4257	
#26	HEXAGONAL	0.58	1074.57	0.20	18.0	0.2454	0.0000	0.7546	
#30	HEXAGONAL	0.12	1074.57	0.50	24.0	0.0245	0.0000	0.9755	
#49	HEXAGONAL	1.51	1074.57	0.15	11.0	0.3561	0.4441	0.1998	
#104	HEXAGONAL	1.16	1074.57	0.50	0.1	9.0000	9.0000	0.0000	****
#261	HEXAGONAL	0.62	1611.86	0.38	22.0	0.5306	0.0000	0.4694	
#187	HEXAGONAL	0.78	1611.86	0.30	6.0	0.3834	0.1355	0.4810	
#190	HEXAGONAL	0.78	1611.86	0.50	22.0	0.7738	0.1585	0.0677	
#147	HEXAGONAL	0.19	1611.86	0.25	6.0	0.0314	0.0000	0.9686	
#10	HEXAGONAL	0.39	1611.86	0.50	25.0	0.2727	0.0000	0.7273	
#19	HEXAGONAL	0.54	1611.86	0.31	7.0	0.3297	0.0009	0.6695	
#26	HEXAGONAL	0.39	1611.86	0.20	18.0	0.1091	0.0000	0.8909	
#30	HEXAGONAL	0.08	1611.86	0.50	24.0	0.0109	0.0000	0.9891	
#49	HEXAGONAL	1.01	1611.86	0.15	11.0	0.4353	0.0589	0.5058	
#104	HEXAGONAL	0.78	1611.86	0.50	0.1	0.5669	0.2620	0.1712	
#261	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.77	565.69	0.38	22.0	9.0000	9.0000	0.0000	****
#187	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	2.21	565.69	0.30	6.0	9.0000	9.0000	0.0000	****
#190	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	2.21	565.69	0.50	38.0	9.0000	9.0000	0.0000	****
#147	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.53	565.69	0.25	66.0	0.1104	0.0000	0.8896	
#190	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.18	1060.66	0.50	38.0	0.7882	0.1513	0.0605	
#147	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.28	1060.66	0.25	66.0	0.0314	0.0000	0.9686	
#10	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.59	1060.66	0.50	35.0	0.2727	0.0000	0.7273	
#19	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.82	1060.66	0.31	7.0	0.3314	0.0000	0.6686	
#26	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.59	1060.66	0.20	18.0	0.1091	0.0000	0.8909	
#30	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.12	1060.66	0.50	24.0	0.0109	0.0000	0.9891	
#49	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.53	1060.66	0.15	11.0	0.5356	0.0087	0.4557	
#104	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.18	1060.66	0.50	0.0	0.7892	0.1508	0.0600	
#261	SQUARE	1.00	1000.00	0.38	RANDOM	0.5789	0.3075	0.1137	
#187	SQUARE	1.25	1000.00	0.30	RANDOM	3.3562	3.4839	0.0730	****
#190	SQUARE	1.25	1000.00	0.50	RANDOM	9.0000	9.0000	0.0000	****
#147	SQUARE	0.30	1000.00	0.25	RANDOM	0.0707	0.0000	0.9293	
#10	SQUARE	0.63	1000.00	0.50	RANDOM	0.5719	0.0208	0.4072	
#19	SQUARE	0.88	1000.00	0.31	RANDOM	0.5597	0.0930	0.3474	
#26	SQUARE	0.43	1000.00	0.20	RANDOM	0.2393	0.0031	0.7577	
#30	SQUARE	0.13	1000.00	0.50	RANDOM	0.0245	0.0000	0.9755	
#49	SQUARE	1.23	1000.00	0.15	RANDOM	0.4427	0.3738	0.1835	
#104	SQUARE	1.25	1000.00	0.50	RANDOM	9.0000	9.0000	0.0000	****

END OF RUN (OR ERROR IN SHAPE)

**** INDICATES THAT THE PROBABILITY OF MISSING IS ZERO FOR AT LEAST ONE
ORIENTATION AND PROB1 AND PROB>1 SHOULD NOT BE USED FOR THIS TARGET

B-8

Table B.4. ELIPGRID-2 output file listing

ELIPGRD2 Output File

Data from: Test100.In input test file for ELIPGRD1, 2, M, and HOTSPOT, 02/03/94.

TARGET	GRID TYPE	SEMI-MAJOR AXIS IN RELATIVE UNITS	GRIDSPACE IN ORIG UNITS	SHAPE	ANGLE	PROB(1)	PROB(>1)	PROB(0)	
#261	SQUARE	1.25	800.00	0.38	22.0	9.0000	9.0000	0.0000	****
#187	SQUARE	1.56	800.00	0.30	6.0	0.1241	0.8448	0.0311	
#190	SQUARE	1.56	800.00	0.50	38.0	9.0000	9.0000	0.0000	****
#147	SQUARE	0.38	800.00	0.25	24.0	0.1104	0.0000	0.8896	
#10	SQUARE	0.78	800.00	0.50	35.0	0.8600	0.0494	0.0906	
#19	SQUARE	1.09	800.00	0.31	7.0	0.2283	0.4658	0.3059	
#26	SQUARE	0.78	800.00	0.20	18.0	0.3835	0.0000	0.6165	
#30	SQUARE	0.16	800.00	0.50	24.0	0.0383	0.0000	0.9617	
#49	SQUARE	2.03	800.00	0.15	11.0	0.1775	0.7755	0.0470	
#104	SQUARE	1.56	800.00	0.50	0.1	9.0000	9.0000	0.0000	****
#261	SQUARE	1.00	1000.00	0.38	22.0	0.6825	0.2557	0.0619	
#187	SQUARE	1.25	1000.00	0.30	6.0	0.1672	0.5990	0.2337	
#190	SQUARE	1.25	1000.00	0.50	38.0	9.0000	9.0000	0.0000	****
#147	SQUARE	0.30	1000.00	0.25	24.0	0.0707	0.0000	0.9293	
#10	SQUARE	0.63	1000.00	0.50	35.0	0.6136	0.0000	0.3864	
#19	SQUARE	0.88	1000.00	0.31	7.0	0.3362	0.2047	0.4591	
#26	SQUARE	0.63	1000.00	0.20	18.0	0.2454	0.0000	0.7546	
#30	SQUARE	0.13	1000.00	0.50	24.0	0.0245	0.0000	0.9755	
#49	SQUARE	1.63	1000.00	0.15	11.0	0.2576	0.4925	0.2499	
#104	SQUARE	1.25	1000.00	0.50	0.1	9.0000	9.0000	0.0000	****
#261	SQUARE	0.67	1500.00	0.38	22.0	0.5306	0.0000	0.4694	
#187	SQUARE	0.83	1500.00	0.30	6.0	0.3241	0.1652	0.5107	
#190	SQUARE	0.83	1500.00	0.50	38.0	0.8560	0.1174	0.0266	
#147	SQUARE	0.20	1500.00	0.25	24.0	0.0314	0.0000	0.9686	
#10	SQUARE	0.42	1500.00	0.50	35.0	0.2727	0.0000	0.7273	
#19	SQUARE	0.58	1500.00	0.31	7.0	0.3119	0.0097	0.6783	
#26	SQUARE	0.42	1500.00	0.20	18.0	0.1091	0.0000	0.8909	
#30	SQUARE	0.08	1500.00	0.50	24.0	0.0109	0.0000	0.9891	
#49	SQUARE	1.08	1500.00	0.15	11.0	0.3856	0.0837	0.5307	
#104	SQUARE	0.83	1500.00	0.50	0.1	0.4696	0.3106	0.2198	
#261	HEXAGONAL	1.16	859.66	0.38	22.0	9.0000	9.0000	0.0000	****
#187	HEXAGONAL	1.45	859.66	0.30	6.0	9.0000	9.0000	0.0000	****
#190	HEXAGONAL	1.45	859.66	0.50	22.0	9.0000	9.0000	0.0000	****
#10	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.10	565.69	0.50	35.0	0.7376	0.1105	0.1518	
#19	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.55	565.69	0.31	7.0	0.7058	0.2296	0.0646	
#26	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.10	565.69	0.20	18.0	0.3835	0.0000	0.6165	
#30	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.22	565.69	0.50	24.0	0.0383	0.0000	0.9617	
#49	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	2.87	565.69	0.15	11.0	9.0000	9.0000	0.0000	****
#104	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	2.21	565.69	0.50	0.0	9.0000	9.0000	0.0000	****
#261	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.77	707.11	0.38	22.0	0.8018	0.1960	0.0022	
#187	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.77	707.11	0.30	6.0	9.0000	9.0000	0.0000	****
#190	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.77	707.11	0.50	38.0	9.0000	9.0000	0.0000	****
#147	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.42	707.11	0.25	66.0	0.0707	0.0000	0.9293	
#10	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.88	707.11	0.50	35.0	0.6100	0.0018	0.3882	
#19	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.24	707.11	0.31	7.0	0.6565	0.0446	0.2989	
#26	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.88	707.11	0.20	18.0	0.2454	0.0000	0.7546	
#30	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.18	707.11	0.50	24.0	0.0245	0.0000	0.9755	
#49	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	2.30	707.11	0.15	11.0	0.7376	0.2534	0.0090	
#104	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.77	707.11	0.50	0.0	9.0000	9.0000	0.0000	****
#261	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.94	1060.66	0.38	22.0	0.5306	0.0000	0.4694	
#187	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.18	1060.66	0.30	6.0	0.6013	0.0266	0.3721	
#147	HEXAGONAL	0.35	859.66	0.25	6.0	0.1104	0.0000	0.8896	
#10	HEXAGONAL	0.73	859.66	0.50	25.0	0.8187	0.0700	0.1113	

Table B.4. (cont.)

#19	HEXAGONAL	1.02	859.66	0.31	7.0	0.3162	0.4244	0.2594	
#26	HEXAGONAL	0.73	859.66	0.20	18.0	0.3835	0.0000	0.6165	
#30	HEXAGONAL	0.15	859.66	0.50	24.0	0.0383	0.0000	0.9617	
#49	HEXAGONAL	1.89	859.66	0.15	11.0	0.1836	0.8141	0.0023	
#104	HEXAGONAL	1.45	859.66	0.50	0.1	9.0000	9.0000	0.0000	****
#261	HEXAGONAL	0.93	1074.57	0.38	22.0	0.7795	0.2072	0.0134	
#187	HEXAGONAL	1.16	1074.57	0.30	6.0	0.2162	0.6030	0.1807	
#190	HEXAGONAL	1.16	1074.57	0.50	22.0	9.0000	9.0000	0.0000	****
#147	HEXAGONAL	0.28	1074.57	0.25	6.0	0.0707	0.0000	0.9293	
#10	HEXAGONAL	0.58	1074.57	0.50	25.0	0.6136	0.0000	0.3864	
#19	HEXAGONAL	0.81	1074.57	0.31	7.0	0.4030	0.1713	0.4257	
#26	HEXAGONAL	0.58	1074.57	0.20	18.0	0.2454	0.0000	0.7546	
#30	HEXAGONAL	0.12	1074.57	0.50	24.0	0.0245	0.0000	0.9755	
#49	HEXAGONAL	1.51	1074.57	0.15	11.0	0.3561	0.4441	0.1998	
#104	HEXAGONAL	1.16	1074.57	0.50	0.1	9.0000	9.0000	0.0000	****
#261	HEXAGONAL	0.62	1611.86	0.38	22.0	0.5306	0.0000	0.4694	
#187	HEXAGONAL	0.78	1611.86	0.30	6.0	0.3834	0.1355	0.4810	
#190	HEXAGONAL	0.78	1611.86	0.50	22.0	0.7738	0.1585	0.0677	
#147	HEXAGONAL	0.19	1611.86	0.25	6.0	0.0314	0.0000	0.9686	
#10	HEXAGONAL	0.39	1611.86	0.50	25.0	0.2727	0.0000	0.7273	
#19	HEXAGONAL	0.54	1611.86	0.31	7.0	0.3297	0.0009	0.6695	
#26	HEXAGONAL	0.39	1611.86	0.20	18.0	0.1091	0.0000	0.8909	
#30	HEXAGONAL	0.08	1611.86	0.50	24.0	0.0109	0.0000	0.9891	
#49	HEXAGONAL	1.01	1611.86	0.15	11.0	0.4353	0.0589	0.5058	
#104	HEXAGONAL	0.78	1611.86	0.50	0.1	0.5669	0.2620	0.1712	
#261	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.77	565.69	0.38	22.0	9.0000	9.0000	0.0000	****
#187	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	2.21	565.69	0.30	6.0	9.0000	9.0000	0.0000	****
#190	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	2.21	565.69	0.50	38.0	9.0000	9.0000	0.0000	****
#147	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.53	565.69	0.25	66.0	0.1104	0.0000	0.8896	
#190	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.18	1060.66	0.50	38.0	0.7154	0.1877	0.0969	
#147	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.28	1060.66	0.25	66.0	0.0314	0.0000	0.9686	
#10	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.59	1060.66	0.50	35.0	0.2727	0.0000	0.7273	
#19	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.82	1060.66	0.31	7.0	0.3314	0.0000	0.6686	
#26	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.59	1060.66	0.20	18.0	0.1091	0.0000	0.8909	
#30	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	0.12	1060.66	0.50	24.0	0.0109	0.0000	0.9891	
#49	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.53	1060.66	0.15	11.0	0.5531	0.0000	0.4469	
#104	RECTANGULAR X AXIS OF GRID= 2.0 Y AXIS	1.18	1060.66	0.50	0.0	0.7892	0.1508	0.0600	
#261	SQUARE	1.00	1000.00	0.38	RANDOM	0.5789	0.3075	0.1137	
#187	SQUARE	1.25	1000.00	0.30	RANDOM	3.3562	3.4839	0.0730	****
#190	SQUARE	1.25	1000.00	0.50	RANDOM	9.0000	9.0000	0.0000	****
#147	SQUARE	0.30	1000.00	0.25	RANDOM	0.0707	0.0000	0.9293	
#10	SQUARE	0.63	1000.00	0.50	RANDOM	0.5719	0.0208	0.4072	
#19	SQUARE	0.88	1000.00	0.31	RANDOM	0.5597	0.0930	0.3474	
#26	SQUARE	0.63	1000.00	0.20	RANDOM	0.2393	0.0031	0.7577	
#30	SQUARE	0.13	1000.00	0.50	RANDOM	0.0245	0.0000	0.9755	
#49	SQUARE	1.43	1000.00	0.15	RANDOM	0.4427	0.3738	0.1835	
#104	SQUARE	1.25	1000.00	0.50	RANDOM	9.0000	9.0000	0.0000	****

END OF RUN (OR ERROR IN SHAPE)

**** INDICATES THAT THE PROBABILITY OF MISSING IS ZERO FOR AT LEAST ONE
ORIENTATION AND PROB1 AND PROB>1 SHOULD NOT BE USED FOR THIS TARGET

B.4 ELIPGRID-PC RESULTS

File TEST100.IN was used for input data to ELIPGRID-PC. Table B.5 is a listing of the output file, TEST100.HSE, produced from running ELIPGRID-PC on TEST100.IN. All square, triangular, and rectangular grid output values from ELIPGRID-PC matched Singers's Table 2 output values.

ELIPGRID-PC can also take input data from an SIF file. Table B.6 is a listing of the input file, TEST100.SIF, used to test this option. Table B.7 is a listing of the output file, TEST100.HSS, produced from running ELIPGRID-PC on TEST100.SIF. All square, triangular, and rectangular grid output values produced from the SIF file matched Singers's Table 2 output values.

B.5 CONCLUSION

ELIPGRID-1, essentially a duplicate of the published version of ELIPGRID, cannot reproduce Singer's published output for rectangular grids. ELIPGRID-2 and ELIPGRID-PC, utilizing modified RECT subroutines, can, however, reproduce Singer's published output. Therefore, the published output must have been produced by a slightly different program than the code published in the appendix of Singer's document (1972). Since ELIPGRID-2 and ELIPGRID-PC have successfully matched the published output in 100 out of 100 cases, they can be considered equivalent to the ELIPGRID code that produced the output published in Singer's Table 2 (Singer 1972).

B-11

Table B.5. ELIPGRID-PC output file listing

File: C:\CLIPPER2\EDITOR\EGPC\VALID100\TEST100.HSE Print Date: 08/11/94

Page: 1

Output from ORNL/GJ ELIPGRID-PC Program Version: 08/11/94

File Name.: C:\CLIPPER2\EDITOR\EGPC\VALID100\TEST100.HSE

Created on: 08/11/94

Input file: TEST100.IN using ELIPGRID format.

Title line: Test100.In input test file for ELIPGRD1, 2, M, and HOTSPOT, 02/03/94.

Target	Grid Type	Semimajor Axis in Relative Units	Gridspace in Orig Units	Shape	Angle	Prob(0)
#261	Square	1.2500	800.00	0.38	22.0	0.0000
#187	Square	1.5625	800.00	0.30	6.0	0.0311
#190	Square	1.5625	800.00	0.50	38.0	0.0000
#147	Square	0.3750	800.00	0.25	24.0	0.8896
#10	Square	0.7813	800.00	0.50	35.0	0.0906
#19	Square	1.0938	800.00	0.31	7.0	0.3059
#26	Square	0.7813	800.00	0.20	18.0	0.6165
#30	Square	0.1563	800.00	0.50	24.0	0.9617
#49	Square	2.0313	800.00	0.15	11.0	0.0470
#104	Square	1.5625	800.00	0.50	0.0	0.0000
#261	Square	1.0000	1000.00	0.38	22.0	0.0619
#187	Square	1.2500	1000.00	0.30	6.0	0.2337
#190	Square	1.2500	1000.00	0.50	38.0	0.0000
#147	Square	0.3000	1000.00	0.25	24.0	0.9293
#10	Square	0.6250	1000.00	0.50	35.0	0.3864
#19	Square	0.8750	1000.00	0.31	7.0	0.4591
#26	Square	0.6250	1000.00	0.20	18.0	0.7546
#30	Square	0.1250	1000.00	0.50	24.0	0.9755
#49	Square	1.6250	1000.00	0.15	11.0	0.2499
#104	Square	1.2500	1000.00	0.50	0.0	0.0000
#261	Square	0.6667	1500.00	0.38	22.0	0.4694
#187	Square	0.8333	1500.00	0.30	6.0	0.5107
#190	Square	0.8333	1500.00	0.50	38.0	0.0266
#147	Square	0.2000	1500.00	0.25	24.0	0.9686
#10	Square	0.4167	1500.00	0.50	35.0	0.7273
#19	Square	0.5833	1500.00	0.31	7.0	0.6783
#26	Square	0.4167	1500.00	0.20	18.0	0.8909
#30	Square	0.0833	1500.00	0.50	24.0	0.9891
#49	Square	1.0833	1500.00	0.15	11.0	0.5307
#104	Square	0.8333	1500.00	0.50	0.0	0.2198
#261	Triangular	1.1633	859.66	0.38	22.0	0.0000
#187	Triangular	1.4541	859.66	0.30	6.0	0.0000
#190	Triangular	1.4541	859.66	0.50	22.0	0.0000
#10	Rectangular, 2.0/1	1.1048	565.69	0.50	35.0	0.1518
#19	Rectangular, 2.0/1	1.5468	565.69	0.31	7.0	0.0646
#26	Rectangular, 2.0/1	1.1048	565.69	0.20	18.0	0.6165
#30	Rectangular, 2.0/1	0.2210	565.69	0.50	24.0	0.9617
#49	Rectangular, 2.0/1	2.8726	565.69	0.15	11.0	0.0000
#104	Rectangular, 2.0/1	2.2097	565.69	0.50	0.0	0.0000
#261	Rectangular, 2.0/1	1.4142	707.11	0.38	22.0	0.0022
#187	Rectangular, 2.0/1	1.7678	707.11	0.30	6.0	0.0000
#190	Rectangular, 2.0/1	1.7678	707.11	0.50	38.0	0.0000
#147	Rectangular, 2.0/1	0.4243	707.11	0.25	66.0	0.9293
#10	Rectangular, 2.0/1	0.8839	707.11	0.50	35.0	0.3882
#19	Rectangular, 2.0/1	1.2374	707.11	0.31	7.0	0.2989
#26	Rectangular, 2.0/1	0.8839	707.11	0.20	18.0	0.7546
#30	Rectangular, 2.0/1	0.1768	707.11	0.50	24.0	0.9755
#49	Rectangular, 2.0/1	2.2981	707.11	0.15	11.0	0.0090
#104	Rectangular, 2.0/1	1.7678	707.11	0.50	0.0	0.0000
#261	Rectangular, 2.0/1	0.9428	1060.66	0.38	22.0	0.4694
#187	Rectangular, 2.0/1	1.1785	1060.66	0.30	6.0	0.3721
#147	Triangular	0.3490	859.66	0.25	6.0	0.8896
#10	Triangular	0.7270	859.66	0.50	25.0	0.1113
#19	Triangular	1.0178	859.66	0.31	7.0	0.2594
#26	Triangular	0.7270	859.66	0.20	18.0	0.6165
#30	Triangular	0.1454	859.66	0.50	24.0	0.9617

Table B.5. (cont.)

File: C:\CLIPPER2\EDITOR\EGPC\VALID100\TEST100.HSE

Print Date: 08/11/94

Page: 2

#49	Triangular	1.8903	859.66	0.15	11.0	0.0023
#104	Triangular	1.4541	859.66	0.50	0.0	0.0000
#261	Triangular	0.9306	1074.57	0.38	22.0	0.0134
#187	Triangular	1.1633	1074.57	0.30	6.0	0.1807
#190	Triangular	1.1633	1074.57	0.50	22.0	0.0000
#147	Triangular	0.2792	1074.57	0.25	6.0	0.9293
#10	Triangular	0.5816	1074.57	0.50	25.0	0.3864
#19	Triangular	0.8143	1074.57	0.31	7.0	0.4257
#26	Triangular	0.5816	1074.57	0.20	18.0	0.7546
#30	Triangular	0.1163	1074.57	0.50	24.0	0.9755
#49	Triangular	1.5122	1074.57	0.15	11.0	0.1998
#104	Triangular	1.1633	1074.57	0.50	0.0	0.0000
#261	Triangular	0.6204	1611.86	0.38	22.0	0.4694
#187	Triangular	0.7755	1611.86	0.30	6.0	0.4810
#190	Triangular	0.7755	1611.86	0.50	22.0	0.0677
#147	Triangular	0.1861	1611.86	0.25	6.0	0.9686
#10	Triangular	0.3878	1611.86	0.50	25.0	0.7273
#19	Triangular	0.5429	1611.86	0.31	7.0	0.6695
#26	Triangular	0.3878	1611.86	0.20	18.0	0.8909
#30	Triangular	0.0776	1611.86	0.50	24.0	0.9891
#49	Triangular	1.0082	1611.86	0.15	11.0	0.5058
#104	Triangular	0.7755	1611.86	0.50	0.0	0.1712
#261	Rectangular, 2.0/1	1.7678	565.69	0.38	22.0	0.0000
#187	Rectangular, 2.0/1	2.2097	565.69	0.30	6.0	0.0000
#190	Rectangular, 2.0/1	2.2097	565.69	0.50	38.0	0.0000
#147	Rectangular, 2.0/1	0.5303	565.69	0.25	66.0	0.8896
#190	Rectangular, 2.0/1	1.1785	1060.66	0.50	38.0	0.0969
#147	Rectangular, 2.0/1	0.2828	1060.66	0.25	66.0	0.9686
#10	Rectangular, 2.0/1	0.5893	1060.66	0.50	35.0	0.7273
#19	Rectangular, 2.0/1	0.8250	1060.66	0.31	7.0	0.6686
#26	Rectangular, 2.0/1	0.5893	1060.66	0.20	18.0	0.8909
#30	Rectangular, 2.0/1	0.1179	1060.66	0.50	24.0	0.9891
#49	Rectangular, 2.0/1	1.5321	1060.66	0.15	11.0	0.4469
#104	Rectangular, 2.0/1	1.1785	1060.66	0.50	0.0	0.0600
#261	Square	1.0000	1000.00	0.38	Random	0.1137
#187	Square	1.2500	1000.00	0.30	Random	0.0730
#190	Square	1.2500	1000.00	0.50	Random	0.0000
#147	Square	0.3000	1000.00	0.25	Random	0.9293
#10	Square	0.6250	1000.00	0.50	Random	0.4072
#19	Square	0.8750	1000.00	0.31	Random	0.3474
#26	Square	0.6250	1000.00	0.20	Random	0.7577
#30	Square	0.1250	1000.00	0.50	Random	0.9755
#49	Square	1.6250	1000.00	0.15	Random	0.1835
#104	Square	1.2500	1000.00	0.50	Random	0.0000

END OF RUN (OR ERROR IN SHAPE OR L/G RATIO > 3.0)

Table B.6. ELIPGRID-PC SIF-style input file listing

File: C:\CLIPPER2\EDITOR\EGPC\VALID100\TEST100.SIF Print Date: 08/10/94

Page: 1

Test100.SIF, an SIF format input test file for HOTSPOT, 02/06/94.

* This sample SIF (Simplified Input Format) file illustrates the format specs:

- * (1) The 1st line in the file is always the title line,
- * just as in an ELIPGRID formatted input file.
- * (2) Any line can be commented by using an asterisk, *,
- * as the 1st nonblank character.
- * (3) The data values must be separated by 1 or more spaces.
- * (4) They must come in the order shown below, but the 2nd, 3rd, & 4th data
- * rows below illustrate that between-column spacing does not matter.
- * (5) No worries with column spacing is what makes this format "simple" in
- * contrast to ELIPGRID's rigid FORTRAN style column format.
- * (6) Note that for rectangular grids, the long/short side ratio follows the
- * data line as in ELIPGRID. However, it need not be in columns 1-10.
- * (7) End of file can now be either Shape > 1, as in ELIPGRID's format, or
- * simply no more data lines in the file.
- *

Semimajor	Shape	Angle	GridSize	Type	Orient.	TargetID
1000.0	0.38	22.0	800.0	1	0	#261
* Note how next 3 lines do not match ELIPGRID's column format.						
1250.0	0.30	6.0	800.0	1	0	#187
1250.0	0.50	38.0	800.0	1	0	#190
300.0	0.25	24.0	800.0	1	0	#147
625.0	0.50	35.0	800.0	1	0	#10
875.0	0.31	7.0	800.0	1	0	#19
625.0	0.20	18.0	800.0	1	0	#26
125.0	0.50	24.0	800.0	1	0	#30
1625.0	0.15	11.0	800.0	1	0	#49
1250.0	0.50	0.0	800.0	1	0	#104
1000.0	0.38	22.0	1000.0	1	0	#261
1250.0	0.30	6.0	1000.0	1	0	#187
1250.0	0.50	38.0	1000.0	1	0	#190
300.0	0.25	24.0	1000.0	1	0	#147
625.0	0.50	35.0	1000.0	1	0	#10
875.0	0.31	7.0	1000.0	1	0	#19
625.0	0.20	18.0	1000.0	1	0	#26
125.0	0.50	24.0	1000.0	1	0	#30
1625.0	0.15	11.0	1000.0	1	0	#49
1250.0	0.50	0.0	1000.0	1	0	#104
1000.0	0.38	22.0	1500.0	1	0	#261
1250.0	0.30	6.0	1500.0	1	0	#187
1250.0	0.50	38.0	1500.0	1	0	#190
300.0	0.25	24.0	1500.0	1	0	#147
625.0	0.50	35.0	1500.0	1	0	#10
875.0	0.31	7.0	1500.0	1	0	#19
625.0	0.20	18.0	1500.0	1	0	#26
125.0	0.50	24.0	1500.0	1	0	#30
1625.0	0.15	11.0	1500.0	1	0	#49
1250.0	0.50	0.0	1500.0	1	0	#104
1000.0	0.38	22.0	859.66	2	0	#261
1250.0	0.30	6.0	859.66	2	0	#187
1250.0	0.50	22.0	859.66	2	0	#190
625.0	0.50	35.0	565.69	3	0	#10
2.0						
875.0	0.31	7.0	565.69	3	0	#19
2.0						
625.0	0.20	18.0	565.69	3	0	#26
2.0						
125.0	0.50	24.0	565.69	3	0	#30
2.0						
1625.0	0.15	11.0	565.69	3	0	#49
2.0						
1250.0	0.50	0.0	565.69	3	0	#104
2.0						
1000.0	0.38	22.0	707.11	3	0	#261

B-14

Table B.6. (cont.)

File: C:\CLIPPER2\EDITOR\EGPC\VALID100\TEST100.SIF

Print Date: 08/10/94

Page: 2

2.0						
1250.0	0.30	6.0	707.11	3	0	#187
2.0						
1250.0	0.50	38.0	707.11	3	0	#190
2.0						
300.0	0.25	66.0	707.11	3	0	#147
2.0						
625.0	0.50	35.0	707.11	3	0	#10
2.0						
875.0	0.31	7.0	707.11	3	0	#19
2.0						
625.0	0.20	18.0	707.11	3	0	#26
2.0						
125.0	0.50	24.0	707.11	3	0	#30
2.0						
1625.0	0.15	11.0	707.11	3	0	#49
2.0						
1250.0	0.50	0.0	707.11	3	0	#104
2.0						
1000.0	0.38	22.0	1060.66	3	0	#261
2.0						
1250.0	0.30	6.0	1060.66	3	0	#187
2.0						
300.0	0.25	6.0	859.66	2	0	#147
625.0	0.50	25.0	859.66	2	0	#10
875.0	0.31	7.0	859.66	2	0	#19
625.0	0.20	18.0	859.66	2	0	#26
125.0	0.50	24.0	859.66	2	0	#30
1625.0	0.15	11.0	859.66	2	0	#49
1250.0	0.50	0.0	859.66	2	0	#104
1000.0	0.38	22.0	1074.57	2	0	#261
1250.0	0.30	6.0	1074.57	2	0	#187
1250.0	0.50	22.0	1074.57	2	0	#190
300.0	0.25	6.0	1074.57	2	0	#147
625.0	0.50	25.0	1074.57	2	0	#10
875.0	0.31	7.0	1074.57	2	0	#19
625.0	0.2	18.0	1074.57	2	0	#26
125.0	0.50	24.0	1074.57	2	0	#30
1625.0	0.15	11.0	1074.57	2	0	#49
1250.0	0.50	0.0	1074.57	2	0	#104
1000.0	0.38	22.0	1611.86	2	0	#261
1250.0	0.30	6.0	1611.86	2	0	#187
1250.0	0.50	22.0	1611.86	2	0	#190
300.0	0.25	6.0	1611.86	2	0	#147
625.0	0.50	25.0	1611.86	2	0	#10
875.0	0.31	7.0	1611.86	2	0	#19
625.0	0.20	18.0	1611.86	2	0	#26
125.0	0.50	24.0	1611.86	2	0	#30
1625.0	0.15	11.0	1611.86	2	0	#49
1250.0	0.50	0.0	1611.86	2	0	#104
1000.0	0.38	22.0	565.69	3	0	#261
2.0						
1250.0	0.30	6.0	565.69	3	0	#187
2.0						
1250.0	0.50	38.0	565.69	3	0	#190
2.0						
300.0	0.25	66.0	565.69	3	0	#147
2.0						
1250.0	0.50	38.0	1060.66	3	0	#190
2.0						
300.0	0.25	66.0	1060.66	3	0	#147
2.0						
625.0	0.50	35.0	1060.66	3	0	#10
2.0						

B-15

Table B.6. (cont.)

File: C:\CLIPPER2\EDITOR\EGPC\VALID100\TEST100.SIF					Print Date: 08/10/94	Page: 3
875.0	0.31	7.0	1060.66	3	0	#19
2.0						
625.0	0.20	18.0	1060.66	3	0	#26
2.0						
125.0	0.50	24.0	1060.66	3	0	#30
2.0						
1625.0	0.15	11.0	1060.66	3	0	#49
2.0						
1250.0	0.50	0.0	1060.66	3	0	#104
2.0						
1000.0	0.38	22.0	1000.0	1	1	#261
1250.0	0.30	6.0	1000.0	1	1	#187
1250.0	0.50	38.0	1000.0	1	1	#190
300.0	0.25	24.0	1000.0	1	1	#147
625.0	0.50	35.0	1000.0	1	1	#10
875.0	0.31	7.0	1000.0	1	1	#19
625.0	0.20	18.0	1000.0	1	1	#26
125.0	0.50	24.0	1000.0	1	1	#30
1625.0	0.15	11.0	1000.0	1	1	#49
1250.0	0.50	0.0	1000.0	1	1	#104
9.9	9.9	9.9	9.9	9	9	EOF

Table B.7. ELIPGRID-PC SIF-style output file listing

File: C:\CLIPPER2\EDITOR\EGPC\VALID100\TEST100.HSS Print Date: 08/11/94

Page: 1

Output from ORNL/GJ ELIPGRID-PC Program Version: 08/11/94

File Name.: C:\CLIPPER2\EDITOR\EGPC\VALID100\TEST100.HSS

Created on: 08/11/94

Input file: TEST100.SIF using SIF format.

Title line: Test100.SIF, an SIF format input test file for HOTSPOT, 02/06/94.

Target	Grid Type	Semimajor Axis in Relative Units	Gridspace in Orig Units	Shape	Angle	Prob(0)
#261	Square	1.2500	800.00	0.38	22.0	0.0000
#187	Square	1.5625	800.00	0.30	6.0	0.0311
#190	Square	1.5625	800.00	0.50	38.0	0.0000
#147	Square	0.3750	800.00	0.25	24.0	0.8896
#10	Square	0.7813	800.00	0.50	35.0	0.0906
#19	Square	1.0938	800.00	0.31	7.0	0.3059
#26	Square	0.7813	800.00	0.20	18.0	0.6165
#30	Square	0.1563	800.00	0.50	24.0	0.9617
#49	Square	2.0313	800.00	0.15	11.0	0.0470
#104	Square	1.5625	800.00	0.50	0.0	0.0000
#261	Square	1.0000	1000.00	0.38	22.0	0.0619
#187	Square	1.2500	1000.00	0.30	6.0	0.2337
#190	Square	1.2500	1000.00	0.50	38.0	0.0000
#147	Square	0.3000	1000.00	0.25	24.0	0.9293
#10	Square	0.6250	1000.00	0.50	35.0	0.3864
#19	Square	0.8750	1000.00	0.31	7.0	0.4591
#26	Square	0.6250	1000.00	0.20	18.0	0.7546
#30	Square	0.1250	1000.00	0.50	24.0	0.9755
#49	Square	1.6250	1000.00	0.15	11.0	0.2499
#104	Square	1.2500	1000.00	0.50	0.0	0.0000
#261	Square	0.6667	1500.00	0.38	22.0	0.4694
#187	Square	0.8333	1500.00	0.30	6.0	0.5107
#190	Square	0.8333	1500.00	0.50	38.0	0.0266
#147	Square	0.2000	1500.00	0.25	24.0	0.9686
#10	Square	0.4167	1500.00	0.50	35.0	0.7273
#19	Square	0.5833	1500.00	0.31	7.0	0.6783
#26	Square	0.4167	1500.00	0.20	18.0	0.8909
#30	Square	0.0833	1500.00	0.50	24.0	0.9891
#49	Square	1.0833	1500.00	0.15	11.0	0.5307
#104	Square	0.8333	1500.00	0.50	0.0	0.2198
#261	Triangular	1.1633	859.66	0.38	22.0	0.0000
#187	Triangular	1.4541	859.66	0.30	6.0	0.0000
#190	Triangular	1.4541	859.66	0.50	22.0	0.0000
#10	Rectangular, 2.0/1	1.1048	565.69	0.50	35.0	0.1518
#19	Rectangular, 2.0/1	1.5468	565.69	0.31	7.0	0.0646
#26	Rectangular, 2.0/1	1.1048	565.69	0.20	18.0	0.6165
#30	Rectangular, 2.0/1	0.2210	565.69	0.50	24.0	0.9617
#49	Rectangular, 2.0/1	2.8726	565.69	0.15	11.0	0.0000
#104	Rectangular, 2.0/1	2.2097	565.69	0.50	0.0	0.0000
#261	Rectangular, 2.0/1	1.4142	707.11	0.38	22.0	0.0022
#187	Rectangular, 2.0/1	1.7678	707.11	0.30	6.0	0.0000
#190	Rectangular, 2.0/1	1.7678	707.11	0.50	38.0	0.0000
#147	Rectangular, 2.0/1	0.4243	707.11	0.25	66.0	0.9293
#10	Rectangular, 2.0/1	0.8839	707.11	0.50	35.0	0.3882
#19	Rectangular, 2.0/1	1.2374	707.11	0.31	7.0	0.2989
#26	Rectangular, 2.0/1	0.8839	707.11	0.20	18.0	0.7546
#30	Rectangular, 2.0/1	0.1768	707.11	0.50	24.0	0.9755
#49	Rectangular, 2.0/1	2.2981	707.11	0.15	11.0	0.0090
#104	Rectangular, 2.0/1	1.7678	707.11	0.50	0.0	0.0000
#261	Rectangular, 2.0/1	0.9428	1060.66	0.38	22.0	0.4694
#187	Rectangular, 2.0/1	1.1785	1060.66	0.30	6.0	0.3721
#147	Triangular	0.3490	859.66	0.25	6.0	0.8896
#10	Triangular	0.7270	859.66	0.50	25.0	0.1113
#19	Triangular	1.0178	859.66	0.31	7.0	0.2594
#26	Triangular	0.7270	859.66	0.20	18.0	0.6165
#30	Triangular	0.1454	859.66	0.50	24.0	0.9617

B-17

Table B.7. (cont.)

File: C:\CLIPPER2\EDITOR\EGPC\VALID100\TEST100.HSS Print Date: 08/11/94 Page: 2

#49	Triangular	1.8903	859.66	0.15	11.0	0.0023
#104	Triangular	1.4541	859.66	0.50	0.0	0.0000
#261	Triangular	0.9306	1074.57	0.38	22.0	0.0134
#187	Triangular	1.1633	1074.57	0.30	6.0	0.1807
#190	Triangular	1.1633	1074.57	0.50	22.0	0.0000
#147	Triangular	0.2792	1074.57	0.25	6.0	0.9293
#10	Triangular	0.5816	1074.57	0.50	25.0	0.3864
#19	Triangular	0.8143	1074.57	0.31	7.0	0.4257
#26	Triangular	0.5816	1074.57	0.20	18.0	0.7546
#30	Triangular	0.1163	1074.57	0.50	24.0	0.9755
#49	Triangular	1.5122	1074.57	0.15	11.0	0.1998
#104	Triangular	1.1633	1074.57	0.50	0.0	0.0000
#261	Triangular	0.6204	1611.86	0.38	22.0	0.4694
#187	Triangular	0.7755	1611.86	0.30	6.0	0.4810
#190	Triangular	0.7755	1611.86	0.50	22.0	0.0677
#147	Triangular	0.1861	1611.86	0.25	6.0	0.9686
#10	Triangular	0.3878	1611.86	0.50	25.0	0.7273
#19	Triangular	0.5429	1611.86	0.31	7.0	0.6695
#26	Triangular	0.3878	1611.86	0.20	18.0	0.8909
#30	Triangular	0.0776	1611.86	0.50	24.0	0.9891
#49	Triangular	1.0082	1611.86	0.15	11.0	0.5058
#104	Triangular	0.7755	1611.86	0.50	0.0	0.1712
#261	Rectangular, 2.0/1	1.7678	565.69	0.38	22.0	0.0000
#187	Rectangular, 2.0/1	2.2097	565.69	0.30	6.0	0.0000
#190	Rectangular, 2.0/1	2.2097	565.69	0.50	38.0	0.0000
#147	Rectangular, 2.0/1	0.5303	565.69	0.25	66.0	0.8896
#190	Rectangular, 2.0/1	1.1785	1060.66	0.50	38.0	0.0969
#147	Rectangular, 2.0/1	0.2828	1060.66	0.25	66.0	0.9686
#10	Rectangular, 2.0/1	0.5893	1060.66	0.50	35.0	0.7273
#19	Rectangular, 2.0/1	0.8250	1060.66	0.31	7.0	0.6686
#26	Rectangular, 2.0/1	0.5893	1060.66	0.20	18.0	0.8909
#30	Rectangular, 2.0/1	0.1179	1060.66	0.50	24.0	0.9891
#49	Rectangular, 2.0/1	1.5321	1060.66	0.15	11.0	0.4469
#104	Rectangular, 2.0/1	1.1785	1060.66	0.50	0.0	0.0600
#261	Square	1.0000	1000.00	0.38	Random	0.1137
#187	Square	1.2500	1000.00	0.30	Random	0.0730
#190	Square	1.2500	1000.00	0.50	Random	0.0000
#147	Square	0.3000	1000.00	0.25	Random	0.9293
#10	Square	0.6250	1000.00	0.50	Random	0.4072
#19	Square	0.8750	1000.00	0.31	Random	0.3474
#26	Square	0.6250	1000.00	0.20	Random	0.7577
#30	Square	0.1250	1000.00	0.50	Random	0.9755
#49	Square	1.6250	1000.00	0.15	Random	0.1835
#104	Square	1.2500	1000.00	0.50	Random	0.0000

END OF RUN (OR ERROR IN SHAPE OR L/G RATIO > 3)

APPENDIX C

TRIANGULAR GRID DISCONTINUITY

TRIANGULAR GRID DISCONTINUITY

C.1 Introduction

A modification to the ELIPGRID triangular grid code was required to correct for a discontinuity in ELIPGRID results for a small number of triangular grid cases. This appendix documents the discontinuity and explains the modification made to the original code.

The graph of the probability of missing a hot spot versus a range of increasing hot spot sizes is a good indicator of any discontinuities in the ELIPGRID algorithm. Fig. C.1 is a graph of the probability of missing a hot spot versus the semi-major axis length to grid size (L/G) ratio for a triangular grid. The orientation of the angle and the assumed hot-spot shape (the ratio of minor axis to major axis) are 15° and 0.99. As the L/G ratio increases along the x axis, the probability of missing should smoothly decrease to zero. This is obvious from the fact that larger L/G ratios imply larger hot spots and, hence, a smaller probability of missing. When the hot-spot size is such that it will always be sampled at some sampling node, then the probability of missing must be zero.

Fig. C.1 clearly reveals a discontinuity in the ELIPGRID algorithm between L/G ratios of 0.5 and 0.6. The probability of missing falls below zero at an L/G ratio of about 0.54, then jumps back to zero near an L/G ratio of 0.58. By definition, a probability less than zero is in error. In addition, the large jump in probability makes the ELIPGRID algorithm suspect for this set of parameters.

C.2 Effect on Discontinuity of the Orientation Angle

In order to determine if the discontinuity is related to hot-spot orientation angle, two additional graphs were produced with the same parameters as Fig. C.1 except for different hot-spot orientation angles of 0° and 30° (Figs. C.2 and C3). Due to symmetry

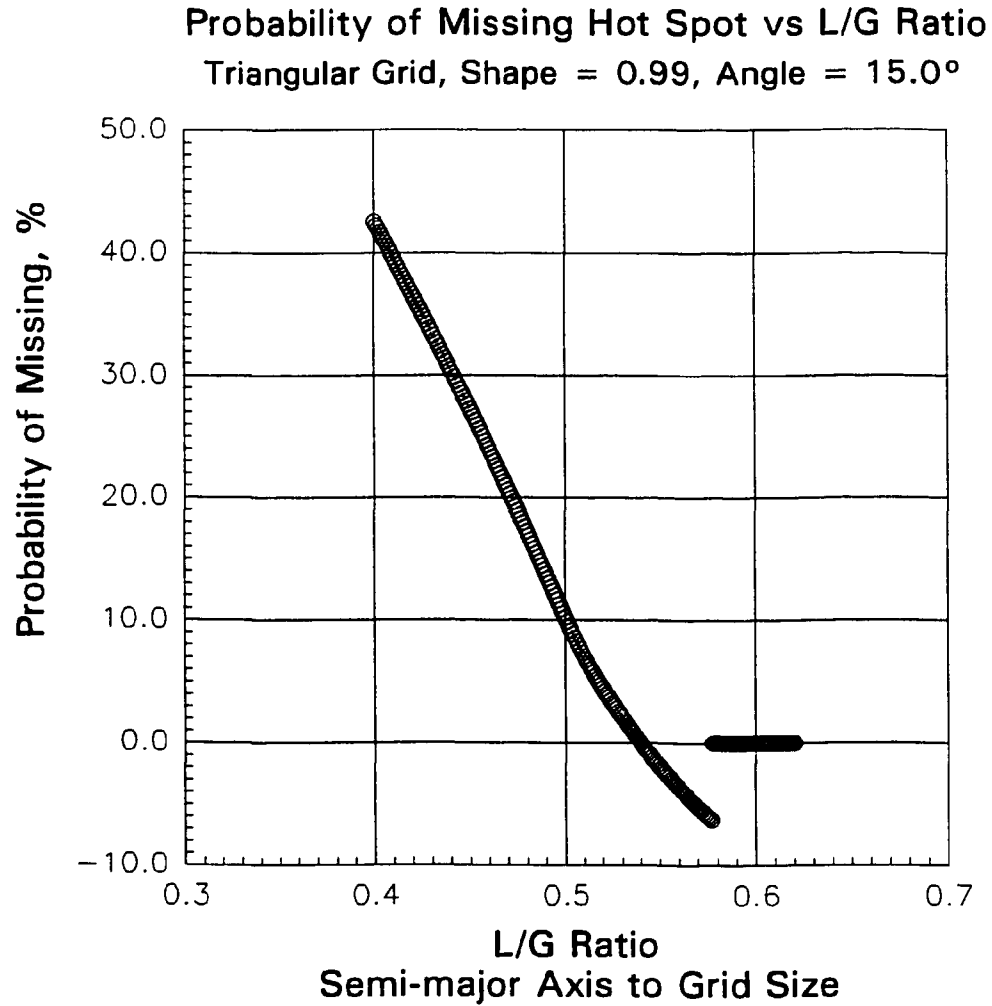


Fig. C.1. Probability of missing hot spot vs L/G ratio, triangular grid, 0.99 shape, and 15° angle.

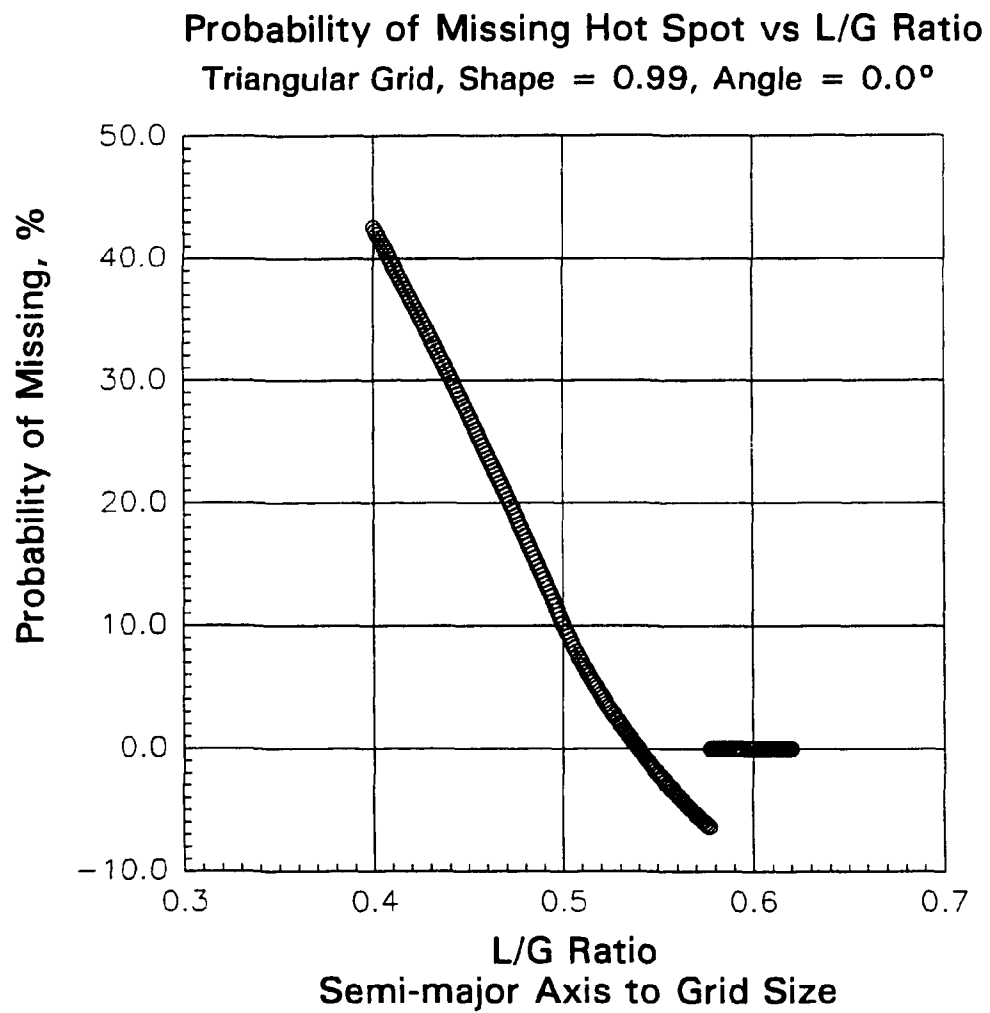


Fig. C.2. Probability of missing hot spot vs L/G ratio, triangular grid, 0.99 shape, and 0° angle.

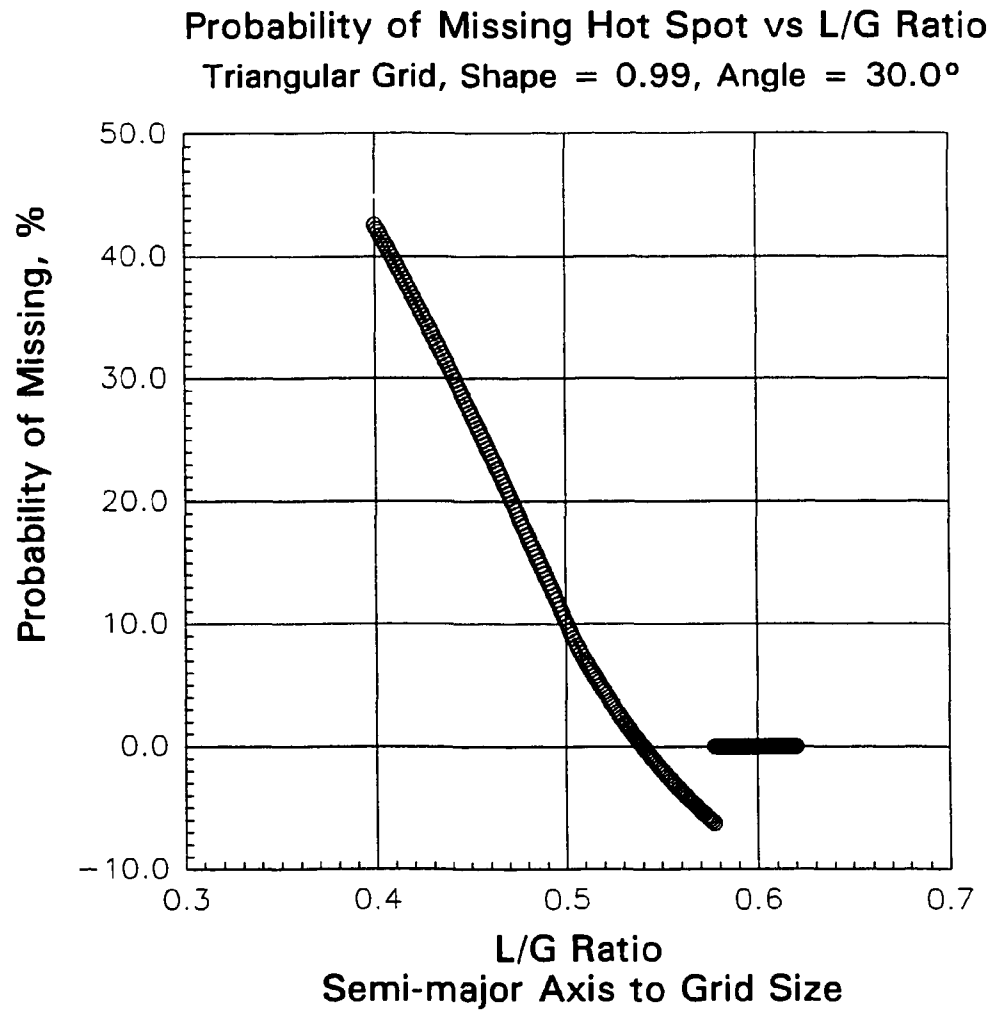


Fig. C.3. Probability of missing hot spot vs L/G ratio, triangular grid, 0.99 shape, and 30° angle.

considerations, 30° is the largest angle necessary for triangular grids (Singer and Wickman 1969). Note that the three graphs are virtually identical, although the orientation angles differ widely. These graphs illustrate that the orientation angle is of no material significance to the triangular grid discontinuity problem.

C.3 Effect on Discontinuity of the Hot-Spot Shape

To determine the effect of the hot-spot shape on the discontinuity problem, the graph in Fig. C.4 was produced unchanged from that in Fig. C.1 except that the hot-spot shape was decreased from 0.99 to 0.90. Note the large decrease in the part of the graph below a probability of zero. Fig. C.5 has an even smaller assumed hot-spot shape of 0.85. Note that the discontinuity problem has now disappeared. Also, examination of the results from the largest shape possible, 1.0, reveals no discontinuity problems (Fig. C.6). Thus, this problem can be confined to hot-spot shapes of less than 1.0 and greater than approximately 0.85.

C.4 Resolution of the Discontinuity by a 4th Order Polynomial Regression

Linear regression was used to provide a smooth curve for the triangular grid cases listed above. The solid lines in Figs. C.7, C.8, and C.9 are the 4th order polynomial regression lines calculated after the values near the discontinuity have been removed. The regression equation used is

$$P(0) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$$

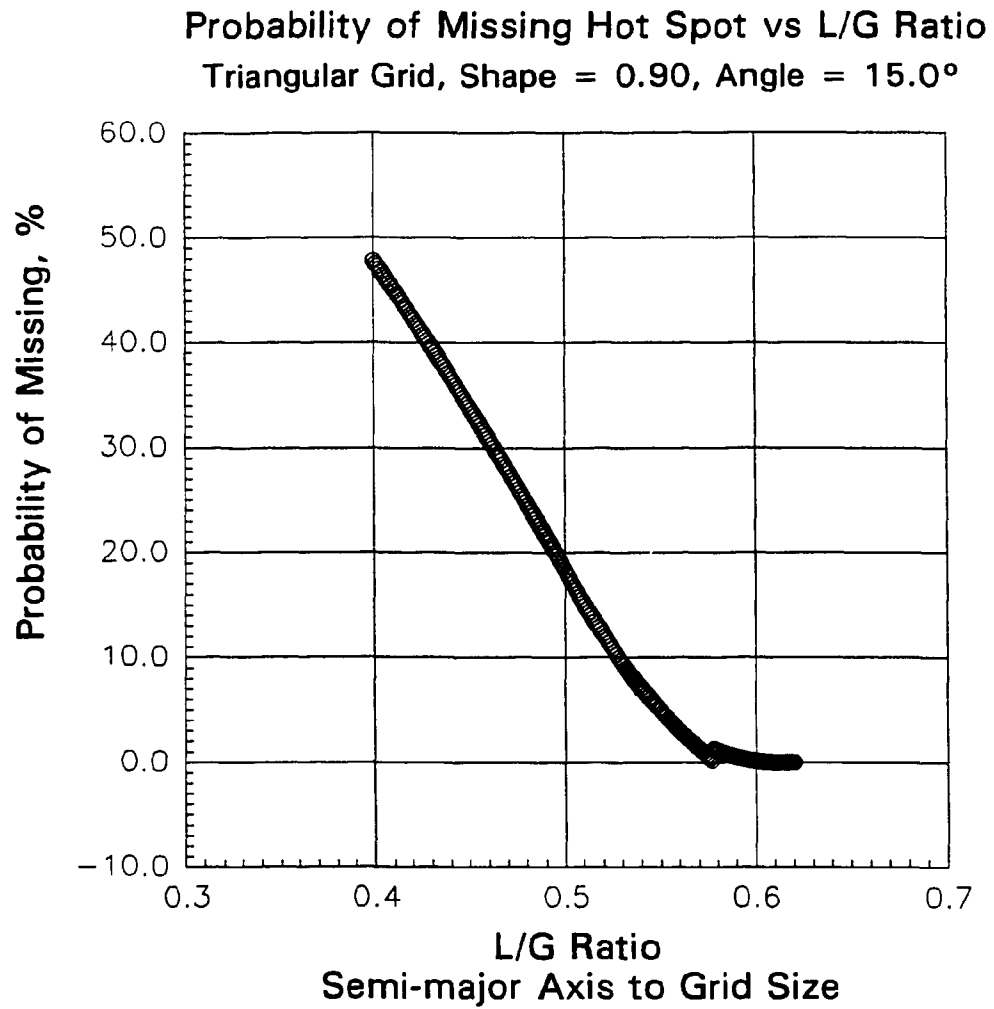


Fig. C.4. Probability of missing hot spot vs L/G ratio, triangular grid, 0.90 shape, and 15° angle.

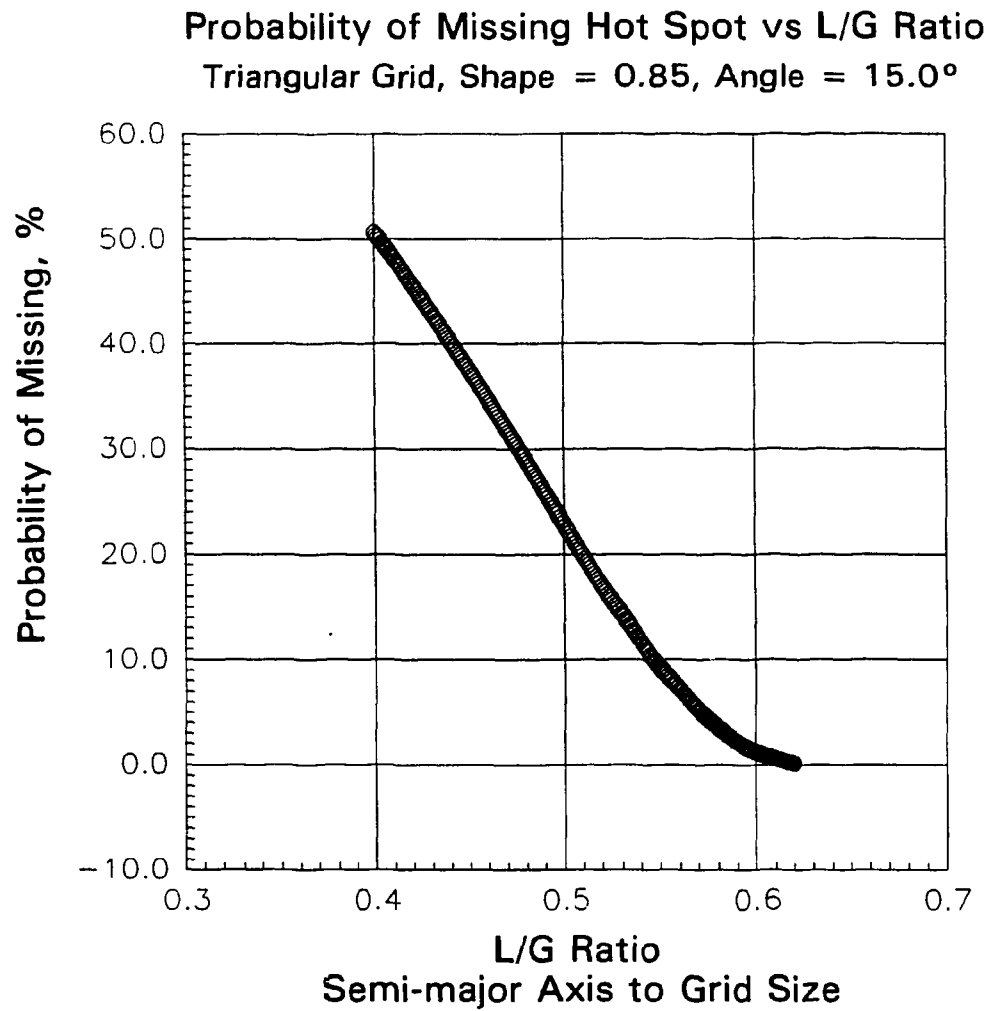


Fig. C.5. Probability of missing hot spot vs L/G ratio, triangular grid, 0.85 shape, and 15° angle.

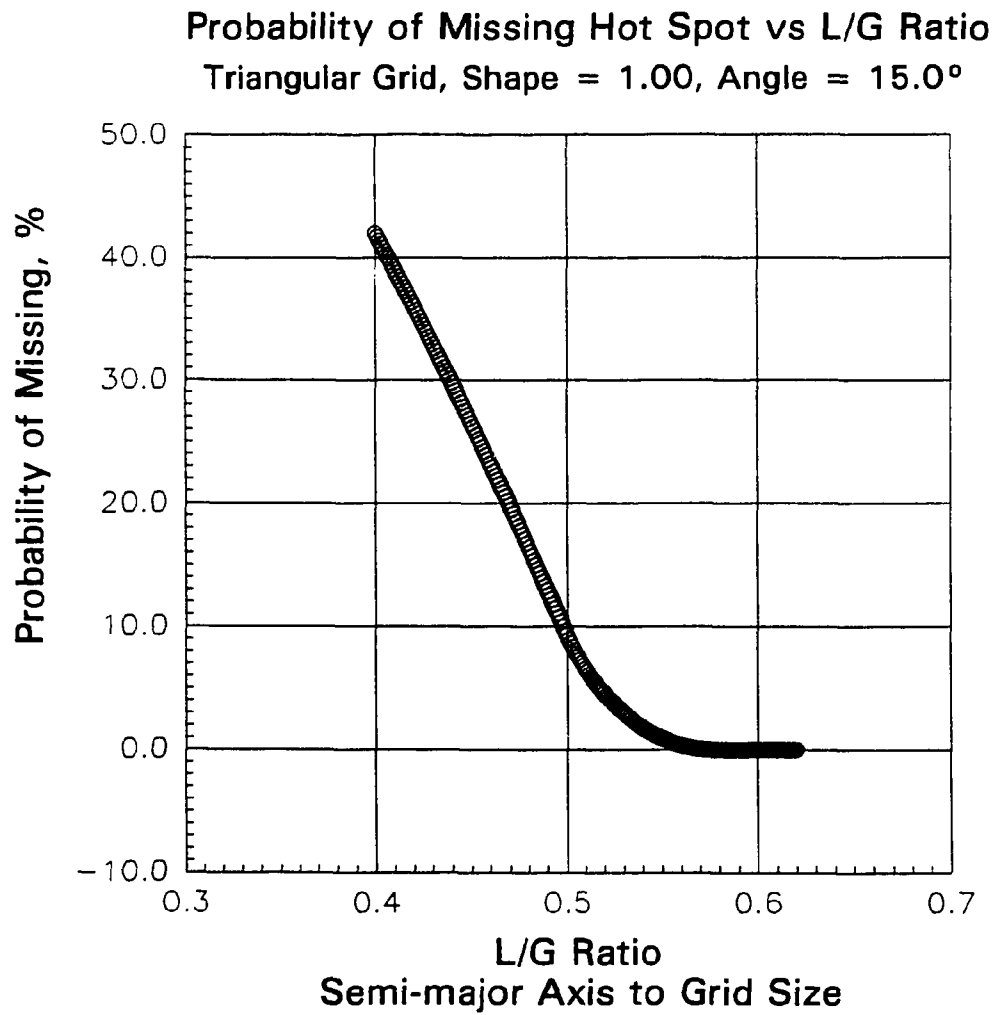
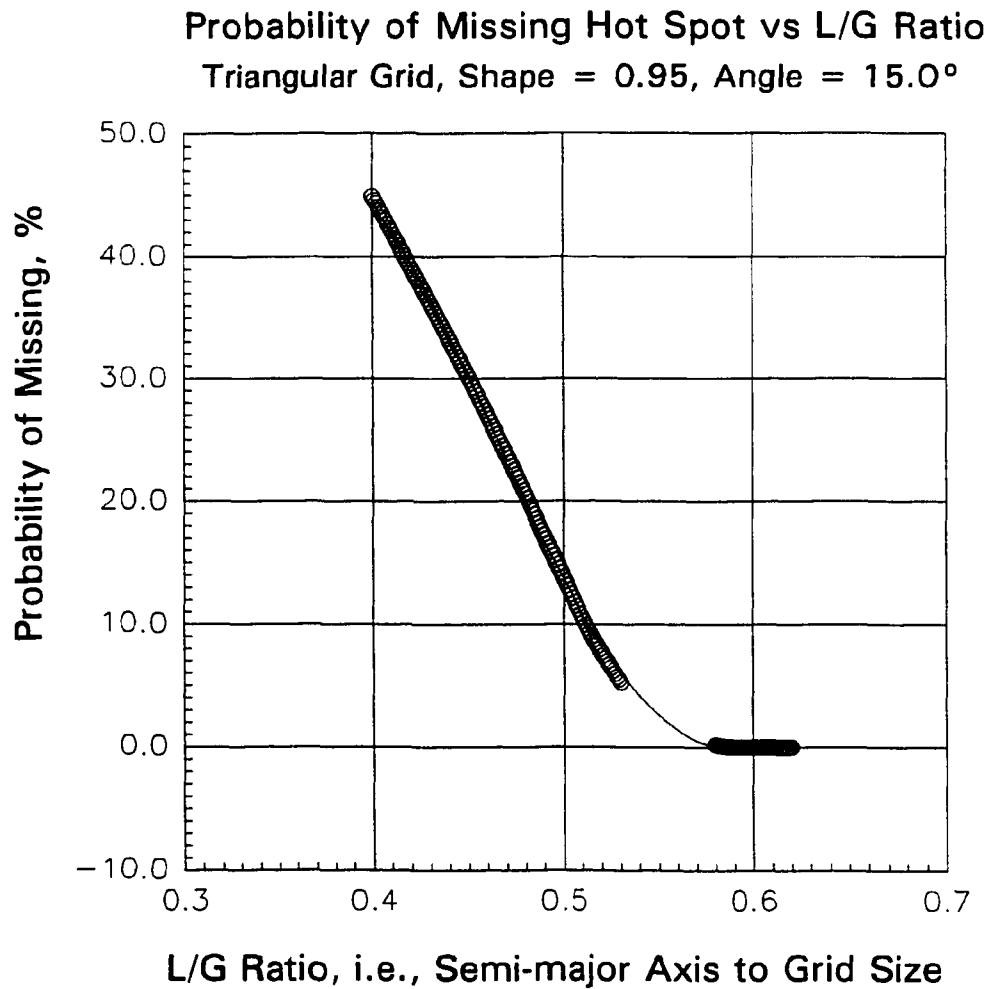
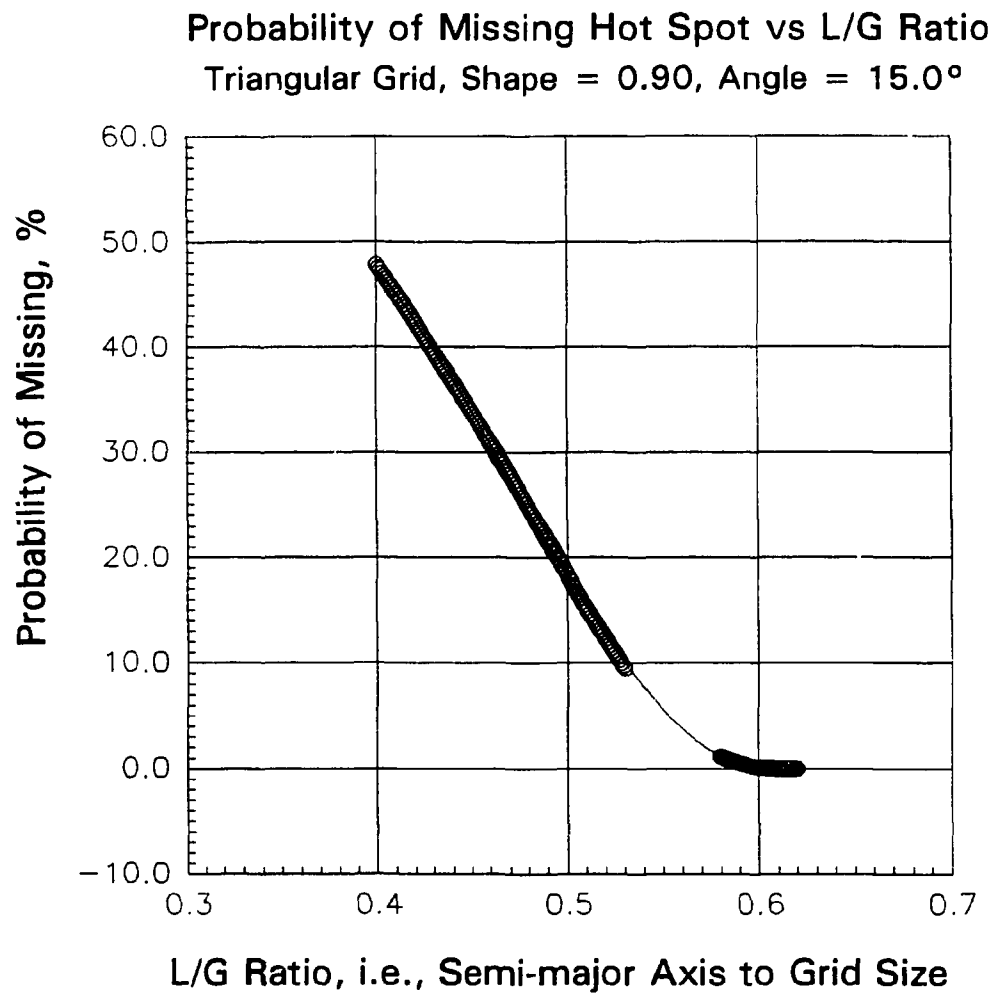


Fig. C.6. Probability of missing hot spot vs L/G ratio, triangular grid, 1.00 shape, and 15° angle.



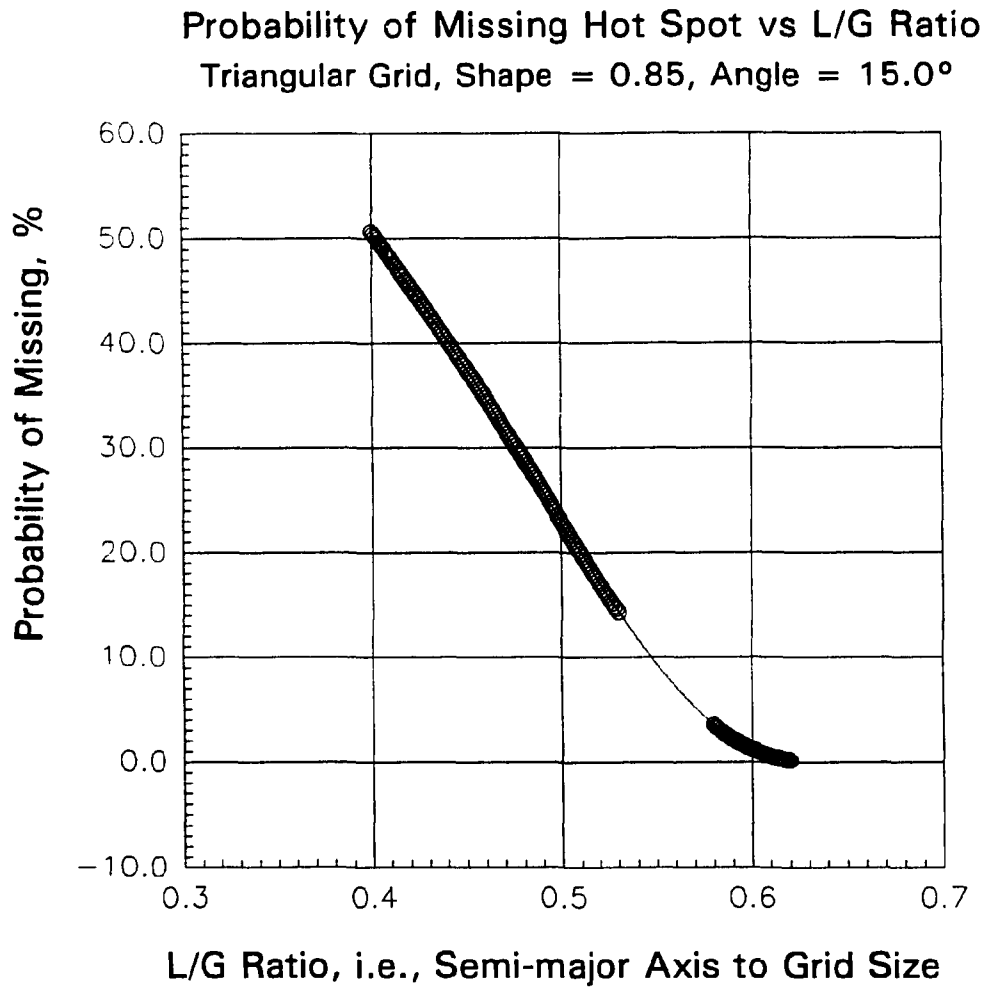
A regression line based on a 4th order polynomial is shown. The values near the discontinuity have been removed.

Fig. C.7. Probability of missing hot spot vs L/G ratio, triangular grid, 0.99 shape, and 15° angle.



A regression line based on a 4th order polynomial is shown. The values near the discontinuity have been removed.

Fig. C.8. Probability of missing hot spot vs L/G ratio, triangular grid, 0.90 shape, and 15° angle.



A regression line based on a 4th order polynomial is shown. The values near the discontinuity have been removed.

Fig. C.9. Probability of missing hot spot vs L/G ratio, triangular grid, 0.85 shape, and 15° angle.

where

$P(0)$ = probability of missing the hot spot,

β_n = one of the $n = 0$ to 4 regression parameters,

x = the L/G ratio.

The regression parameters were determined using SigmaPlot® 5.01 on data sets with the values near the discontinuity removed. Since the regression changes with the shape, (Sect. C.3), parameters for seven sets of shapes were determined. The actual parameter values determined may be found in the code in function Prob0_Regr() listed in file EGPCFORT.PRG in Appendix E.

Figures C.10 and C.11 are the graphs for shapes equal to 0.99 and 0.85 after ELIP-GRID-PC was modified with the regression equation substituted for the ELIPGRID algorithm in the L/G range greater than 0.5 and less than 0.6. These graphs demonstrate that the ELIPGRID-PC regression modification smooths out the discontinuity and removes the negative probability problem (the small negative values that still remain when the shape is near 0.99 are rounded up to zero).

The cause of the discontinuity seems to be related to the mathematical problem of dealing with the tangent of 90° . Singer and Wickman's mathematical derivation for the triangular grid case states, "It can be seen from (45) that acute angles can only occur if $k < 1/\sqrt{3} \sim 0.577$. The function $\tan \tau$ has a discontinuity at $\tau = 90^\circ$; therefore it is practical to use (47) to determine where τ is acute or obtuse" (Singer and Wickman 1969). Note their reference to the well-known mathematical discontinuity at the tangent of 90° . It is also interesting to note that the constant referred to above, $1/\sqrt{3}$, is in the range of L/G ratios where the discontinuity occurs.

Further study of their algorithm could possibly reveal a more fundamental solution to this problem than the regression method used here. However, the regression method adopted should be satisfactory in most, if not all, practical cases.

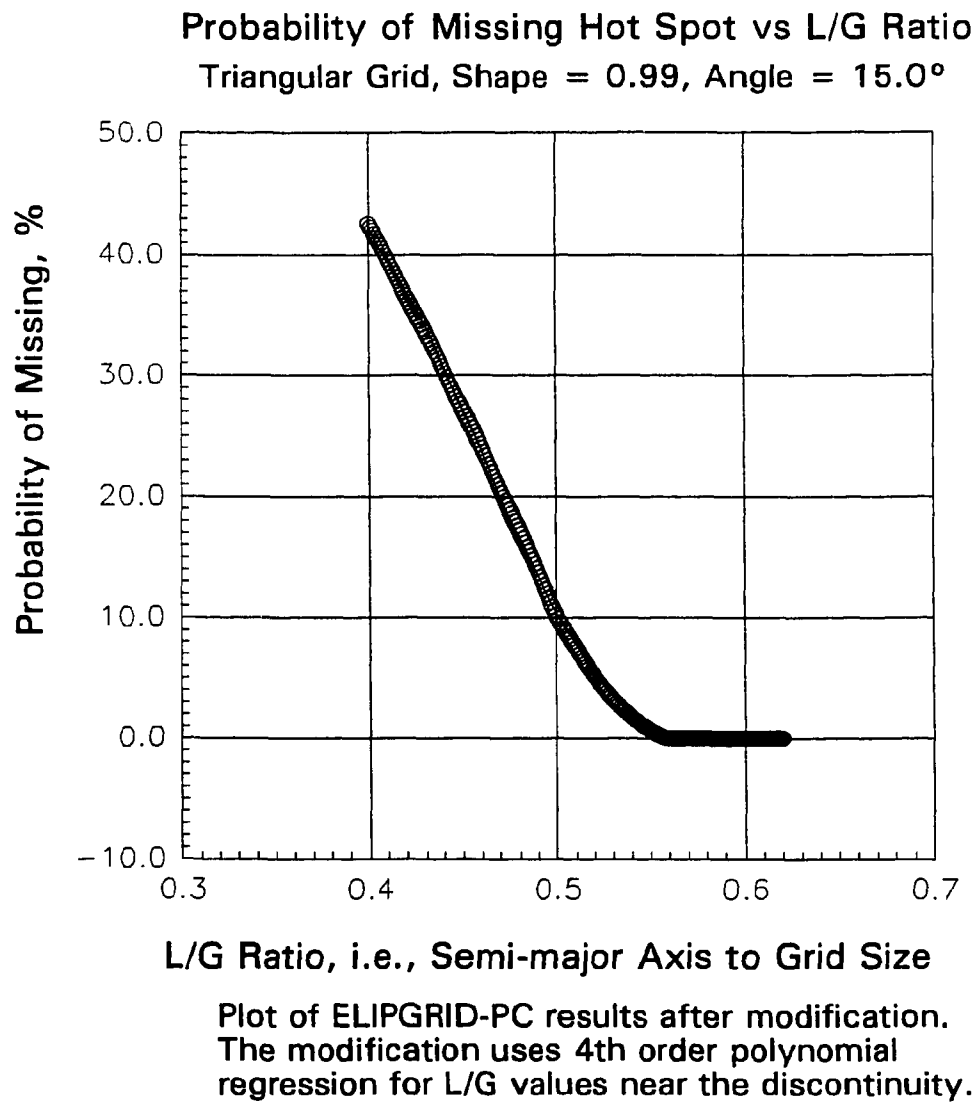
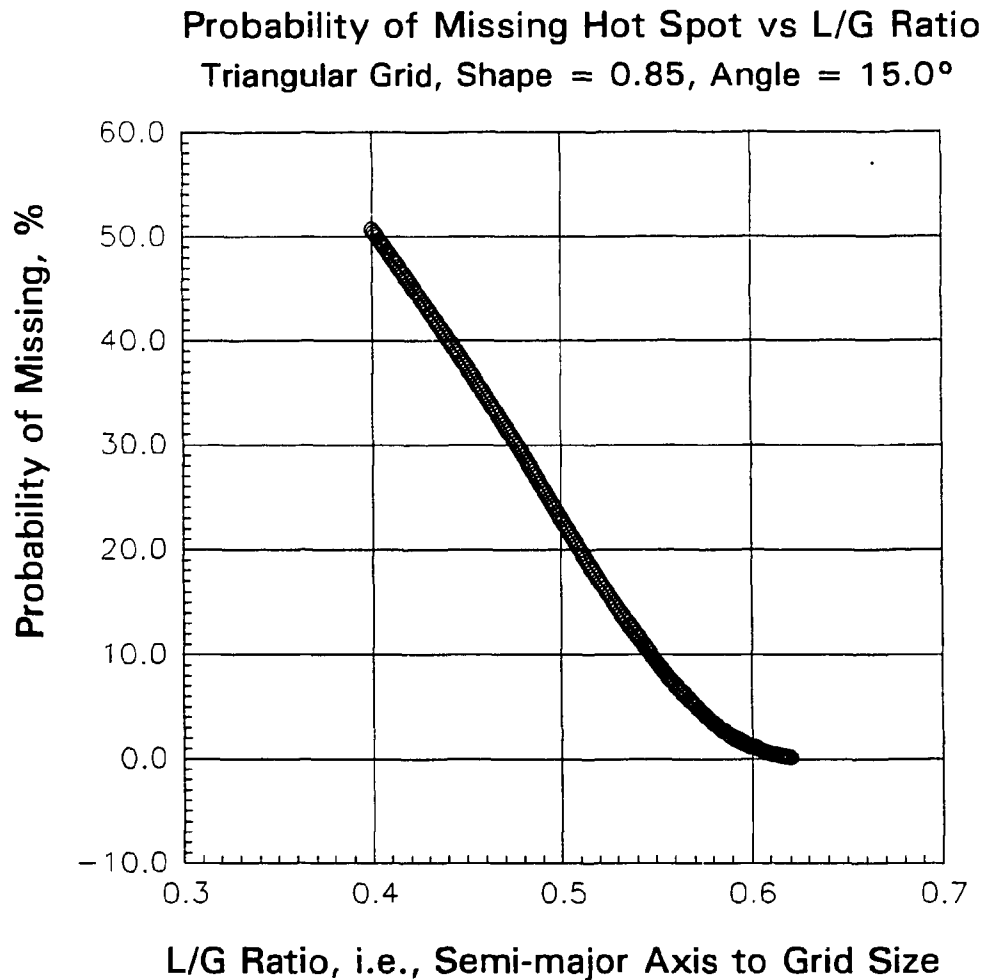


Fig. C.10. Probability of missing hot spot vs L/G ratio, triangular grid, 0.99 shape, and 15° angle.



Plot of ELIPGRID-PC results after modification.
The modification uses 4th order polynomial
regression for L/G values near the discontinuity.

Fig. C.11. Probability of missing hot spot vs L/G ratio, triangular grid, 0.85 shape, and 15° angle.

APPENDIX D

ELIPGRID-2 SOURCE CODE

APPENDIX D
ELIPGRID-2 SOURCE CODE

The first page of this appendix is an example make file for the Lahey F77L@-EM32 compiler version 5.10. The remaining pages contain all the main code and subroutines in one file.

D-2

```
# Example make file for ELIPGRID-2 using an E: RAM drive with C: drive.
FC = E:\f77l3
LINKER = E:\386link
PROGRAM = C:ELIPGRD2
DEST = C:
EXTHDRS =
FFLAGS =
HDRS =
LD_FLAGS = -STUB RUNB
LDMAP = nul
LIBS = E:\F77l3.lib, E:\UTIL3.LIB
MAKEFILE = Makefile
OBS = ELIPGRD2.obj
SRCS = ELIPGRD2.for
$(PROGRAM):      $(OBS) $(LIBS)
                  $(LINKER) $(OBS) -EXE $@ -MAP $(LDMAP) -LIB $(LIBS) $(LD_FLAGS)
clean:;           @del -f $(OBS)
depend:;          @mkmf -f $(MAKEFILE) PROGRAM=$(PROGRAM) DEST=$(DEST)
install:  $(PROGRAM)
              @echo Installing $(PROGRAM) in $(DEST)
              @if not $(DEST)x==.x copy $(PROGRAM) $(DEST)
```

Code File: ELIPGRD2.For

```

=====
* File....: ELIPGRD2.For
* Purpose.: ELIPGR2 is a modification of ELIPGRID for hot spot location.
* Note....: This code modifies ELIPGRID.For by Singer as little as
*           possible. The goal is to provide a PC version of ELIPGRID.
*           The calculation algorithms were not changed; one exception follows.
*           The RECT() subroutine appears to be in error in the original.
*           The formula for the transformed angle, REVANG, was changed to
*           essentially match the Eq. (35) in (Singer and Wickman 1969, p. 16).
*           Note that the term (1 - k) in Eq. (35) is probably a typo for
*           (1 - k2). ELIPGRID's (1.0 - SQK) is retained here.
*           The constat PI was left at 3.141492 instead of being changed to the
*           more correct 3.141593. This should be of no practical consequence.
*           Some input/output format changes were made.
*           Note that ELIPGRD2 adds explicit declaration of all variables.
* Usage....: ELIPGRD2 [<InputFileName> or <Help>]
*           The InputFileName may include a path. The output file will be
*           written to the input path.
* Input file format:
*           Line 1 = Title: (A80)
*           Line n = data.: (4F10.2,2I4,A4)
*
*           F10.4,    F10.4,    F10.4,    F10.4, I4,  I4,  A4
*           Data Variables:  A,    SHAPE,    ANGLE,    GDSPAC,MET, MET,TARGET
*           Example, Sq...: 1250.0    0.30    6.0    800.0  1  0#187
*           Example, Rec...: 1250.0    0.50    38.0   1060.66  3  0#190
*
*           2.0  <--- 2nd data line for rect.= Long/short
*           Example, Hex...: 1250.0    0.50    0.0   1611.86  2  0#104
*           Line EOF.....: Use ALL valid values EXCEPT SHAPE is > 1.0.
*           Output is to file <InputFileName>.Out.
*           When no input file passed, ELIPGRD2.In will be default file name.
* Orig. by: D.A Singer
* Mods by.: Jim Davidson
* Started.: 11/12/93
* Last mod: 02/06/94
* Compiler: Lahey Fortran F77L-EM/32 Ver. 5.10
* Options: /n0/n2/4/n7/nA2/nB/nC/nC1/nD/nF/nH/nI/nK/nL/nO/P/nQ1/nQ2/nQ3
*          /R/nS/nT/nV/W/nX/Z1  from current F77L3.FIG file.
* Linker...: Phar Lap 386Link 4.1L
* Options: Use LDFLAGS = -STUB RUNB to bind in the RUN386 loader.
*          LIBS = E:\F77L3.LIB, E:\UTIL3.LIB
*          UTIL3.LIB is the Lahey Spindrift Utility Lib. EM/32 Ver. 2.01.
* Notes....: Original code lines are commented out with a * in the left column.
*           The replacement line is usually just beneath the original line.
*           The line numbers and ID numbers on the right side were left alone.

```

Code File: ELIPGRD2.For

```

*           All variables are explicitly declared in this version.
*
*           The original ELIPGRID code documentation header starts below:
*=====
C           PROGRAM ELIPGRID                                GRID  5
C                                                    GRID  15
C   PROGRAM TO DETERMINE THE PROBABILITY OF LOCATING AN ELLIPTIC OR  GRID  25
C   CIRCULAR TARGET WITH A SQUARE, HEXAGONAL OR RECTANGULAR GRID  GRID  35
C                                                    GRID  45
C                                                    GRID  55
C   DESCRIPTION OF PARAMETERS                                GRID  65
C                                                    GRID  75
C   TARGET= ANY IDENTIFICATION OF TARGET  (READ IN "A" FORMAT)  GRID  85
C   A= LENGTH OF SEMIMAJOR AXIS OF TARGET                    GRID  95
C   SHAPE= SHAPE OF TARGET - SEMIMINOR AXIS DIVIDED BY THE SEMIMAJORGRID 105
C   ANGLE= POSITIVE ANGLE BETWEEN LONG AXIS OF TARGET AND GRID  GRID 115
C   DIRECTION - FOR A SQUARE GRID ANGLE CAN BE ANY ABGLE FROMGRID 125
C   0 TO 45 DEGRESS, FOR A HEXAGONAL GRID ANGLE CAN BE ANY  GRID 135
C   ANGLE FROM 0 TO 30 DEGREES INCLUSIVE, FOR A RECTANGULAR  GRID 145
C   GRID ANGLE CAN BE ANY ANGLE FROM 0 TO 90 DEGREES  GRID 155
C   INCLUSIVE AND IS MEASURED FROM THE X AXIS OF THE GRID  GRID 165
C   GDSPAC= DISTANCE BETWEEN POINTS ON THE GRID (IN THE SAME UNITS ASGRID 175
C   "A") - FOR A RECTANGULAR GRID GDSPAC IS THE DISTANCE  GRID 185
C   BETWEEN POINTS ALONG THE Y AXIS OF THE GRID  GRID 195
C   NET= GRID TYPE - SQUARE GRID=1,HEXAGONAL GRID=2, RECTANGULAR  GRID 205
C   GRID=3  GRID 215
C   MET= SPECIFIC OR RANDOM ORIENTATION - IF MET>0 - RANDOM  GRID 225
C   Q= SHAPE OF RECTANGULAR GRID - LONG(X) AXIS DIVIDED BY THE  GRID 235
C   SHORT(Y) AXIS  GRID 245
C  GRID 255

program ELIPGRD2
implicit none          ! Force all variables to be declared
character*8 cv_DATE    ! Version date
parameter(cv_DATE = '02/06/94')
! FMT var. replaced by hardcoded format.
*   DIMENSION TITLE(20),FMT(20)                                GRID 265
character*80 TITLE      ! First line of input file
character*4  TARGET     ! Four char. ID for each data record
character*64 cInFile    ! Input file
character*64 cOutFile   ! Output file
character*4  cUpParam    ! Uppcase of 1st 4 letters of passed param

integer I
integer IBLANK

```

Code File: ELIPGRD2.For

```

integer iDotPos      ! "." position in input filename
integer IPRIN        ! Outfile unit number
integer IREAD        ! Infile unit number
integer IROT         ! 45, 30, or 90 angles to test if random
integer IWARN
integer IZONK
integer M
integer MET          ! Target orientation, > 0=random
integer MROT         ! Current random angle
integer NET          ! 1=square, 2=hex, 3=rectangle

real A              ! Length of semimajor axis
real ALPHA
real ANGLE
real ANP
real AQUAR
real AREA1
real AREA2
real AREA3
real AREA4
real AREA5
real AREA6
real AREA7
real AREA8
real AREA9
real AREA10
real ASQ
real AVPRO
real AVPR1
real AVPR2
real B
real BALLS
real BSQU
real C
real CAROL
real CIM
real CNM
real D
real DJ0
real DJ1
real DMO
real DM1
real EOU
real FIN

```

Code File: ELIPGRD2.For

```

real FORM
real GAME
real GDSPAC          ! Grid spacing
real GRO
real HAI
real HALFC          ! C/2
real HALFD          ! D/2
real HALFJO          ! JO/2
real HALFJ1          ! etc.
real HALFMO
real HALFM1
real HORN
real PET
real PI              ! Constant pi, 3.141592 in ELIPGRID
real POT
real PROBO          ! Prob. of no hits
real PROB1          ! Prob. of 1 hit
real PROB2
real Q              ! For rect. grid, Q=longside/shortside
real RO
real RDW
real REVA          ! For rect. grid, transformed A
real REVANG        ! For rect. grid, transformed angle
real REVK          ! For rect. grid, transformed SHAPE
real SER
real SHAPE          ! Major/minor axis
real SLING
real SNGLE
real SUMO
real SUM1
real SUM2
real T
real TIN
real TI2
real WINE
real XHAPE
real XI
real XM
real Y1
real YI
real YAM
real YM
real ZAP

```

Code File: ELIPGRD2.For

```

! External functions.
real ARCO                      ! Function in this code
logical fexist                 ! Lahey Spindrift Utility Lib

print *, 'Program..: ORNL/GJ ELIPGRD2  RECT '//
& 'subroutine modified in this code.'
print *, 'Version..: '//CV_DATE
print *, 'Note.....: Program is an ORNL/GJ modification of '//
& 'ELIPGRID by Singer.'

DATA IBLANK/4H      /,1WARN/4H****/
IREAD=5
IPRIN=6
* READ (IREAD,5) TITLE replaced by below

! Get input file name.
call GetInFile(cInFile)
call upc(cInFile,cUpParam)

! Check for HELP parameter or missing file error.
if (cUpParam(1:4)=='HELP') then
    call HelpScreen()
    stop
elseif (.not. fexist(cInFile)) then
    call tone(220,18)          ! From the Lahey Spindrift Utility Lib.
    print *, 'ERROR 1..: Input file not found.'
    print *, 'File Name: '//charnb(cInFile)
    print *, 'Usage.....: ELIPGRD2 [<InputFileName> or <HELP>]'
    stop
endif

! Make cOutFile name from cInFile.
iDotPos = index(cInFile, ".")
if (iDotPos == 0) then
    cOutFile = charnb(cInFile)//'.OUT'
else
    cOutFile = cInFile(1:iDotPos)//'.OUT'
endif

! Open input and output files.
open (IREAD, file=cInFile, status="OLD")
open (IPRIN, file=cOutFile)
print *, 'Input.....: '//charnb(cInFile)
print *, 'Output...: '//charnb(cOutFile)

```

Code File: ELIPGRD2.For

```

      print *,

      ! Begin to read and write files.
      read (IREAD, fmt='(A80)') TITLE

C                                         GRID 315
C      READS FORMAT FOR DATA (replaced by hardcoded format)          GRID 325
C                                         GRID 335
*      READ (IREAD,5) FMT                                         GRID 345
*      5 FORMAT (20A4)                                         GRID 355
*      WRITE (IPRIN,10) TITLE                                         GRID 365
*      10 FORMAT (1H1,25X,20A4//)                                         GRID 375
      write (IPRIN, fmt='(1H ,10X,A/ )') 'ELIPGRD2 Output File'
      write (IPRIN, fmt='(1H ,10X,A//)') 'Data from: '//TITLE
      PI=3.141592                                         GRID 385
      WRITE (IPRIN,15)                                         GRID 395
*      15 FORMAT (1H0,6HTARGET,4X,9HGRID TYPE,29X,14HSEMIMAJOR AXIS,4X,9HGRID 405
      15 FORMAT (1H ,6HTARGET,4X,9HGRID TYPE,29X,14HSEMIMAJOR AXIS,4X,9HGRID
      10SPACE,4X,5HSHAPE,4X,5HANGLE,4X,7HPROB(1),4X,8HPROB(>1),4X,7HPROB(GRID 415
*      10)/46X,17HIN RELATIVE UNITS,3X,13HIN ORIG UNITS//)          GRID 425
      10)/46X,17HIN RELATIVE UNITS,3X,13HIN ORIG UNITS)
      T1Z=0.50000                                         GRID 435
      RDW=SQRT(3.0)*0.5                                         GRID 445
      20 MET=0                                         GRID 455
C                                         GRID 465
C      READ DATA                                         GRID 475
C                                         GRID 485
*      READ (IREAD,FMT) A,SHAPE,ANGLE,GDSPAC,NET,MET,TARGET          GRID 495
      READ (IREAD, '(4F10.2,214,A4)') A,SHAPE,ANGLE,GDSPAC,NET,MET,TARGET
      IZONK=IBLANK                                         GRID 505
      A=A/GDSPAC                                         GRID 515
      SLING=A                                         GRID 525
      XHAPE=SHAPE                                         GRID 535
      SNGLE=ANGLE                                         GRID 545
      SUM1=0.0                                         GRID 555
      SUM2=0.0                                         GRID 565
      SUM0=0.0                                         GRID 575
      MROT=0                                         GRID 585
      IF (MET) 35,35,30                                         GRID 595
      30 ANGLE=MROT                                         GRID 605
C                                         GRID 615
C      AREAS 1 TO 10 ARE RELATIVE AREAS OF OVERLAP IN THE TRANSFORMED NETGRID 625
C                                         GRID 635
      35 AREA1=0.0                                         GRID 645
      AREA2=0.0                                         GRID 655

```

Code File: ELIPGRD2.For

AREA3=0.0	GRID 665
AREA4=0.0	GRID 675
AREA5=0.0	GRID 685
AREA6=0.0	GRID 695
AREA7=0.0	GRID 705
AREA8=0.0	GRID 715
AREA9=0.0	GRID 725
AREA10=0.0	GRID 735
C	GRID 745
C PROB0 IS THE PROBABILITY OF MISSING THE TARGET	GRID 755
C PROB1 IS THE PROBABILITY OF LOCATING THE TARGET ONCE	GRID 765
C PROB2 IS THE PROBABILITY OF LOCATING THE TARGET TWO OR MORE TIMES	GRID 775
C	GRID 785
PROB0=0.0	GRID 795
PROB1=0.0	GRID 805
PROB2=0.0	GRID 815
C	GRID 825
C DETERMINES THE GRID TYPE	GRID 835
C	GRID 845
GO TO (65,40,45),NET	GRID 855
C	GRID 865
C HEXAGONAL NET	GRID 875
C	GRID 885
40 FIN=RDW	GRID 895
IROT=30	GRID 905
ZAP=6.0	GRID 915
BALLS=0.57735	GRID 925
GO TO 75	GRID 935
C	GRID 945
C RECTANGULAR NET	GRID 955
C	GRID 965
45 IF (MROT) 50,50,60	GRID 975
C	GRID 985
C READ SHAPE OF RECTANGULAR GRID	GRID 995
C	GRID1005
50 READ (IREAD,55) Q	GRID1015
55 FORMAT (F10.5)	GRID1025
* 60 CALL RECT(SLING,XHAPE,ANGLE,Q,REVK,REVA,REVANG)	GRID1035
* Argument SLING is never used by subroutine RECT().	
60 CALL RECT(XHAPE,ANGLE,Q,REVK,REVA,REVANG)	
SHAPE=REVK	GRID1045
A=REVA*SLING	GRID1055
ANGLE=REVANG	GRID1065
IROT=90	GRID1075

Code File: ELIPGRD2.For

```

      GO TO 70                                GRID1085
C                                           GRID1095
C                                           GRID1105
C      SQUARE NET                            GRID1115
C                                           GRID1125
      65 IROT=45                              GRID1135
      70 FIN=1.000                            GRID1145
      ZAP=4.0                                GRID1155
      BALLS=0.707107                         GRID1165
      75 IF (SHAPE-0.05) 80,95,95            GRID1175
      80 IF (A-2.0) 95,95,85                 GRID1185
      85 WRITE (IPRIN,90) TARGET              GRID1195
* 90 FORMAT (1H0,6HTARGET,A4,45H IS TOO NEEDLE-LIKE AND LONG FOR THIS PGRID1205
90 FORMAT (1H ,6HTARGET,A4,45H IS TOO NEEDLE-LIKE AND LONG FOR THIS P
1ROGRAM)                                GRID1215
      GO TO 20                                GRID1225
95 IF (SHAPE-1.0) 140,115,100               GRID1235
C                                           GRID1245
C      RUN IS TERMINATED WHEN A SHAPE IS GREATER THAN ONE GRID1255
C                                           GRID1265
100 WRITE (IPRIN,105)                       GRID1275
* 105 FORMAT (1H0,////,50X,31HEND OF RUN (OR ERROR IN SHAPE)) GRID1285
105 FORMAT (1H ,//, ' END OF RUN (OR ERROR IN SHAPE)')
      WRITE (IPRIN,110)                      GRID1295
* 110 FORMAT (1H0,25X,93H**** INDICATES THAT THE PROBABILITY OF MISSING GRID1305
* 11S ZERO FOR AT LEAST ONE ORIENTATION AND PROB1,/,25X,45HAND PROB>1GRID1315
* 1 SHOULD NOT BE USED FOR THIS TARGET)      GRID1325
110 FORMAT (/, ' **** INDICATES THAT THE PROBABILITY OF MISSING',
&' IS ZERO FOR AT LEAST ONE',/, ' ORIENTATION AND PROB1 ',
&'AND PROB>1 SHOULD NOT BE USED FOR THIS TARGET')
      GO TO 525                              GRID1335
C                                           GRID1345
C      CIRCLE                                GRID1355
C                                           GRID1365
115 ASQ=A**2                                GRID1375
      IF (A-T1Z) 120,120,125                 GRID1385
120 PROB2=0.0                                GRID1395
      PROB1=PI*ASQ/FIN                        GRID1405
      PROB0=1.0-PROB1                         GRID1415
      GO TO 425                              GRID1425
125 IF (A-BALLS) 130,135,135                GRID1435
130 CIM=ARCO(T1Z,A)                         GRID1445
      PROB2=ZAP*(ASQ*CIM-T1Z*SQRT(ASQ-0.25))/FIN GRID1455
      PROB1=PI*ASQ/FIN-2.0*PROB2             GRID1465

```

Code File: ELIPGRD2.For

PROB0=1.0-PROB1-PROB2	GRID1475
GO TO 425	GRID1485
C	GRID1495
C IF THE RADIUS OF THE CIRCLE IS GREATER THAN 0.7071 THE PROBABILITY	GRID1505
C OF MISSING IS ZERO AND PROB1 AND PROB2 ARE SET EQUAL TO 9. AS	GRID1515
C FLAGS	GRID1525
C	GRID1535
135 PROB1=9.0	GRID1545
PROB2=9.0	GRID1555
PROB0=0.0	GRID1565
GO TO 425	GRID1575
C	GRID1585
C ELLIPSE	GRID1595
C	GRID1605
140 B=A*SHAPE	GRID1615
C	GRID1625
C B IS THE RADIUS OF THE CIRCLE IN THE TRANSFORMED NET	GRID1635
C	GRID1645
IF (A-T1Z) 145,145,150	GRID1655
145 PROB1=PI*A*B/FIN	GRID1665
PROB2=0.0	GRID1675
PROB0=1.0-PROB1	GRID1685
GO TO 425	GRID1695
150 IF(ANGLE-0.1) 155,155,160	GRID1705
C	GRID1715
C ALPHA IS THE ANGLE IN RADIANS	GRID1725
C	GRID1735
155 ANGLE=ANGLE+0.1	GRID1745
160 ALPHA=ANGLE/57.295779	GRID1755
CNM=1.0-SHAPE**2	GRID1765
C	GRID1775
C C,D,DJ1,DJ0,DM1,DM0 ARE DISTANCES BETWEEN CIRCLES IN THE	GRID1785
C TRANSFORMED NET	GRID1795
C	GRID1805
C=SQRT(1.0-CNM*COS(ALPHA)**2)	GRID1815
GO TO (170,165,170),NET	GRID1825
165 Y1=3.*SHAPE**2-2.*CNM*SIN(ALPHA)**2-CNM*4.*FIN*SIN(ALPHA)*COS(ALPHA)	GRID1835
1A)	GRID1845
D=SQRT(Y1)*0.5000	GRID1855
GO TO 175	GRID1865
170 D=SQRT(1.0-CNM*SIN(ALPHA)**2)	GRID1875
175 BSQU=B**2	GRID1885
FORN=C*C	GRID1895
HORN=D*D	GRID1905

Code File: ELIPGRD2.For

WINE=FIN*SHAPE	GRID1915
HALFC=C*0.50	GRID1925
HALFD=D*0.50	GRID1935
IF (B-HALFC) 185,185,180	GRID1945
180 EQU=ARCO(HALFC,B)	GRID1955
AREA1=2.0*(BSQU*EQU-HALFC*SQR(BSQU-HALFC**2))	GRID1965
GO TO 190	GRID1975
185 AREA1=0.0	GRID1985
190 IF (B-HALFD) 200,200,195	GRID1995
195 HAI=ARCO(HALFD,B)	GRID2005
AREA2=2.0*(BSQU*HAI-HALFD*SQR(BSQU-HALFD**2))	GRID2015
GO TO 205	GRID2025
200 AREA2=0.0	GRID2035
205 IF (A-BALLS) 210,210,215	GRID2045
210 PROB2=(AREA1+AREA2)/WINE	GRID2055
PROB1=PI*BSQU/WINE-2.0*PROB2	GRID2065
PROB0=1.0-PROB1-PROB2	GRID2075
GO TO 425	GRID2085
215 IF (ANGLE) 220,220,225	GRID2095
220 C=C+0.05	GRID2105
225 CAROL=C*D	GRID2115
* T=ARSIN(WINE/CAROL)	GRID2125
T=ASIN(WINE/CAROL)	
IF (ANGLE) 235,230,235	GRID2135
230 DJ1=SQR(FORN+HORN)	GRID2145
DJ0=5.0	GRID2155
C	GRID2165
C R0 IS THE RADIUS NECESSARY FOR THE TARGET TO BE HIT WITH CERTAINTY	GRID2175
C	GRID2185
R0=DJ1/2.0	GRID2195
GO TO 250	GRID2205
235 I=1.0+(D*COS(T)/C)	GRID2215
IF (I-1) 240,240,245	GRID2225
240 DJ1=SQR((FORN+HORN)-2.0*CAROL*COS(T))	GRID2235
DJ0=5.0	GRID2245
R0=DJ1/(2.0*SIN(T))	GRID2255
GO TO 250	GRID2265
245 XI=I	GRID2275
YI=I-1	GRID2285
DJ1=SQR(XI**2*FORN+HORN-2.0*XI*CAROL*COS(T))	GRID2295
DJ0=SQR(YI**2*FORN+HORN-2.0*YI*CAROL*COS(T))	GRID2305
R0=DJ1*DJ0/(2.0*D*SIN(T))	GRID2315
250 IF (B-R0) 260,255,255	GRID2325
255 PROB1=9.0	GRID2335

Code File: ELIPGRD2.For

PROB2=9.0	GRID2345
PROB0=0.0	GRID2355
GO TO 425	GRID2365
260 HALFJ1=DJ1*0.50	GRID2375
HALFJ0=DJ0*0.50	GRID2385
IF (B-HALFJ1) 270,270,265	GRID2395
265 GRO=ARCO(HALFJ1,B)	GRID2405
AREA3=2.0*(BSQU*GRO-HALFJ1*SQRT(BSQU-HALFJ1**2))	GRID2415
GO TO 275	GRID2425
270 AREA3=0.0	GRID2435
275 IF (B-HALFJ0) 285,285,280	GRID2445
280 PET=ARCO(HALFJ0,B)	GRID2455
AREA4=2.0*(BSQU*PET-HALFJ0*SQRT(BSQU-HALFJ0**2))	GRID2465
GO TO 290	GRID2475
285 AREA4=0.0	GRID2485
290 M=1.0+(2.0*D*COS(T))/C	GRID2495
YM=M-1	GRID2505
XM=M	GRID2515
IF (M-1) 295,295,300	GRID2525
295 DM1=SQRT(FORN+HORN*4.0-4.0*CAROL*COS(T))	GRID2535
DM0=5.0	GRID2545
GO TO 305	GRID2555
300 DM1=SQRT(XM**2*FORN+4.0*HORN-4.0*CAROL*COS(T))	GRID2565
DM0=SQRT(YM**2*FORN+4.0*HORN-4.0*CAROL*COS(T))	GRID2575
305 HALFM1=DM1*0.50	GRID2585
HALFM0=DM0*0.50	GRID2595
IF (HALFM1-DJ1) 310,325,310	GRID2605
310 IF (HALFM1-DJ0) 315,325,315	GRID2615
315 IF (HALFM0-DJ1) 320,325,320	GRID2625
320 IF (HALFM0-DJ0) 330,325,330	GRID2635
325 AREA5=0.0	GRID2645
AREA6=0.0	GRID2655
GO TO 360	GRID2665
330 IF (B-HALFM1) 340,340,335	GRID2675
335 YAM=ARCO(HALFM1,B)	GRID2685
AREA5=2.0*(BSQU*YAM-HALFM1*SQRT(BSQU-HALFM1**2))	GRID2695
GO TO 345	GRID2705
340 AREA5=0.0	GRID2715
345 IF (B-HALFM0) 355,355,350	GRID2725
350 GAME=ARCO(HALFM0,B)	GRID2735
AREA6=2.0*(BSQU*GAME-HALFM0*SQRT(BSQU-HALFM0**2))	GRID2745
GO TO 360	GRID2755
355 AREA6=0.0	GRID2765
360 IF (B-DJ1) 370,370,365	GRID2775

365	SER=ARCO(DJ1,B)	GRID2785
	AREA7=2.0*(BSQU*SER-DJ1*SQR(TBSQU-DJ1**2))	GRID2795
	GO TO 375	GRID2805
370	AREA7=0.0	GRID2815
375	IF (B-DJ0) 385,385,380	GRID2825
380	AQUAR=ARCO(DJ0,B)	GRID2835
	AREA8=2.0*(BSQU*AQUAR-DJ0*SQR(TBSQU-DJ0**2))	GRID2845
	GO TO 390	GRID2855
385	AREA8=0.0	GRID2865
390	IF (B-C) 400,400,395	GRID2875
395	POT=ARCO(C,B)	GRID2885
	AREA9=2.0*(BSQU*POT-C*SQR(TBSQU-FORN))	GRID2895
	GO TO 405	GRID2905
400	AREA9=0.0	GRID2915
405	IF (B-D) 415,415,410	GRID2925
410	TIN=ARCO(D,B)	GRID2935
	AREA10=2.0*(BSQU*TIN-D*SQR(TBSQU-HORN))	GRID2945
	GO TO 420	GRID2955
415	AREA10=0.0	GRID2965
420	PROB2=(AREA1+AREA2+AREA3+AREA4+AREA5+AREA6-AREA7-AREA8-AREA9-AREA10)/WINE	GRID2975
	PROB1=PI*BSQU/WINE-2.0*PROB2-(AREA7+AREA8+AREA9+AREA10)/WINE	GRID2985
	PROB0=1.0-PROB1-PROB2	GRID2995
425	IF (MET) 480,480,430	GRID3005
430	MROT=MROT+1	GRID3015
	IF (PROB1-1.0) 440,440,435	GRID3025
435	IZONK=IWARN	GRID3035
440	SUM1=PROB1+SUM1	GRID3045
	SUM2=PROB2+SUM2	GRID3055
	SUM0=PROB0+SUM0	GRID3065
	IF (IROT-MROT) 445,30,30	GRID3075
445	ANP=MROT	GRID3085
	AVPR1=SUM1/ANP	GRID3095
	AVPR2=SUM2/ANP	GRID3105
	AVPRO=SUM0/ANP	GRID3115
	GO TO (450,460,470),NET	GRID3125
450	WRITE (IPRIN,455) TARGET,SLING,GDSPAC,SHAPE,AVPR1,AVPR2,AVPRO,IZON	GRID3135
	1K	GRID3145
* 455	FORMAT (1H0,A4,3X,11HSQUARE,34X,F5.2,7X,F9.2,5X,F4.2,5X,6HRAN	GRID3155
455	FORMAT (1H,A4,3X,11HSQUARE,34X,F5.2,7X,F9.2,5X,F4.2,5X,6HRAN	GRID3165
	1DOM,5X,3(F6.4,4X),A4)	GRID3175
	GO TO 20	GRID3185
460	WRITE (IPRIN,465) TARGET,SLING,GDSPAC,SHAPE,AVPR1,AVPR2,AVPRO,IZON	GRID3195
	1K	GRID3205

Code File: ELIPGRD2.For

```

* 465 FORMAT (1H0,A4,3X,11HHEXAGONAL ,34X,F5.2,7X,F9.2,5X,F4.2,5X,6HHRANGRID3215
465 FORMAT (1H ,A4,3X,11HHEXAGONAL ,34X,F5.2,7X,F9.2,5X,F4.2,5X,6HRAN
1DOM,5X,3(F6.4,4X),A4) GRID3225
GO TO 20 GRID3235
470 WRITE (IPRIN,475) TARGET,Q,SLING,GDSPAC,XHAPE,AVPR1,AVPR2,AVPRO,IZGRID3245
1ONK GRID3255
* 475 FORMAT (1H0,A4,3X,11HRECTANGULAR,18H X AXIS OF GRID= ,F3.1,8H Y AGRID3265
475 FORMAT (1H ,A4,3X,11HRECTANGULAR,18H X AXIS OF GRID= ,F3.1,8H Y A
1XIS ,5X,F5.2,7X,F9.2,5X,F4.2,5X,6HRANDOM,5X,3(F6.4,4X),A4) GRID3275
GO TO 20 GRID3285
480 IF (PROB1-1.0) 490,490,485 GRID3295
485 IZONK=IWARN GRID3305
490 GO TO (495,505,515),NET GRID3315
495 WRITE (IPRIN,500) TARGET,SLING,GDSPAC,SHAPE,ANGLE,PROB1,PROB2,PROBGRID3325
10,IZONK GRID3335
* 500 FORMAT (1H0,A4,3X,11HSQUARE ,34X,F5.2,7X,F9.2,5X,F4.2,3X,4X,F4GRID3345
500 FORMAT (1H ,A4,3X,11HSQUARE ,34X,F5.2,7X,F9.2,5X,F4.2,3X,4X,F4
1.1,5X,3(F6.4,4X),A4) GRID3355
GO TO 20 GRID3365
505 WRITE (IPRIN,510) TARGET,SLING,GDSPAC,SHAPE,ANGLE,PROB1,PROB2,PROBGRID3375
10,IZONK GRID3385
* 510 FORMAT (1H0,A4,3X,11HHEXAGONAL ,34X,F5.2,7X,F9.2,5X,F4.2,3X,4X,F4GRID3395
510 FORMAT (1H ,A4,3X,11HHEXAGONAL ,34X,F5.2,7X,F9.2,5X,F4.2,3X,4X,F4
1.1,5X,3(F6.4,4X),A4) GRID3405
GO TO 20 GRID3415
515 WRITE (IPRIN,520) TARGET,Q,SLING,GDSPAC,XHAPE,SNGL,PROB1,PROB2,PRGRID3425
1080,IZONK GRID3435
* 520 FORMAT (1H0,A4,3X,11HRECTANGULAR,18H X AXIS OF GRID= ,F3.1,8H Y AGRID3445
520 FORMAT (1H ,A4,3X,11HRECTANGULAR,18H X AXIS OF GRID= ,F3.1,8H Y A
1XIS ,5X,F5.2,7X,F9.2,5X,F4.2,3X,4X,F4.1,5X,3(F6.4,4X),A4) GRID3455
GO TO 20 GRID3465
* 525 STOP GRID3475
525 print *, 'End of run: ORNL/GJ ELIPGRD2.'
STOP GRID3475
END GRID3485

* FUNCTION ARCO(CYB,ERN) ARCO 5
real FUNCTION ARCO(CYB,ERN)
C ARCO 15
C ARCOSINE FUNCTION ARCO 25
C ARCO 35
C NOTE THIS FUNCTION IS AVAILABLE AT SOME COMPUTER FACILITIES ARCO 45
C AND MIGHT THEREFORE REPLACE THIS SECTION OF THE PROGRAM ARCO 55

```

Code File: ELIPGRD2.For

```

C
implicit none
! Arguments:
real CYB
real ERN
! Local variables:
real ZEP
real PEP
real HOPE
real ZOP
real ATP
ZEP=0.0
PEP=3.141592
HOPE=CYB/ERN
ZOP=ABS(HOPE)
IF (HOPE) 5,15,10
5 ZEP=3.141592
10 IF (ZOP-0.5) 15,15,20
15 ARCO=ABS(ZEP-(PEP/2.0*asin(ZOP))) ! changed ARSIN to asin
RETURN
20 ATP=SQRT((1.0-ZOP)/2.0)
ARCO=ABS(ZEP-2.0*asin(ATP)) ! changed ARSIN to asin
RETURN
END

* SUBROUTINE RECT(A,SHAPE,ANGLE,Q,REVK,REVA,REVANG)
* (Note: Argument A is never used by ELIPGRID algorithm.)
SUBROUTINE RECT(SHAPE,ANGLE,Q,REVK,REVA,REVANG)

C
C
C THIS SUBROUTINE REDUCES THE RECTANGULAR POINT NET TO A SQUARE
C POINT NET WITH AN AFFINE TRANSFORMATION
C
implicit none
! Arguments:
real SHAPE,ANGLE,Q,REVK,REVA,REVANG
! Local variables:
real AQ
real SQK
real TIS
real ALPHA
real COAL
real SIAL

```

ARCO 65

ARCO 75

ARCO 85

ARCO 95

ARCO 105

ARCO 115

ARCO 125

ARCO 135

ARCO 145

ARCO 155

ARCO 165

ARCO 175

ARCO 185

ARCO 195

RECT 5

RECT 15

RECT 25

RECT 35

RECT 45

RECT 55

Code File: ELIPGRD2.For

```

      real T

      AQ=Q*Q
      SQK=SHAPE**2
      TIS=AQ*SQK
      IF (ANGLE-0.1) 5,5,10
5  ANGLE=ANGLE+0.1
10 ALPHA=ANGLE/57.295779
      COAL=COS(ALPHA)**2
      SIAL=SIN(ALPHA)**2
      T=SQRT(((1.0-TIS)*COAL-(AQ-SQK)*SIAL)**2+4.0*AQ*(1.0-SQK)**2*SIAL*RECT 145
1COAL)
      REVK=((1.0+TIS)*COAL+(AQ+SQK)*SIAL-T)/(2.0*Q*SHAPE)
      ! Below appears to be an error in the original code.
      ! See (Singer and Wickman 1969, p. 16) for the original math formula.
*  REVANG=(ATAN(2.0*Q*(1.0-SQK)*TAN(ALPHA)/((AQ-SQK)*TAN(ALPHA)**2*TIRECT 175
*  1S-1.0))/2.0)*57.295779
      ! Next line is corrected formula.
      REVANG=(ATAN(2.0*Q*(1.0-SQK)*TAN(ALPHA)/(1.0-TIS-(AQ-SQK)*
&  TAN(ALPHA)**2))/2.0)*57.295779
      REVA=SQRT(SHAPE/(Q*REVK))
      REVANG=ABS(REVANG)
* The following optional code matches (Singer and Wickman 1969, 16)
* and can be used in place of line RECT 205 above. However, no differences
* in output values were seen when testing Singer's 30 rect. grid examples after
* this code was substituted for line RECT 205.
* if (tan(2.0 * REVANG) >= 0.0) then
*   REVANG = abs(REVANG)
* else
*   REVANG = 90.0 - abs(REVANG)
* endif
      RETURN
      END

      !=====
      Subroutine GetInFile(cInFile)
      ! Purpose: Checks for command line file name parameter.
      !           Defaults to ELIPGRD2.In.
      ! Input...: cInFile passed in as blank char*64 variable.
      ! Output..: cInFile is updated.
      ! Errors..: Warns if no param. passed, then uses ELIPGRD1.In.
      implicit none
      ! Arguments
      character*(*) cInFile

```

Code File: ELIPGRD2.For

```

! Local vars
character*64 cCmdLine
call getcl(cCmdLine)
if (nblank(cCmdLine) == 0) then
  ! No file name passed.
  print *, 'Warning...: No file name passed!'
  print *, 'Default...: Will default to ELIPGRD2.In for input.'
  print *, 'Usage....: ELIPGRD2 [<InputFileName> or <Help>]'
  print *,
  cInFile = 'ELIPGRD2.In'
else
  cInFile = charnb(cCmdLine)
endif
return
end
!*** End of Sub: GetInFile()

!=====
Subroutine HelpScreen
! Purpose: Displays help screen.
! Input...: None.
! Output...: None.
! Errors...: None
implicit none
print *, 'ELIPGRID II Help Screen'
print *, '===== '
print *, 'Usage: ELIPGRD2 <InputFileName> or'
print *, '      ELIPGRD2 <Help> or'
print *, '      ELIPGRD2 '//
& 'ELIPGRD2.In is input default file.'
print *,
print *, 'Format of input file is:'
print *, 'Line 1 = Title: (A80)'
print *, 'Line n = data.: (4F10.2,2I4,A4)'
print *, '      F10.4,   F10.4,   F10.4,   F10.4, I4,  I4,  A4'
print *, ' SEMIMAJOR,   SHAPE,   ANGLE,   GDSPAC, NET,  MET, TARGET'
print *, '      1250.0    0.30     6.0     800.0  1   0#187'///
& ' <--Sqr. grid data.'
print *, '      1250.0    0.50     38.0   1060.66  3   0#190'///
& ' <--Rec. grid data.'
print *, '      2.0 <--- 2nd data line for rectangular grid = '//
& 'long/short sides.'
print *, '      1250.0    0.50     0.0   1611.86  2   0#104'///
& ' <--Hex. (triangle) data.'

```

Code File: ELIPGRD2.For

```

      print *,'          9.9      9.9      9.9      9.9  9  9 EOF'//
& ' <--End of file data line.'
      print *, 'Line EOF.....: Use ALL valid values '//
& 'EXCEPT SHAPE is > 1.0.'
      print *,
      print *, 'Output is to file <InputFileName>.Out.'
      print *, 'ELIPGRD2.In is default file name when none passed.'
      return
    end
  !*** End of Sub: HelpScreen()

  !*** End of File: ELIPGRD2.For

```

APPENDIX E

ELIPGRID-PC SOURCE CODE

APPENDIX E
ELIPGRID-PC SOURCE CODE

The first page of this appendix is an index of all user-defined functions in the program. The source code file name is listed for each function. The next two pages are example make and link files. Next are listed two ORNL-developed CA-Clipper® language header files. These have a .CH extension and are listed in alphabetic order. Following these are all the CA-Clipper® source code files. These files have a .PRG extension and are listed in alphabetic order after the main file named EGPCMAIN.PRG.

Program execution begins with function "Main". Execution can be traced from function to function from that point.

Index of all user-defined functions in the ELIPGRID-PC program.

All filenames have a .PRG extension.

Function Name	File Name
AlertBox	EGPCFILE
BlErrc50	EGPCBERR
BoxCenter	EGPCMAIN
ChangeDrive	EGPCMAIN
ChangeDrOrSub	EGPCMAIN
ChangeSubdir	EGPCMAIN
ChooseGrid	EGPCMAIN
ChooseInput	EGPCMAIN
DispTitle	EGPCMAIN
DOS_Prompt	EGPCMAIN
ElipGrid	EGPCFORT
ErrMsgBox	EGPCMAIN
ErrorUDF	EGPCMAIN
ExtrctPath	EGPCFILE
FileInput	EGPCFILE
F10_Key	EGPCMAIN
GetCostGrd	EGPCMAIN
GetFileBox	EGPCFILE
GetFilOutFile	EGPCFILE
GetGridSiz	EGPCMAIN
GetProbHit	EGPCMAIN
GetScnOutFile	EGPCFILE
GetSmallestArea	EGPCMAIN
Help	EGPCHELP
HelpScnN	EGPCHELP
InputFromFile	EGPCMAIN
Main	EGPCMAIN
MenuBox	EGPCMAIN
MenuCenter	EGPCMAIN
NotReadyYet	EGPCMAIN
NumTrim	EGPCMAIN
PastDemoDate	EGPCBERR
ParamHelp	EGPCMAIN
Prob0_Regr	EGPCFILE
Rect	EGPCFORT
SayCenter	EGPCMAIN
SIF_FileInput	EGPCFILE
Subdir	EGPCFILE
WriteData	EGPCMAIN
YN_MsgBox	EGPCMAIN

E-3

```
// File....: EGPC.rmk
// Purpose.: Make file for EGPC program, ELIPGRID-PC.
// Compiler: Clipper 5.2
// Author..: Jim Davidson
// Started.: 08/09/94 from Hotspot.rmk
// Last Mod: 08/09/94
// Compiler Switches below:
// /A = Automatic declaration of publics/privates as memvars.
// /B = Include debugging info., delete this switch for final exe.
// /N = No automatic main proc., must be used for file-wide var declarations.
// /Q = Quiet, suppress line number display.
// /W = Warn of ambiguous var references.
// /V = Treat all ambiguous var references as dynamic vars, not as fields.

"e:\EGPCMain.OBJ": "C:\CLIPPER2\EDITOR\EGPC\EGPCMain.PRG"
e:\Clipper C:\CLIPPER2\EDITOR\EGPC\EGPCMain /A/N/Q/V/W /Oe:\ /Te:\ /Ie:\

"e:\EGPCFile.OBJ": "C:\CLIPPER2\EDITOR\EGPC\EGPCFile.PRG"
e:\Clipper C:\CLIPPER2\EDITOR\EGPC\EGPCFile /A/N/Q/V/W /Oe:\ /Te:\ /Ie:\

"e:\EGPCFort.OBJ": "C:\CLIPPER2\EDITOR\EGPC\EGPCFort.PRG"
e:\Clipper C:\CLIPPER2\EDITOR\EGPC\EGPCFort /A/N/Q/V/W /Oe:\ /Te:\ /Ie:\

"e:\EGPCHelp.OBJ": "C:\CLIPPER2\EDITOR\EGPC\EGPCHelp.PRG"
e:\Clipper C:\CLIPPER2\EDITOR\EGPC\EGPCHelp /A/N/Q/V/W /Oe:\ /Te:\ /Ie:\

"e:\EGPCBErr.OBJ": "C:\CLIPPER2\EDITOR\EGPC\EGPCBErr.PRG"
e:\Clipper C:\CLIPPER2\EDITOR\EGPC\EGPCBErr /A/N/Q/V/W /Oe:\ /Te:\ /Ie:\

"e:\EGPCScrn.OBJ": "C:\CLIPPER2\EDITOR\EGPC\EGPCScrn.PRG"
e:\Clipper C:\CLIPPER2\EDITOR\EGPC\EGPCScrn /A/N/Q/V/W /Oe:\ /Te:\ /Ie:\

"e:\EGPCGrph.OBJ": "C:\CLIPPER2\EDITOR\EGPC\EGPCGrph.PRG"
e:\Clipper C:\CLIPPER2\EDITOR\EGPC\EGPCGrph /A/N/Q/V/W /Oe:\ /Te:\ /Ie:\

"e:\EGPC.EXE": "e:\EGPCMain.OBJ" "e:\EGPCFort.OBJ" "e:\EGPCHelp.OBJ" \
"e:\EGPCFile.OBJ" "e:\EGPCBErr.OBJ" "e:\EGPCScrn.OBJ" "e:\EGPCGrph.OBJ"
e:\blinker @C:\CLIPPER2\EDITOR\EGPC\EGPC.LNK
```

```

# File....: EGPC.Lnk
# Purpose.: Blinker 3.0 response file for EGPC program, ELIPGRID-PC.
# Compiler: Clipper 5.2
# Author...: Jim Davidson
# Started..: 08/09/94 from Hotspot.lnk.
# Last Mod: 09/06/94
# remove comment character below on last link to force full link, smaller exe
blinker incremental off
blinker incremental pad 256          # 128 is default pad size, Manual p. 9-39.
blinker incremental file e:\EGPC.bif
blinker message noblink
blinker demonstration date 1995/06/30
output e:\EGPC
file e:\EGPCMain
# Start of dynamic overlay area for Clipper code.
beginarea
    file e:\EGPCFile
    file e:\EGPCFort
    file e:\EGPCHelp
    file e:\EGPCScrn
    file e:\EGPCGrph
endarea
# Blinker error handler below should not be in overlay area, Manual p. 7-7.
file e:\EGPCBErr          # Blinker error handler for HOTSPOT
lib e:\clipper
# Blinker 3.0 Manual p. 7-20 says "some non-CA-Clipper code must be
# overlaid for demo features to take effect, for instance EXTEND.LIB."
# This link file is similar to Blinker's minimal overlay scheme, CL520Min.Lnk.
beginarea
    lib e:\extend
endarea
lib e:\terminal
lib e:\dbfntx
lib e:\ct                  # Clipper Tools Library

```

```

//=====
// File....: Colors.Ch
// Purpose.: Provides color definitions for Clipper 5.x programs
// By.....: Jim Davidson
// Started.: 07/24/91
// Last Mod: 04/28/94
// Example.: #include "Colors.Ch"
//          setcolor( C_WhtBlk )
//=====

/** Below is example use in a program **
public C_Normal := C_WhtBlk // Normal screen colors
public C_HighLight := C_CynBlu // Color Data highlight
public C_Help := C_CynBrn // Help screens
public C_Error := C_WhtRed // Error screens
// later...
C_Normal := C_WhtBlk // etc. for rest of colors
****/

#define C_BLK_WHT "n/w,w+/n,,w/n" // Black on white

#define C_W_BLK "w/n" // White on black
#define C_WHT_BLK "w+/n,n/w" // Bright white on black
#define C_W_BLU "w/b,w+/n,,gr+/n" // White on blue
#define C_WHT_BLU "w+/b,w+/n,,gr+/n" // Bright white on blue
#define C_WHT_BLK_B "w+*/n,n/w" // Bright blinking white on black
#define C_WHT_RED "w+/r,w+/b,,gr+/n" // Bright white on red
#define C_WHT_MAG "w+/rb,gr+/n,,bg+/n" // Bright white on magenta

#define C_CYN_BLK "bg+/n" // Bright cyan on black
#define C_CYN_BLU "bg+/b,gr+/n,,bg+/n" // Bright cyan on blue
#define C_CYN_BRN "bg+/gr" // Bright cyan on brown
#define C_CYN_MAG "bg+/rb,gr+/n,,bg+/n" // Bright cyan on magenta

#define C_YEL_BLK "gr+/n,gr+/n" // Bright yellow on black
#define C_YEL_BLK_B "gr+*/n" // Bright blinking yellow on black
#define C_YEL_BLU "gr+/b,w+/n,,bg+/n" // Bright yellow on blue
#define C_YEL_MAG "gr+/rb" // Bright yellow on magenta

#define C_RED_BLK "r+/n,w+/n" // Bright red on black
#define C_GRN_BLK "g+/n,w+/n,,bg+/n" // Bright green on black

* Colors for Flipper grf_colors() *
* D refers to #define origin
* LIGHT COLORS *
#define DBLACK 0
#define DBLUE 1
#define DGREEN 2
#define DCYAN 3
#define DRED 4
#define DMAGENTA 5
#define DYELLOW 6
#define DWHITE 7
* DARK COLORS
#define DLGRAY 8
#define DLBLUE 9
#define DLGREEN 10
#define DLCYAN 11
#define DLRED 12
#define DLMAGENTA 13
#define DLYELLOW 14
#define DLWHITE 15

```

```

//=====
// File:          EGPCMax.Ch
// For:           ELIPGRID-PC, EGPC.Exe.
// Purpose:       Provides maximum value #defined constants.
// Author:        Jim Davidson
// Prog Started:  10/03/93
// Last Mod:      08/09/94 changed name from HotSMax.Ch to EGPC.Ch.
//=====

* Max semimajor axis.
#define cMAX_SemiMajor      "9,999.99"

* Max grid cell side, short side for rec. grid.
#define cMAX_GSize         "9,999.99"

* Max hot spot area.
#define cMAX_HotSArea       "9999.99"

* Max total sample area.
#define nMAX_SampleArea     99999999.9
#define cMAX_SampleArea     "99,999,999.9"

* Max sample cost.
#define cMAX_SampleCost     "99,999.99"

* Max number of samples.
#define cMAX_Samples        "999,999"

* Max total cost.
#define cMAX_TotalCost      "999,999,999.99"

* Max rec ratio.
#define cMAX_RecRatio       "99.9"

* Max orientation angle.
#define cMAX_Angle          "99.9"

* Max L to G ratio, semimajor axis/grid size.
#define cMAX_LtoG           "9999.99"

* Max elliptical shape ratio.
#define cMAX_Shape          "9.99"

* Max prob of a hit.
#define cMAX_ProbHit        "999.9999"

* Maximum acres.
#define cMAX_Acres          "9999.99"

* Desired probability.
#define cDESIRD_PROB        "999.9"

* Desired cost.
#define cDESIRD_COST        "99,999,999"

*** End of File: EGPC.Ch

```

E-7

Code File: EGPCMain.Prg

```
//=====
// File.....: EGPCMain.Prg
// Program...: EGPC.Exe
// Purpose...: Main program file for ELIPGRID-PC program.
//             Program provides interface to ELIPGRID (Singer 1972) algorithm.
//             Adds features utilizing Singer's original algorithm.
// Author....: Jim Davidson
// Started...: 10/03/93 as HotSMain.prg.
// Last Mod.: 09/06/94
//
// Files.....: EGPCMain.Prg          This file, main module
//             EGPCBErr.Prg          Blinker 3.0 related error handler
//             EGPCFile.Prg          File input code
//             EGPCFort.Prg          Code translated from ELIPGRID FORTRAN
//             EGPCGrph.Prg          Code to write cost-based graph data
//             EGPCHelp.Prg          Help screen code
//             EGPCScrn.Prg          Screen input/output related code
//
//             Functions are arranged in the files in alphabetical order,
//             although main() is the first function in this file.
//
// Notes.....: Compiler = CA-Clipper 5.2
//             Linker   = BLINKER 3.0
//             Uses CA-Clipper Tools 3.0 library
//
/*=====
Modification history:
-----
03/30/94 On file input, trap rectangular grids with a 1/1 ratio and
         use sq. grid instead. Screen input does not allow 1/1 rect. grids.
03/31/94 Restrict desired % prob.s to 99.9% on input screens. Changed error
         criteria to a slightly smaller value. Worked better near 100%.
04/04/94 Modified triangular grid routine. See EGPCFort.Prg code.
04/06/94 Correction levels finished to level 3.
04/07/94 Added checks for negative P(0) cases. See EGPCFort.Prg code.
04/12/94 Upgraded from Blinker 2.10 to 3.0.
04/13/94 Modified Blinker error file, EGPCBErr.Prg, added for demo exp. date
         use. DOS_Prompt() function now uses Blinker swap function.
04/15/94 All shapes are now restricted to >= 0.05. L/G ratios must be <= 3.
         See EGPCFort.Prg comment.
04/17/94 90° angles now trapped in RECT subroutine. See EGPCFort.Prg.
04/22/94 Search techniques updated. See EGPCScrn.Prg.
04/28/94 Write cost-based graph data option added. See EGPCGrph.Prg.
04/28/94 Validation against Singer's 100 cases, OK.
05/08/94 Only minor (cosmetic) changes made since validation.
05/09/94 This is beta version given at DOE TIE conf. in Kennewick, WA.
05/16/94 Corrected error in "Smallest Hot Spot Hit, Given Grid."
         See EGPCScrn.Prg.
05/17/94 Validated screen input for 67 cases as documented in Scn_Test.Sqr,
         Scn_Test.Tri, and Scn_Test.Rec files in Valid100 subdir.
05/17/94 Upgraded Clipper to version 5.2d. Validation against Singer's 100
         cases, OK.
06/14/94 Upgraded Blinker from 3.0 to 3.01. Validation against Singer's 100
         cases, OK. Input = Test100.In & .SIF; Output = .HSE & .HSS files.
08/09/94 Changed name to ELIPGRID-PC, EGPC.Exe, from Hotspot.
         Other minor changes.
08/11/94 Forced correction level flag to 3, full correction, see EGPCFort.prg.
         Validation against Singer's 100 cases, OK.
08/26/94 Changed all semiminor/semimajor text to semi-minor/semi-major.
         Other minor editorial changes.
09/02/94 Changed formula for the number of samples for a triangular grid.
         See EGPCScrn.Prg and EGPCGrph.Prg files.
09/06/94 Added "Number Samples" column to graphics output data file.
*/
//=====
// Version Info
```

Code File: EGPCMain.Prg

```

#define VER_DATE "09/06/94"

// Include files.
// Clipper supplied include files.
#include "Directry.Ch"           // File info definitions
#include "Inkey.Ch"             // Key definitions
#include "Set.Ch"               // set() definitions
#include "Setcurs.Ch"           // setcursor() related
#include "Box.Ch"               // Box drawing constants

// ORNL developed include files.
#include "Colors.Ch"            // Color definitions

// User-defined command
#define DEFAULT <TheParam> TO <DefaultVal> => ;
  IF (<TheParam> == NIL); <TheParam>:=<DefaultVal>; ENDIF

*=====| Main Module |=====
Function Main()
* Main module of program.
* Initialize local variables.
local nCh      := 1           // Main menu choice
local nLCol    := 0           // Left col chosen by MenuCenter()
local nTRow    := 8           // Main Menu top screen row
local cDOSScreen := savescreen(0,0,24,79)
local nDOSRow  := row()
local lDone    := .F.         // Main Menu loop flag
local cDOSCmdLine := ""       // DOS command line params

* Program wide publics.
public cBasicUnit := "M"      // M = meters, F = feet
* The ElipGrid correction levels below provide ability to test old algorithms.
public nElpGrdCor := 3        // ElipGrid correction level
  // 0 = Crig. ELIPGRID algorithm
  // 1 = Rectangular grid corrections only
  // 2 = Level 1's corrections + triangular grid corrections
  // 3 = Level 2's corrections + angle of 0.0 is not incremented to 0.1
  // Correction level 3 is the default.
public C_Normal      := C_WHT_BLU // Normal screen colors
public C_HighLight   := C_CYN_BLU // Current subdir color
public C_Help        := C_WHT_MAG // Help screens
public C_Error       := C_WHT_RED  // Error screens
public C_Menu1       := C_WHT_MAG  // Menu screen color 1
public C_Menu2       := C_YEL_MAG  // Menu screen color 2

* Get DOS command line parameters.
cDOSCmdLine := upper(dosparam())
if "H" $ cDOSCmdLine
  * Help param. passed.
  ParamHelp(VER_DATE)
  quit
elseif "M" $ cDOSCmdLine
  * Monochrome param. passed.
  * Black on white for LCD screens.
  m->C_Normal      := C_BLK_WHT // Normal screen colors
  m->C_HighLight   := C_WHT_BLK // Current subdir color
  m->C_Help        := C_WHT_BLK // Help screens
  m->C_Error       := C_WHT_BLK // Error screens
  m->C_Menu1       := C_WHT_BLK // Menu screen color 1
  m->C_Menu2       := C_BLK_WHT // Menu screen color 2
endif
if "F" $ cDOSCmdLine
  * Feet param. passed.
  cBasicUnit := "F"
endif

```

Code File: EGPCMain.Prg

```

*--- 08/11/94 Note, JRD ---*
* EGPCFort.prg code now forces level 3 correction all the time.
* Below left in to facilitate any future return to correction levels.
*---
* Determine which level of correction to ELIPGRID algorithm to use.
if "0" $ cDOSCmdLine
  * Use original Singer, 1972 algorithm.
  m->nElpGrdCor := 0
elseif "1" $ cDOSCmdLine
  * Use rect. grid corrections only.
  m->nElpGrdCor := 1
elseif "2" $ cDOSCmdLine
  * Add triangular grid corrections to level "1" corrections.
  m->nElpGrdCor := 2
endif

set escape on
set scoreboard off
set bell off
set confirm on
set wrap on

do while ! lDone
  setcolor(m->C_Normal)           // Reset since looping back
  cls
  dispbox(00,00,04,79, B_DOUBLE_SINGLE)
  setcolor(m->C_Help)
  SayCenter(1, " ORNL ELIPGRID-PC ")
  setcolor(m->C_Normal)
  SayCenter(2, "PC-Based Hot Spot Probability Calculations")
  SayCenter(3, "Version " + VER_DATE)
  @05, 2 say "Current subdirectory: "
  @row(),col() say diskname() + ":" + dirname() color (m->C_HighLight)
  @23, 2 say "F1 key for Help" color(m->C_Help)
  @23,63 say "Esc key to Exit" color(m->C_Help)
  dispbox(06,00,22,79, B_DOUBLE_SINGLE)
  dispbox(22,00,24,79, B_DOUBLE_SINGLE)
  @22,00 say "| "
  @22,79 say "| "

  SayCenter(nTRow-1, "Main Menu")

  nCh := MenuCenter(nTRow, {
    "P Probability of Hitting Hot Spot"      ;;
    "G Grid Size Required, Given Prob."      ;;
    "S Smallest Hot Spot Hit, Given Grid"    ;;
    "C Cost-Based Grid"                     ;;
    "W Write Cost-Based Graph Data"          ;;
    "N New Drive or Subdirectory"           ;;
    "D DOS Prompt"                          ;;
    "Q Quit program..." }, nCh, 1, @nLCol)

  do case
    case nCh == 1
      * P Probability of Hitting Hot Spot
      ChooseInput(nTRow+nCh+1, nLCol+2)
    case nCh == 2
      * G Grid Size Required, Given Prob.
      GetGridSiz(ChooseGrid(nTRow+nCh+1, nLCol+2, m->C_Menu1), VER_DATE)
    case nCh == 3
      * S Smallest Hot Spot Area Hit, Given Grid
      GetSmallestArea(ChooseGrid(nTRow+nCh+1, nLCol+2, m->C_Menu1), VER_DATE)
    case nCh == 4
      * C Cost-Based Grid
      GetCostGrd(ChooseGrid(nTRow+nCh+1, nLCol+2, m->C_Menu1), VER_DATE)
    case nCh == 5
      * W Write Cost-Based Graph Data
      WriteGphData(ChooseGrid(nTRow+nCh+1, nLCol+2, m->C_Menu1), VER_DATE)
  endcase
enddo

```

Code File: EGPCMain.Prg

```

case nCh == 6
    * N New Drive or Subdirectory
    ChangeDrOrSub(nTRow+nCh+1,nLCol+2)
case nCh == 7
    * D DOS Prompt
    DOS_Prompt()
otherwise
    * Q Quit program... (or Esc key)
    lDone := .T.

set color to
restscreen(0,0,24,79,cDOSScreen)
devpos(nDOSRow-1,0)           // -1 makes DOS prompt come in just
* return to DOS               // below last prompt.
return (0)                    // Return 0 to DOS ErrorLabel
*** End of Func: Main()
=====| End of Main Module |=====

*--- Begin Other Functions ---*
*****
Function BoxCenter(nTRow, nRows, nWidth, nType)
* Displays box centered on nRow.
* nType of 1 = double line top, single side, 2 = double all.
* Returns left column.
local nCol := (80-nWidth)/2
default nType to 1
if nType == 1
    MenuBox(nTRow,nCol,nTRow+nRows,nCol+nWidth-1, B_DOUBLE_SINGLE)
else
    MenuBox(nTRow,nCol,nTRow+nRows,nCol+nWidth-1,B_DOUBLE)
endif
return (nCol)
*** End of Func: BoxCenter()

*****
Function ChangeDrive()
* Change current drive.
local cCurrDrive
local lOrgConfrm
local lDone      := .F.
local GetList    := {}
lOrgConfrm       := set(_SET_CONFIRM,.F.)

do while ! lDone
    cls
    cCurrDrive    := diskname()
    MenuBox(02,01,7,67)
    @03,02 say " Change current drive to?"
    @05,03 say "Enter new drive letter" get cCurrDrive pict "!"
    @05,col() say ":"
    read

    if ! diskchange(cCurrDrive)
        * Invalid drive.
        Err_MsgBox(10,"E","Error: Invalid drive.", ;
            "Drive: " + cCurrDrive)
        loop
    else
        lDone := .T.
        loop
    endif
enddo
set(_SET_CONFIRM,lOrgConfrm)

```

Code File: EGPCMain.Prg

```

return (NIL)
*** End of Func: ChangeDrive()

*****
Function ChangeDrOrSub(nTR,nLC)
* Menu for changing drive or subir.
* Input: nTR is top row.
*       nLC is left col.
*
static nLastCh   := 1           // Remembers last choice
local nBR        := nTR + 4
local nRC        := nLC + 32
* Change drive.
ChangeDrive()
if lastkey() != K_ESC
    * Change subdirectory.
    ChangeSubdir(10)
endif
return (NIL)
*** End of Func: ChangeDrOrSub()

*****
Function ChangeSubdir(nTR)
* Changes current subdir.
local cCurrSubdir := ""
local cCurrDrive  := ""
local cDOSCmd     := ""
local lDone       := .F.
local GetList     := ()

cCurrDrive      := diskname()
cCurrSubdir     := dirname()

do while ! lDone
    cCurrSubdir := padr(cCurrSubdir, 64)
    MenuBox(nTR,1,nTR+6,66)
    @nTR+1,03 say "Change current subdirectory. Must be on drive " + ;
        diskname() + ":..."
    @nTR+3,03 say "Change to " + cCurrDrive + ":"
    @nTR+3,col() get cCurrSubdir pict "a50!"
    @nTR+5,03 say "Current path: " + diskname() + ":" + dirname()
    keyboard chr(K_END)
    read
    cCurrSubdir := alltrim(cCurrSubdir)

    if lastkey() == K_ESC
        * Esc key abort.
        lDone := .T.
        loop
    elseif ":" $ cCurrSubdir
        * Error, drive name entered.
        Err_MsgBox(nTR+6,"E","Error: Drive name entered.", ;
            "Note.: This option only changes subdirectories" + ;
            " on current drive.", ;
            " Use change drive option for new drive.")
        loop
    elseif ! subdir(cCurrSubdir)
        * Error, invalid subdirectory.
        Err_MsgBox(nTR+6,"E","Error: Invalid subdirectory.", ;
            "Path.: " + cCurrDrive + ":" + cCurrSubdir )
        loop
    else
        * Do the DOS CD command.
        if len(cCurrSubdir) > 3 .and. right(cCurrSubdir,1) == "\"
            * If not root subdir and we have trailing "\", remove it.
            * Will mess up DOS CD command.

```

Code File: EGPCMain.Prg

```

        cCurrSubdir := left(cCurrSubdir,len(cCurrSubdir)-1)
    endif
    * If no characters in subdir name, default to root of current drive.
    if empty(cCurrSubdir)
        cCurrSubdir := "\"
    endif

    * Form the command and do the work.
    cDOSCmd := "CD " + cCurrSubdir
    run (cDOSCmd)

    lDone := .T.
    loop
    endif
enddo
return (NIL)
*** End of Func: ChangeSubdir()

*****
Function ChooseGrid(nTR,nLC, cColor)
* Choose grid type desired.
* Input:   Top row, left col., menu color.
* Returns: Grid type as "S", "R", or "T", for square, rect., or triangle.
*         NIL returned if Esc pressed.
static nLastGrid := 1          // Remembers last grid type chosen
local nBR       := nTR + 5
local nRC       := nLC + 30
local cGridType := NIL
local cOrgColor := ""
default cColor to m->C_Normal
cOrgColor      := setcolor(cColor)

MenuBox(nTR,nLC,nBR,nRC)
@nTR+1,nLC+2 say "Choose Grid Type "
@nTR+2,nLC+2 prompt " Square "
@nTR+3,nLC+2 prompt " Rectangular "
@nTR+4,nLC+2 prompt " Triangular "
menu to nLastGrid
if nLastGrid == 1
    * Square grid.
    cGridType := "S"
elseif nLastGrid == 2
    * Rect. grid.
    cGridType := "R"
elseif nLastGrid == 3
    * Triangular grid.
    cGridType := "T"
endif
setcolor(cOrgColor)
return (cGridType)
*** End of File: ChooseGrid()

*****
Function ChooseInput(nTR,nLC)
* Choose input from Screen/File.
* Input: nTR is top row.
*        nLC is left col.
*
static nLastInput := 1          // Remembers last input type chosen
local nBR       := nTR + 4
local nRC       := nLC + 30
local cOrgColor := setcolor(m->C_Menu1)

MenuBox(nTR,nLC,nBR,nRC)
@nTR+1,nLC+2 say "Enter Data From?"
@nTR+2,nLC+2 prompt " S Screen Input "

```

Code File: EGPCMain.Prg

```

@nTR+3,nLC+2 prompt " F File Input "
menu to nLastInput

setcolor(m->c_Normal)
if nLastInput == 1
  * Screen input.
  GetProbHit(ChooseGrid(nTR+2,nLC+1), VER_DATE)
elseif nLastInput == 2
  * File input.
  InputFromFile(nTR+2,nLC+1)
endif
setcolor(cOrgColor)
return (NIL)
*** End of File: ChooseInput()

*****
Function DispTitle(cGridType, cOption, cOutFile, lOutFile)
* Displays correct title for input screen. Also displays output file, if any.
* Input:  cGridType = "S", "R", or "T" for Square, Rec., or Tri. grids.
*
      cOption  = "P"  for Probability of Hitting Hot Spot
              = "G"  for Grid Size Required, Given Prob.
              = "S"  for Smallest Hot Spot Hit, Given Grid
              = "C"  for Cost-Based Grid
              = "W"  for Write graph data

      local cTitle1 := ""
      local cTitle2 := ""
      local cUnit   := ""
      * Get 1st portion of title.
      if cOption == "P"
        cTitle1 := "Determine Probability of Hitting Hot Spot for "
      elseif cOption == "G"
        cTitle1 := "Determine Size of "
      elseif cOption == "C"
        cTitle1 := "Determine Size of Cost-Based "
      elseif cOption == "W"
        cTitle1 := "Write Cost-Based Graph Data for "
      endif

      if cGridType == "S"
        * Square grid.
        if cOption == "PGCW"
          cTitle2 := "Square Grid in "
        elseif cOption == "S"
          cTitle2 := "Smallest Hot Spot for Square Grid in "
        endif
      elseif cGridType == "R"
        * Rectangular grid.
        if cOption == "PGCW"
          cTitle2 := "Rectangular Grid in "
        elseif cOption == "S"
          cTitle2 := "Smallest Hot Spot for Rectangular Grid in "
        endif
      elseif cGridType == "T"
        * Triangular grid.
        if cOption == "PGCW"
          cTitle2 := "Triangular Grid in "
        elseif cOption == "S"
          cTitle2 := "Smallest Hot Spot for Triangular Grid in "
        endif
      endif
      cUnit := if(m->cBasicUnit=="F","Feet","Meters")
      cls
      @0,0 to 4,79 double
      @5,0 to 24,79
      SayCenter(1,cTitle1+cTitle2+cUnit)

```

Code File: EGPCMain.Prg

```

if ! (cOption == "W")
    SayCenter(2, "See Gilbert Chapter 10 for general information.")
else
    SayCenter(2, "Writes data input file (ASCII format) for spreadsheets and "+;
    "graphics programs.")
endif
if ! outFile
    @03,02 say "Current Output File: " + cOutFile color(m->C_HighLight)
else
    @03,02 say "Current Output File: None chosen." color(m->C_HighLight)
endif
return (NIL)
*** End of Func: DispTitle()

*****
Function DOS_Prompt()
* Shell to DOS.
* Returns: NIL
local nMajErr := 0           // Major error code
local nMinErr := 0           // Minor error code
local lSuccess := .F.
set color to
cls
setcolor(m->C_Help)
scroll(0,0,4,79)
@0,0 to 4,79
@1,2 say "Type EXIT at DOS prompt to return to ELIPGRID-PC program."
@3,2 say "The DOS MEM command will give largest executable program size."
* Blinker 3.0 command, swpruncmd("",0,"",""),
* leaves much more memory free than Clipper run command.
* Default swpruncmd() parameters: run command.com, free as much mem as possible,
* leave current path the default, swap to current path.
lSuccess := swpruncmd("",0,"", "")
if ! lSuccess
    scroll(0,0,2,79)
    ? "DOS access failed."
    nMajErr := swprerrmaj()
    nMinErr := swprerrmin()
    ? "Blinker major, minor error codes: ", NumTrim(nMajErr)+",", NumTrim(nMinErr)
    ? "Press a key to continue..."
    inkey(0)
endif
return(NIL)
*** End of Func: DOS_Prompt()

*****
Function Err_MsgBox(nTR, cType, cLin1, cLin2, cLin3)
* Generic error or msg box. Defaults to error box.
* Displays up to 3 lines + Press key msg and waits for keypress.
* Returns: NIL
local cTmpScn      := ""
local lDispMsg     := .T.
local nMaxLineWidth := 0
local nWidth       := 0
local cOrgClr      := ""
local nLC          := 0
local nBR          := 0
local nRC          := 0
local nLines       := 0
local nCurRow      := 0
local nCurCol      := 0
default cType to "E"           // Default to error box

* Set box color.
if upper(cType) == "E"
    cOrgClr := setcolor(m->C_Error)

```

Code File: EGPCMain.Prg

```

else
  cOrgClr := setcolor(m->c_Help)
endif

* Get current cursor pos.
nCurRow := row()
nCurCol := col()

if (valtype(cLin3) == "C")
  * 3 lines to display
  nBR := nTR + 4 + 3           // 4 lines for misc. + 3 msg lines
  nMaxLineWdth := max( max(len(cLin1), len(cLin2)), len(cLin3) )
  nLines := 3
elseif (valtype(cLin2) == "C")
  * 2 lines to display
  nBR := nTR + 4 + 2           // 4 lines for misc. + 2 msg lines
  nMaxLineWdth := max(len(cLin1), len(cLin2))
  nLines := 2
elseif (valtype(cLin1) == "C")
  * 1 line to display
  nBR := nTR + 4 + 1
  nMaxLineWdth := len(cLin1)
  nLines := 1
else
  * Incorrect params. passed
  lDispMsg := .F.
endif

* Display message.
if (lDispMsg)
  nMaxLineWdth := max( nMaxLineWdth, len("Press a key to continue...") )
  nWidth := 4 + nMaxLineWdth   // 2 lines/blanks + largest line
  nLC := (79 - nWidth)/2       // center
  nRC := nLC + nWidth - 1
  cTmpScn := savescreen(nTR, nLC, nBR+1, nRC+1)
  MenuBox(nTR, nLC, nBR, nRC)
  if (nLines >= 1)
    @nTR+2, nLC + 2 say cLin1
  endif
  if (nLines >= 2)
    @nTR+3, nLC + 2 say cLin2
  endif
  if (nLines == 3)
    @nTR+4, nLC + 2 say cLin3
  endif
  @nBR-1, nLC + 2 say "Press a key to continue..."
  tone(440,1)
  inkey(0)
  restscreen(nTR, nLC, nBR+1, nRC+1, cTmpScn)
else
  @0,0
  @0,0 say " Err_MsgBox() error: Check parameters.  Press a key to return... "
  inkey(0)
endif (lDispMsg)
setcolor( cOrgClr )
@nCurRow, nCurCol say ""
return (NIL)
*** End of Func: Err_MsgBox()

*****
Function ErrorUDF(lPassTest, cErrorMsg, nFldLen)
* Generic error routine for @ say/get valid clauses.
* lPassTest: Logic flag for--pass test?
* cErrorMsg: Message to display
* nFldLen...: Length of get field--as picture specifies for numeric.
* Returns...: .F. if lPassTest == .F., else just returns .T.

```

Code File: EGPCMain.Prg

```

local CurGetName := readvar()           // Name of current get variable
local nTR        := row() + 1          // Current row + 1 for error box
local nBR        := nTR + 3
local nLC        := col() - nFldLen    // Current col is end of get field
local nRC        := nLC + len(cErrorMsg) + 1
local cTmpScr    := savescreen(nTR, nLC, nBR, nRC)
local cCurClr   := setcolor(m->c_Error)
local lRtnVal    := .F.

if ! lPassTest
    * Invalid input failed valid test, display error box.
    scroll(nTR, nLC, nBR, nRC)
    @nTR,nLC to nBR,nRC
    @nTR+1, nLC+1 say cErrorMsg
    @nTR+2, nLC+1 say "Press a key..."
    tone(440,1)
    inkey(0)
    restscren(nTR, nLC, nBR, nRC, cTmpScr)
    lRtnVal := .F.
else
    lRtnVal := .T.
endif
setcolor(cCurClr)
return (lRtnVal)
*** End of Func: ErrorUDF()

*****
Function InputFromFile(nTR,nLC)
* Get input data from ELIPGRID type file or SIF type file.
* Input: nTR = Top row for box.
*       nLC = Left col for box.
static nLastType := 1           // Remembers last file type chosen
static nLastFileE := 1          // Remembers last ELIPGRID type file ch
static nLastFileS := 1          // Remembers last SIF type file ch
local nBR        := nTR + 4
local nRC        := nLC + 30

MenuBox(nTR,nLC,nBR,nRC)
@nTR+1,nLC+2 say "Choose Input File Format"
@nTR+2,nLC+2 prompt "ELIPGRID Type Format"
@nTR+3,nLC+2 prompt "SIF Type Format"
menu to nLastType
if nLastType == 1
    * ELIPGRID Format.
    FileInput(nTR+2,nLC+1,@nLastFileE,VER_DATE) // Pass nLastFileE by refer.
elseif nLastType == 2
    * SIF Format.
    SIF_FileInput(nTR+2,nLC+1,@nLastFileS,VER_DATE) // Pass nLastFileS by refer.
endif
return (NIL)
*** End of File: InputFromFile()

*****
Function MenuBox(nTR,nLC,nBR,nRC, cSides, lShadow)
* Draw box sides for a menu.
* cSides defaults to double top, single sides.
* cSides could be defined constants from from Box.Ch.
* lShadow defaults to .T.
local cOrgColor := setcolor()
default cSides to B_DOUBLE_SINGLE
default lShadow to .T.
if lShadow
    set color to
    scroll(nTR+1,nLC+1,nBR+1,nRC+1)
    setcolor(cOrgColor)
    scroll(nTR,nLC,nBR,nRC)

```

Code File: EGPCMain.Prg

```

endif
dispbox(nTR,nLC,nBR,nRC, cSides)
return (NIL)
*** End of Func: MenuBox()

*****
Function MenuCenter(nRow, aPrmpts, nChoice, nType,nLeftCol)
* Displays centered menu of prompts.
* Returns menu choice.
* Returns left col of menu when nLeftCol is passed in by reference.
local nLong := 0
local nPLen := 0
local nPrmpts := len(aPrmpts)
local nLCol := 0
local i
default nChoice to 1 // 1st choice to highlight
default nType to 1 // 1=Double top, single side,2=all double

* Find longest prompt, set nLong.
for i = 1 to len(aPrmpts)
    nPLen := len(aPrmpts[i])
    nLong := if(nPLen > nLong, nPLen, nLong)
next i
nLCol := BoxCenter(nRow,nPrmpts+1,nLong+4,nType)
nLeftCol := nLCol
for i = 1 to nPrmpts
    @nRow+i,nLCol+1 prompt " " + padr(aPrmpts[i],nLong) + " "
next i
menu to nChoice
return (nChoice)
*** End of Func: MenuCenter()

*****
Function NotReadyYet(cMsg)
* Not ready yet msg.
save screen
cls
@0,0 to 5,79
@ 2, 2 say cMsg + " option is not ready yet."
@ 4, 2 say "Press a key to continue..."
inkey(0)
return (NIL)
*** End of Func: NotReadyYet()

*****
Function NumTrim(nNum)
* Returns nNum in str form trimmed.
local cNumStr := alltrim(str(nNum))
return (cNumStr)
*** End of Func: NumTrim()

*****
Function SayCenter(nRow, cMsg)
* Displays cMsg on centered nRow.
local nCol := (80-len(cMsg))/2
@nRow,nCol say cMsg
return (NIL)
*** End of Func: SayCenter()

*****
Function YN_MsgBox(cMsg)
* Yes/No Message Box, Displays Msg,
* Returns .T. if "y" or "Y" pressed.
local cAns
local cOldClr := setcolor(m->C_Help)
local lRtnVal := .F.

```

Code File: EGPCMain.Prg

```
local TempScr
save screen to TempScr
MenuBox(5,5,10,50)
@ 7,7 say cMsg
cAns := upper(chr(inkey(0)))
if cAns == "Y" .or. (lastkey() == K_ESC)
    lRtnVal := .T.
endif
setcolor(cOldClr)
restore screen from TempScr
return (lRtnVal)
*** End of Func: YN_MsgBox()

*** End of File: EGPCMain.Prg
```

Code File: EGPCBErr.Prg

```

/*****
*
* Program : EGPCBErr.Prg
*          : Blinker/CA-Clipper error handling for ORNL/GJ ELIPGRID-PC.
* Started : 04/13/94 from BLERRC50.PRG supplied by Blinker 3.0.
* Last Mod : 08/09/94, JRD
*
* Compiler : CA-Clipper 5.2
* Linker   : Blinker 3.0
*
* Note    : Overlaying of this file is NOT recommended, because
*          : if a severe error occurs, it may be impossible to
*          : load this error handler into memory, in which case
*          : the error will never be reported, making debugging
*          : difficult.
*/

#command ? <list,...>  => ?? Chr(13) + Chr(10) ; ?? <list>
#command ?? <list,...> => OutErr(<list>)

function BLErrc50()
local bBliError, bOldErrBlk, nErrCode, oErr, lUseErrBlk, i
public lInErr

// First check we're not in a multiple error situation
// (likely cause of multiple error is an error loading
// an overlay while in an error situation)

if m->lInErr
* ? "Blinker error : "
* ?? BliErrNum()
* ? "(Multiple errors occurred while in error handler)"
quit
endif

m->lInErr := .T. // In an error
lUseErrBlk = .t. // Use BLINKER error block

oErr := ErrorNew() // Create error object

nErrCode      := BliErrNum() // Blinker Error Number
oErr:subsystem := [BLINKER] // Failing Subsystem name
oErr:subCode   := nErrCode // Blinker error number
oErr:canRetry  := .F. // Not Retryable
oErr:severity  := 3 // Maximum severity

do case
case nErrCode = 1201
oErr:description := [unable to find overlay file ]+BliErrPrm()+[ in current path]
oErr:filename := BliErrPrm()
case nErrCode = 1202
oErr:description := [DOS read error in file ]+BliErrPrm()
oErr:filename := BliErrPrm()
case nErrCode = 1203
oErr:description := [file ]+BliErrPrm()+[ is not a valid .EXE file]
oErr:filename := BliErrPrm()
case nErrCode = 1204
oErr:description := [overlay file ] + BliErrPrm() + [ does not match the .EXE file]
oErr:filename := BliErrPrm()
case nErrCode = 1205
oErr:description := [not enough memory to load procedure]
case nErrCode = 1206
oErr:description := [maximum procedure nesting depth exceeded]
lUseErrBlk = .f.
case nErrCode = 1207
oErr:description := [demonstration calls limit exceeded]

```

Code File: EGPCBErr.Prg

```

        lUseErrBlk = .f.
    case nERRCode = 1208
        // New ELIPGRID-PC code, 04/13/94, JRD
        PastDemoDate()
        /* original code
        oErr:description := [demonstration date limit exceeded]
        lUseErrBlk = .f.
        */
    case nERRCode = 1209
        oErr:description := [demonstration time limit exceeded]
        lUseErrBlk = .f.
    case nERRCode = 1210
        oErr:description := [overlay has been prematurely freed]
    case nERRCode = 1211
        oErr:description := [overlay manager internal stack overflow]
    case nERRCode = 1212
        oErr:description := [Overlay Opsize exceeded - increase Opsize]
    case nERRCode = 1213
        oErr:description := [attempt to call DEFINED routine]
        lUseErrBlk = .f.
    case nERRCode = 1214
        oErr:description := [error accessing EMS overlay cache]
    case nERRCode = 1215
        oErr:description := [error accessing XMS overlay cache]
    case nERRCode = 1216
        oErr:description := [overlay manager unable to resume]
    case nERRCode = 1217
        oErr:description := [overlay vector corrupted during execution]
    otherwise
        oErr:description := [unknown BLINKER error]
end case

for i = 1 to 60
    ? ""
next
// Cheap substitute for CLS
// So that it does not
// Force in the screen drivers

if lUseErrBlk
    bOldErrBlk := ErrorBlock((|e|BliError(e))) // Install new error handler
else
    ?? "Blinker error" + str (oErr:subCode,5)
    ?? " : ", oErr:description
    // Just in case error handler
    // fails
endif

if (ErrorBlock() <> NIL)
    eval(Errorblock(),oErr) // Evaluate the error block
else
    quit
end

if lUseErrBlk
    Errorblock(bOldErrBlk) // Restore the previous handler
endif

w->lInErr := .f.

return (nil)

// Blinker error handler

Static Function BliError(e)
local i

    ? "Error          : "

    if ( !Empty(e:subsystem()) )

```

Code File: EGPCBErr.Prg

```

    ?? e: subsystem() + "/" + Ltrim(Str(e: subCode()))
end
if ( !Empty(e: description()) )
    ? "Description   : " + e: description()
end
if ( !Empty(e: filename()) )
    ? "Filename      : " + e: filename()
end
? ""
? "Call Trace"
i := 3
while ( !Empty(ProcName(i)) )
    ? "Called from   : ", Left(ProcName(i)+SPACE(20),20) + ;
      ? "            + Substr(SPACE(7)+Str(ProcLine(i)),-7) + " "
    i++
end
? ""
ERRORLEVEL(1)
QUIT                                     // terminate application
return (nil)

*****
Function PastDemoDate()
* New code for ELIPGRID-PC past demo date.
local cDemoDate := blidemdte()
cDemoDate := substr(cDemoDate,5,2)+"/"+right(cDemoDate,2)+"/"+left(cDemoDate,4)
?? repli("-", 80)
?? " Message: ELIPGRID-PC is past the expiration date of " + cDemoDate
? " Date...: " + dtoc(date())
? " Note...: This version of ELIPGRID-PC is not intended for indefinite use."
? "          It is a beta version for testing and validation."
? " Contact: Jim Davidson"
? "          ORNL/GJ"
? "          (303) 248-6259"
? "          for more information."
? repli("-", 80)
// terminate application
errorlevel(1)
quit
return (NIL)
*** End of Func: PastDemoDate()

*** End of File: EGPCBErr.Prg

```

Code File: EGPCFile.Prg

```

//=====
// File:          EGPCFile.Prg
// For:           EGPC.Exe, ELIPGRID-PC program.
// Purpose:       Provides file input code.
// Author:        Jim Davidson
// Prog Started:  10/03/93
// Last Mod:      08/26/94
// Note:          Functions are arranged in alphabetical order.
// Modifications since validation with Singer's 100 cases:
// 04/15/94 Added shape, L/G restrictions. Shapes must be >= 0.05. L/G ratios
//           must be <= 3.0. See EGPCFort.Prg for comments.
// 04/28/94 Added AlertBox() to replace alert(). Works well with mono screens.
// 08/09/94 Name changed from HotSFile.prg to EGPCFile.prg.
//=====

// Include files
#include "Inkey.Ch"           // key definitions
#include "Directry.ch"        // File info definitions

// User-defined command
#define DEFAULT <TheParam> TO <DefaultVal> => ;
    IF (<TheParam> == NIL); <TheParam>:=<DefaultVal>; ENDIF

*****
Function AlertBox(nTR, acOptions, nLin1, nLin2, nLin3)
* Substitute for alert() function. Alert() does not obey color settings.
* AlertBox obeys current color setting. Alert() is hard to read on LCD screens.
* Lines 2 and 3 are optional.
* Returns: Esc = 0, else number of array element of acOptions chosen.
local cTmpScn      := ""
local lDispMsg     := .T.
local nMaxLineWdth := 0
local nPrmptWdth   := 0
local nWidth       := 0
local cOrgClr      := ""
local nLC          := 0
local nBR          := 0
local nRC          := 0
local nLines       := 0
local nCurRow     := 0
local nCurCol     := 0
local nNumOps      := len(acOptions)
local nCurOp      := 1
local nOpCol       := 0
local nRtnVal      := 0

* Set box color.
cOrgClr := setcolor(m->C_Error)

* Get current cursor pos.
nCurRow := row()
nCurCol := col()

if (valtype(nLin3) == "C")
    * 3 lines to display
    nBR := nTR + 4 + 3           // 4 lines for misc. + 3 msg lines
    nMaxLineWdth := max( max(len(nLin1), len(nLin2)), len(nLin3) )
    nLines := 3
elseif (valtype(nLin2) == "C")
    * 2 lines to display
    nBR := nTR + 4 + 2           // 4 lines for misc. + 2 msg lines
    nMaxLineWdth := max(len(nLin1), len(nLin2))
    nLines := 2
elseif (valtype(nLin1) == "C")
    * 1 line to display
    nBR := nTR + 4 + 1

```

Code File: EGPCFile.Prg

```

nMaxLineWidth := len(nLin1)
nLines := 1
else
  * Incorrect params. passed
  (DispMsg := .f.
endif

* Display message.
if (lDispMsg)
  * Get total width of the prompts plus inner spacing.
  for nCurOp = 1 to nNumOps
    nPrmptWidth := nPrmptWidth + len(acOptions[nCurOp])
  next nCurOp
  nPrmptWidth := nPrmptWidth + 3 * (nNumOps-1)

  * Determine overall width of box.
  nMaxLineWidth := max(nMaxLineWidth, nPrmptWidth)
  nWidth := 4 + nMaxLineWidth
  nLC := (79 - nWidth)/2          // center
  nRC := nLC + nWidth - 1
  cTmpScn := savescreen(nTR, nLC, nBR+1, nRC+1)
  MenuBox(nTR, nLC, nBR, nRC)
  if (nLines >= 1)
    @nTR+2, nLC + 2 say nLin1
  endif
  if (nLines >= 2)
    @nTR+3, nLC + 2 say nLin2
  endif
  if (nLines == 3)
    @nTR+4, nLC + 2 say nLin3
  endif

  * Display and get desired menu option.
  for nCurOp = 1 to nNumOps
    if nCurOp == 1
      nOpCol := nLC + 2
    else
      nOpCol := nOpCol + len(acOptions[nCurOp-1]) + 3
    endif
    @nBR-1, nOpCol prompt acOptions[nCurOp]
  next nCurOp
  tone(440,0.3)
  menu to nRtnVal

  restscreen(nTR, nLC, nBR+1, nRC+1, cTmpScn)
else
  @0,0
  @0,0 say " AlertBox() error: Check parameters.  Press a key to return... "
  inkey(0)
endif (lDispMsg)
setcolor( cOrgClr )
@nCurRow, nCurCol say ""
return (nRtnVal)
*** End of Func: AlertBox()

*****
Function: ExtrctPath(cPathFileN)
* Extract path from cPathFileN.
* Example: ExtrctPath("D:\file.ext") ==> "D:\"
* Based on Environ.prg fuction FilePath() supplied by Clipper.
local nBkSlshPos := 0
local cPath := ""
nBkSlshPos := rat("\", cPathFileN)
if nBkSlshPos == 0
  cPath := ""
else

```

Code File: EGPCFile.Prg

```

    cPath := substr(cPathFileW, 1, nBkSlashPos)
endif
return (cPath)
*** End of Func: ExtractPath()

*****
Function FileInput(nTR, nLC, nInitFile, cVerDate)
* Get data from ELIPGRID format input file.
* Writes P(0) output to cInFile.Out.
* Input:   nTR is top row for file selection box,
*          nLC is left col.
*          nInitFile is initial file to highlight.
*          cVerDate is program version date.
* Returns: NIL
local lReadInsert := readinsert(.T.)
local cInFile      := ""           // Input file name
local cOutFile     := ""
local cDataLine    := ""
local nSemiMajor   := 0
local nShape       := 0           // Ellipse semi-major/semi-minor axis
local nAngle       := 0
local nGSize       := 0           // Grid size, for Rec. grids, short side
local nGTyp        := 0           // Grid type, 1=Sq., 2=Tri., 3=Rec.
local nOrientn     := 0           // Specific angle or "random",
local cTrgtID      := ""         // if nOrientn > 0 use "random" angles
local nInputLine   := 0           // Current file input line
local nRecRatio    := 0           // Rec. grid long side/short side ratio
local nLines       := 0           // Lines in input file
local nProbNoHit   := 0
local nProbSum     := 0.0         // Used for "random" angle case
local nCrntAngle   := 0           // Used for "random" angle case
local nLrgstAngle  := 0           // Used for "random" angle case
local cFileText    := ""
local lProceed     := .T.
local GetList      := {}         // Stops compiler warnings

cInFile := GetFileBox(nTR,nLC,,,"*.","",nInitFile)
cFileText := memoread(cInFile)
nLines := mlcount(cFileText)
if lastkey() == K_ESC .or. empty(cInFile)
    * Just return, if Esc key pressed.
elseif nLines < 3
    Err_MsgBox(10,"E","Error: Less than 3 lines in file.", ;
        "File.: " + cInFile, ;
        "Need (1) Title line, (2) Data line, (3) EOF line.")
else
    cOutFile := GetFilOutFile(cInFile, @lProceed)

    if lProceed
        * Do the work!
        cls
        set alternate to (cOutFile)
        set alternate on
        ?? "Output from ORNL/GJ ELIPGRID-PC Program Version: " + cVerDate
        ? "File Name.: " + cOutFile
        ? "Created on: " + dtoc(date())
        ? "Input file: " + cInFile + " using ELIPGRID format."
        ? "Title line: " + memoline(cFileText,,1)
        ?
        ? "Target  Grid Type      Semi-major Axis   Gridspace   Shape   Angle   Prob(0)"
        ? "          in Relative Units  in Orig Units"

        * Get data lines
        nInputLine := 2           // Skip title line
        do while nInputLine <= nLines
            cDataLine := memoline(cFileText,,nInputLine)

```

Code File: EGPCFile.Prg

```

nSemiMajor := val(substr(cDataLine, 1,10))
nShape      := val(substr(cDataLine,11,10))
nAngle      := val(substr(cDataLine,21,10))
nGSize      := val(substr(cDataLine,31,10))
nGTyp       := val(substr(cDataLine,41, 4))
nOrientn    := val(substr(cDataLine,45, 4))
cTrgtID     := substr(cDataLine,49, 4)

if nGTyp == 3
  * If rect. grid, get long/short ratio from next line.
  nInputLine++
  cDataLine := memoline(cFileText,,nInputLine)
  nRecRatio := val(substr(cDataLine, 1,10))
  if nRecRatio == 1.0
    * Trap for a rect. grid with a long/short side ratio of 1.0.
    * Use a sq. grid since problems can develop using rect. grid in
    * certain cases. This problem found in tech. review by J. Wilson.
    nGTyp := 1
  endif
endif

if nShape > 1.0 .or. nShape < 0.05 .or. nSemiMajor/nGSize > 3.0
  * EOF or error in shape or L/G ratio > 3.
  exit // Exit do while loop
endif

*-----| Calculate probability of no hit, P(0) |-----*
if nOrientn <= 0.0
  * Calcualte for a single angle.
  nProbNoHit := ElipGrid(nSemiMajor,nShape,nAngle,nGSize,nGTyp, ;
    nRecRatio)
else
  * Calculate for average of multiple angles,
  * i.e., "random" choice in Singer's 1972 ELIPGRID.
  if nGTyp == 1
    nLrgstAngle := 45
  elseif nGTyp == 2
    * For triangular grid (hexagon).
    nLrgstAngle := 30
  elseif nGTyp == 3
    * For rectangular grid.
    nLrgstAngle := 90
  endif
  * Sum up multiple angles results.
  nProbSum := 0.0
  for nCrntAngle = 0 to nLrgstAngle
    nProbNoHit := ElipGrid(nSemiMajor,nShape,nCrntAngle,nGSize, ;
      nGTyp, nRecRatio)
    nProbSum := nProbSum + nProbNoHit
  next nCrntAngle

  * Calculate average.
  nProbNoHit := nProbSum/(nLrgstAngle+1)
endif
*-----*

* Print a line of data.
? pabr(cTrgtID,8)
if nGTyp == 1
  ?? "Square      " + space(8)
elseif nGTyp == 3
  ?? "Rectangular, " + trans(nRecRatio,"99.9") + "/1  "
elseif nGTyp == 2
  ?? "Triangular  " + space(8)
endif

```

Code File: EGPCFile.Prg

```

* Print data fields.
?? trans(nSemiMajor/nGSize,"9999.9999")+ space(6) + ;
   trans(nGSize,"9999.99")           + space(7) + ;
   trans(nShape,"9.99")               + space(3) + ;
   if(nOrientn > 0,"Random",trans(nAngle,"99.9"+"  ")) + ;
   space(1) + ;
   trans(nProbNoHit,"999.9999")
* Increment line index.
nInputLine++
enddo
?
? "END OF RUN (OR ERROR IN SHAPE OR L/G RATIO > 3.0)"
set alternate to
set alternate off
setcolor(m->C_Help)
scroll(0,0,4,79)
@0,0 to 4,79 double
@1,2 say "Output written to file: " + cOutFile
@2,2 say "Current subdirectory..: " + diskname() + ":" + dirname()
@3,2 say "Press a key to continue..."
inkey(0)
endif
endif
setcolor(m->C_Normal)
readinsert(lReadInsert)
return (NIL)
*** End of Func: FileInput()

*****
Function GetFileBox(nTR, nLC, nBR, nRC, cDirSpec, lDispBox, cColor, nInitFile)
* Pop-up file selector, all params. are optional
* Parameter defaults:
*   nTR top row ==> to 0
*   nLC left col ==> to 0
*   nBR bot row ==> to maxrow
*   nRC right col ==> to nLC + 38
*   cDirSpec ==> "*.*)"
*   lDispBox ==> .T.
*   ColorVar ==> "W+/n,n/W"
*   nInitFile ==> 1
* Returns:
*   if Enter key ==> File name
*   if Esc key ==> NIL
*   if error ==> NIL
**
local cOrgClr := ""
local cFileName := NIL
local cTmpScn := ""
local i // Scratch
local aDirctry := {} // Array of dir info
local acFileNames := {} // Array of file names
local nFileChoice := 0

* If any param. not passed, below assigns defaults as needed.
default nTR to 0
default nLC to 0
default nBR to maxrow()
default nRC to nLC + 38
default cDirSpec to "*.*)"
default lDispBox to .T.
default cColor to (m->C_Help)
default nInitFile to 1

cTmpScn := savescreen(nTR,nLC,nBR+1,nRC+1) // +1 for shadow lines

if (!SubDir(cDirSpec))

```

Code File: EGPCFile.Prg

```

    Err_MsgBox(15,"E","No .SIF files found in current subdir.")
    return (NIL)
endif

cOrgClr := setcolor(cColor)
scroll(nTR,nLC,nBR,nRC)
if (LDispBox)
    MenuBox(nTR,nLC,nBR,nRC)
endif
@nTR,nLC+2 say " Choose Input File... "

aDrctry := directory(cDirSpec)
* Sort array according to file name.
asort(aDrctry,,, (|FrstName,NextName| FrstName[F_NAME] < NextName[F_NAME]))

* Fill an array with file info to display.
acFileNames := {}
for i = 1 to len(aDrctry)
    aadd(acFileNames, ;
        padl(aDrctry[i,F_NAME],13) + ;
        padl(numtrim(aDrctry[i,F_SIZE]),8) + ;
        padl(dtoc(aDrctry[i,F_DATE]), 9) + ;
        padl(substr(aDrctry[i,F_TIME],1,5),6) )
next i

* Display files and get choice.
nFileChoice := achoice(nTR+1,nLC+1,nBR-1,nRC-1, acFileNames,,,nInitFile)
if (nFileChoice != 0)
    * Is 0 if Esc key exit
    cFileName := aDrctry[nFileChoice,F_NAME]
    nInitFile := nFileChoice
endif
setcolor(cOrgClr)
restscreen(nTR,nLC,nBR+1,nRC+1,cTmpScn)
return (cFileName)
*** End of Func: GetFileBox()

*****~*****
Function GetFileOutFile(cInFile, lProceed)
* Returns file output file name entered by user.
* Updates flag lProceed.
* lProceed parameter should be passed in by reference.
static cOutFile := "" // Screen output file
local nChoice := 1
local GetList := {}
local cCurrPath := ""
local lDone := .F.
local lOrgReadins := readinsert(.T.) // Insert mode for read = on.

* Make default outfile name.
cCurrPath := diskname() + ":" + dirname()
* Make output file name, cInFile plus .OUT.
if at(".",cInFile) == 0
    cOutFile := cInFile + ".OUT"
else
    cOutFile := substr(cInFile,1,at(".",cInFile)) + "OUT"
endif
* Add trailing \ to path, if needed.
cOutFile := cCurrPath + iif(right(cCurrPath,1)=="\", "", "\") + cOutFile

do while ! lDone
    cls
    MenuBox(2,1,8,67)
    cOutFile := padr(cOutFile,64)
    @03,03 say "Enter output file name:"
    @04,03 get cOutFile pict "a!"

```

Code File: EGPCFile.Prg

```

@05,03 say "Current path: " + cCurrPath
keyboard chr(K_END)
read
readinsert(lOrgReadIns)
cOutFile := alltrim(cOutFile)

if lastkey() == K_ESC
  lProceed := .F.
else
  lProceed := .T.
  if file(cOutFile)
    * Decide whether to overwrite output file.
    nChoice := AlertBox(8,"YES, Overwrite It", "Enter New Name"), ;
    "Warning: Above output file exists!", "", ;
    "      Overwrite it?      ")
    if nChoice == 1
      * Overwrite output file.
      set alternate to (cOutFile)
    elseif nChoice == 2
      * Enter new name.
      loop
    else
      * Esc key. Don't open output file.
      lProceed := .F.
    endif
  else
    * cOutFile does not exist, try to open it.
    * First test for valid subdir and valid file name.
    if ! Subdir(ExtractPath(cOutFile)) .or. ;
      ! filevalid(token(cOutFile,":\"))
      * Invalid path or file name.
      Err_MsgBox(10,"E","Error: Invalid path or file name.", ;
        "File.: " + cOutfile)
      loop
    else
      * Valid path, open file for output.
      set alternate to (cOutFile)
    endif
  endif
endif
lDone := .T.
enddo
return (cOutFile)
*** End of Func: GetFileOutFile()

*****
Function GetScnOutFile(lOutFile, lWriteHeader)
* Returns screen output file name entered by user.
* Updates flags lOutFile, lWriteHeader.
* Pass flags in by reference.
static cOutFile := "" // Screen output file
local nChoice := 1
local GetList := {}
local cCurrPath := ""
local lDone := .F.
local lOrgReadIns := readinsert(.T.) // Insert mode for read = on.

* Default to no outfile flag for Esc key pressed on read.
lOutFile := .F.
lWriteHeader := .F.
cCurrPath := diskname() + ":" + dirname()

if empty(cOutFile)
  * Default out file is cCurrPath\Screen.Out.
  * Add trailing \ to path, if needed.
  cOutFile := cCurrPath + iif(right(cCurrPath,1)=="\", "", "\") + "Screen.Out"

```

Code File: EGPCFile.Prg

```

else
  * An out file name has been used.
  * Default out file is cCurrPath\cOutFile. Note that cCurrPath may have
  * changed since cOutFile name created.
  cOutFile := substr(cOutFile,at("\",cOutFile)+1) // Get just the filename.
  cOutFile := cCurrPath + iff(right(cCurrPath,1)=="\",","", "\") + cOutFile
endif

do while ! lDone
  lOutFile := .F. // Reset for loops
  lWriteHeader := .F.
  cls
  MenuBox(2,1,8,67)
  cOutFile := padr(cOutFile,64)
  @03,02 say " A Screen Output File Is Optional Esc = None "
  @05,03 say "Enter output file name:"
  @06,03 get cOutFile pict "a!"
  @07,03 say "Current path: " + cCurrPath
  keyboard chr(K_END)
  read
  readinsert(lOrgReadIns)

  cOutFile := alltrim(cOutFile)
  if lastkey() != K_ESC
    * First test for valid subdir and valid file name.
    if ! Subdir(ExtractPath(cOutFile)) .or. ;
      ! filevalid(token(cOutFile,":\"))
      * Invalid path or file name.
      Err_MsgBox(10,"E","Error: Invalid path or file name.", ;
        "File.: " + cOutfile)
      loop
    elseif file(cOutFile)
      * File exists.
      * Decide whether to overwrite output file.
      nChoice := AlertBox(8, ;
        ("NO, Append to it","YES, Overwrite It", "Enter New Name"), ;
        "Warning: Above screen output file exists!", "", ;
        "Overwrite it?")
      if nChoice == 1
        * Append to output file.
        lOutFile := .T.
        set alternate to (cOutFile) additive
      elseif nChoice == 2
        * Overwrite output file.
        lOutFile := .T.
        lWriteHeader := .T.
        delete file (cOutFile)
        set alternate to (cOutFile) additive
      elseif nChoice == 3
        * Try again.
        loop
      else
        * Esc key. Don't open output file.
        lDone := .T.
        loop
      endif
    else
      * If here, we have valid path and file does not exist.
      * Open file for screen output.
      lWriteHeader := .T.
      lOutfile := .T.
      set alternate to (cOutFile)
    endif
  endif
  lDone := .T.
enddo

```

Code File: EGPCFile.Prg

```

return (cOutFile)
*** End of Func: GetScnOutFile()

*****
Function SIF_FileInput(nTR, nLC, nInitFile, cVerDate)
* Gets data from SIF format input file.
* Writes output to cInFile.Out.
* Input:  nTR is row for file selection box,
*         nLC is left col.
*         nInitFile is initial file to highlight.
*         cVerDate is program version date.
* Returns: NIL
local lReadInsert := readlnaert(.T.)
local cInFile      := ""           // Input file name
local cOutFile     := ""
local cDataLine    := ""
local nSemiMajor   := 0
local nShape       := 0           // Ellipse semi-major/semi-minor axis
local nAngle       := 0
local nGSize       := 0           // Grid size, for Rec. grids, short side
local nGType       := 0           // Grid type, 1=Sq., 2=Tri., 3=Rec.
local nOrientn     := 0           // Specific angle or "random",
local cTrgtID      := ""         // if nOrientn > 0 use "random" angles
local nInputLine   := 0           // Current file input line
local nRecRatio    := 0           // Rec. grid long side/short side ratio
local nLines       := 0           // Lines in input file
local nProbNoHit   := 0
local nProbSum     := 0.0         // Used for "random" angle case
local nCrntAngle   := 0           // Used for "random" angle case
local nLrgstAngle  := 0           // Used for "random" angle case
local cFileText    := ""
local lProceed     := .T.
local GetList      := {}         // Stops compiler warnings

cInFile := GetFileBox(nTR,nLC,,,"*.SIF",,,@nInitFile)
cFileText := memoread(cInFile)
nLines := mlcount(cFileText)
if lastkey() == K_ESC .or. empty(cInFile)
* Just return, if Esc key pressed.
elseif nLines < 3
* Error, invalid file.
Err_MsgBox(10,"E","Error: Less than 3 lines in file.", ;
"File.: " + cInFile, ;
"Need (1) Title line, (2) Data line, (3) EOF line.")
else
* Input file looks OK, create output file name.
cOutFile := GetFilOutFile(cInFile, @lProceed)

if lProceed
* Do the work!
cls
set alternate to (cOutFile)
set alternate on
?? "Output from ORNL/GJ ELIPGRID-PC Program Version: " + cVerDate
? "File Name.: " + cOutFile
? "Created on: " + dtoc(date())
? "Input file: " + cInFile + " using SIF format."
? "Title line: " + memoline(cFileText,,1)
?
? "Target  Grid Type      Semi-major Axis  Gridspace  Shape  Angle  Prob(0)"
? "          in Relative Units  in Orig Units"

* Get data lines
nInputLine := 2           // Skip title line
do while nInputLine <= nLines
cDataLine := alltrim(memoline(cFileText,,nInputLine))

```

Code File: EGPCFile.Prg

```

if left(cDataLine,1) == ""
  * Comment line.
  nInputLine++
  loop
endif
* Parse the data values.
nSemiMajor := val(substr(cDataLine, 1,at(" ",cDataLine)))
cDataLine  := ltrim(substr(cDataLine,at(" ",cDataLine)))
nShape     := val(substr(cDataLine, 1,at(" ",cDataLine)))
cDataLine  := ltrim(substr(cDataLine,at(" ",cDataLine)))
nAngle     := val(substr(cDataLine, 1,at(" ",cDataLine)))
cDataLine  := ltrim(substr(cDataLine,at(" ",cDataLine)))
nGSize     := val(substr(cDataLine, 1,at(" ",cDataLine)))
cDataLine  := ltrim(substr(cDataLine,at(" ",cDataLine)))
nGTyp      := val(substr(cDataLine, 1,at(" ",cDataLine)))
cDataLine  := ltrim(substr(cDataLine,at(" ",cDataLine)))
nOrientn   := val(substr(cDataLine, 1,at(" ",cDataLine)))
cDataLine  := ltrim(substr(cDataLine,at(" ",cDataLine)))
cTrgtID    := cDataLine

if nShape > 1.0 .or. nShape < 0.05 .or. nSemiMajor/nGSize > 3.0
  * EOF or error in shape or L/G ratio > 3.
  exit // Exit do while loop
endif

if nGTyp == 3
  * If rect. grid, get long/short ratio from next line.
  nInputLine++
  cDataLine := ltrim(memoline(cFileText,nInputLine))
  nRecRatio := val(cDataLine)
  if nRecRatio == 1.0
    * Trap for a rect. grid with a long/short side ratio of 1.0.
    * Use a sq. grid since problems can develop using rect. grid in
    * certain cases. Problem found in tech. review by John Wilson.
    nGTyp := 1
  endif
endif

*-----| Calculate probability of no hit, P(0) |-----*
if nOrientn <= 0.0
  * Calcualte for a single angle.
  nProbNoHit := ElipGrid(nSemiMajor,nShape,nAngle,nGSize,nGTyp, ;
    nRecRatio)
else
  * Calculate for average of multiple angles,
  * i.e., "random" choice in Singer's 1972 ELIPGRID.
  if nGTyp == 1
    nLrgstAngle := 45
  elseif nGTyp == 2
    * For triangular grid (hexagon).
    nLrgstAngle := 30
  elseif nGTyp == 3
    * For rectangular grid.
    nLrgstAngle := 90
  endif
  * Sum up multiple angles results.
  nProbSum := 0.0
  for nCrntAngle = 0 to nLrgstAngle
    nProbNoHit := ElipGrid(nSemiMajor,nShape,nCrntAngle,nGSize, ;
      nGTyp, nRecRatio)
    nProbSum := nProbSum + nProbNoHit
  next nCrntAngle

  * Calculate average.
  nProbNoHit := nProbSum/(nLrgstAngle+1)
endif

```

Code File: EGPCFile.Prg

```

*-----*

* Print a line of data.
? padr(cTgtID,8)
if nGTyp == 1
?? "Square      " + space(8)
elseif nGTyp == 3
?? "Rectangular, " + trans(nRecRatio,"99.9") + "/1  "
elseif nGTyp == 2
?? "Triangular  " + space(8)
endif

* Print data fields.
?? trans(nSemiMajor/nGSize,"9999.9999")+ space(6) + ;
   trans(nGSize,"9999.99")          + space(7) + ;
   trans(nShape,"9.99")              + space(3) + ;
   iff(nOrientn > 0,"Random",trans(nAngle,"99.9"+"  ")) + ;
   space(1) + ;
   trans(nProbNoHit,"999.9999")
* Increment line index.
ninputLine++
enddo
?
? "END OF RUN (OR ERROR IN SHAPE OR L/G RATIO > 3)"
set alternate to
set alternate off
setcolor(m->C_Help)
scroll(0,0,4,79)
@0,0 to 4,79 double
@1,2 say "Output written to file: " + cOutFile
@2,2 say "Current subdirectory... " + diskname() + ":" + dirname()
@3,2 say "Press a key to continue..."
inkey(0)
endif
endif
readinsert(lReadInsert)
return (NIL)
*** End of Func: SIF_FileInput()

*****
Function Subdir(cTestSubdir)
* Returns .T. if cTestSubdir exists, .F. otherwise.
* The directory() command will return an empty array
* if cTestSubdir does not exist.
local lRtnVal := .F.
local aDirctry := {}

cTestSubdir := alltrim(cTestSubdir)
* directory() returns an empty array, {}, if invalid cTestSubdir.
aDirctry := directory(cTestSubdir, "D") // D to include all subdirs
if len(aDirctry) > 0
lRtnVal := .T.
endif
return (lRtnVal)
*** End of Func: Subdir()

*** End of File: EGPCFile.Prg

```

Code File: EGPCFort.Prg

```
//=====
// File:          EGPCFort.Prg
// For:           ELIPGRID-PC, EGPC.Exe.
// Purpose:       Provides ELIPGRID FORTRAN code in Clipper form.
//               Note correction to ELIPGRID in RECT subroutine.
// Author:        Jim Davidson
// Prog Started:  10/03/93
// Last Mod:      08/26/94
// Note:          Functions are arranged in alphabetical order.
// Modifications since validation with Singer's 100 cases:
// 03/30/94 Modified various if/then/else conditions pointed out in J. Wilson's
//           technical review.
// 04/04/94 Modified ElipGrid algor. to deal with triangular grid discontinuity.
//           Added 3 levels of correction to the ElipGrid algorithm:
//           0 = Use original 1972 Singer code.
//           1 = Use corrected RECT routine.
//           2 = Use corrected RECT routine and corrected triangular grid
//               routine (4th order regression). "2" is the default.
// 04/06/94 Added level 3 correction.
//           3 = Do not increment angles of 0.0 to 0.1, as in ELIPGRID.
// 04/07/94 Added code to force P(0) to 0.0 when hit probabilities > 1.0 make
//           P(0) negative. See relevant ELIPGRID code just above line 435.
//           Tested on file Test100.IN, output checked with previous validation.
//           Tested on file Test100.SIF, output checked with previous validation.
// 04/15/94 Added comments re: Shape restrictions in ELIPGRID at line 75 that
//           are now in the calling code. New L/G ratio restriction noted.
// 04/17/94 Added trap for ANGLE = 90, tan(90) sometimes causes runtime error.
//           Added to level 3 correction: if ANGLE = 90.0, then ANGLE := 89.999.
// 08/09/94 Changed name from HotSFort.Prg to EGPCFort.Prg.
// 08/11/94 Correction level flag, m->nElpGrdCor, now forced to 3 to provide
//           full correction all the time. 04/07/94 mod. makes level 0 dubious.
//=====
```

```
*****
Function ElipGrid(A, Shape, Angle, GdSpac, Met, Q)
* This function is taken from Singer's 1972 ELIPGRID program.
* It retains the original algorithm, but is modified to remove
* all goto type statements. Many line numbers have been left in the
* comments as references back to the original code.
*
* Shape      : Shapes > 1.0 are trapped before reaching this function.
* assumptions: Shapes < 0.05 are trapped before reaching this function.
*             Trapping all Shapes < 0.05 is somewhat more restrictive than
*             ELIPGRID, but should have little practical consequences.
*             I would like to see a verification of the math before accepting
*             arbitrarily small Shapes, JRD, 04/15/94.
* L/G
* assumptions: Code assumes all L/G ratios > 3.0 are trapped. Very large L/G
*             ratios, e.g. 6 or 7, have caused problems. No known practical
*             need requires them. Singer's largest L/G ratio in his 100
*             cases was 2.83. Gilbert's largest L/G ratio in nomographs is 1.0.
*
* Note that the MET parameter described below, is not used in this function.
* Random angle case is taken care of by calling code.
*
* Below is original code documentation.
*
*           PROGRAM ELIPGRID
*
*           PROGRAM TO DETERMINE THE PROBABILITY OF LOCATING AN ELLIPTIC OR
*           CIRCULAR TARGET WITH A SQUARE, HEXAGONAL OR RECTANGULAR GRID
*
*           DESCRIPTION OF PARAMETERS
*
*           TARGET= ANY IDENTIFICATION OF TARGET (READ IN "A" FORMAT)
```

Code File: EGPCFort.Prg

```

*      A= LENGTH OF SEMIMAJOR AXIS OF TARGET
*      SHAPE= SHAPE OF TARGET - SEMIMINOR AXIS DIVIDED BY THE SEMIMAJOR
*      ANGLE= POSITIVE ANGLE BETWEEN LONG AXIS OF TARGET AND GRID
*              DIRECTION - FOR A SQUARE GRID ANGLE CAN BE ANY ABGLE FROM
*              0 TO 45 DEGRESS, FOR A HEXAGONAL GRID ANGLE CAN BE ANY
*              ANGLE FROM 0 TO 30 DEGREES INCLUSIVE, FOR A RECTANGULAR
*              GRID ANGLE CAN BE ANY ANGLE FROM 0 TO 90 DEGREES
*              INCLUSIVE AND IS MEASURED FROM THE X AXIS OF THE GRID
*      GDSPAC= DISTANCE BETWEEN POINTS ON THE GRID (IN THE SAME UNITS AS
*              "A") - FOR A RECTANGULAR GRID GDSPAC IS THE DISTANCE
*              BETWEEN POINTS ALONG THE Y AXIS OF THE GRID
*      NET= GRID TYPE - SQUARE GRID=1, HEXAGONAL GRID=2, RECTANGULAR
*              GRID=3
*      MET= SPECIFIC OR RANDOM ORIENTATION - IF MET>0 - RANDOM
*      Q= SHAPE OF RECTANGULAR GRID - LONG(X) AXIS DIVIDED BY THE
*          SHORT(Y) AXIS
*

```

* These locals are integers in ELIPGRID.

```

local I      := 0
local IBLANK := 0
local IWARN  := 0
local IZONK  := 0
local M      := 0

```

* These locals are reals in ELIPGRID.

```

local ALPHA := 0
local ANP   := 0
local AQUAR := 0
local AREA1 := 0
local AREA2 := 0
local AREA3 := 0
local AREA4 := 0
local AREA5 := 0
local AREA6 := 0
local AREA7 := 0
local AREA8 := 0
local AREA9 := 0
local AREA10:= 0
local ASQ   := 0
local AVPR0 := 0
local AVPR1 := 0
local AVPR2 := 0
local B     := 0
local BALLS := 0
local BSQU  := 0
local C     := 0
local CAROL := 0
local CIM   := 0
local CNM   := 0
local D     := 0
local DJ0   := 0
local DJ1   := 0
local DMO   := 0
local DM1   := 0
local EOU   := 0
local FIN   := 0
local FORN  := 0
local GAME  := 0
local GRO   := 0
local HAI   := 0
local HALFC := 0
local HALFD := 0
local HALFJ0:= 0
local HALFJ1:= 0
local HALFM0:= 0

```

```

// C/2
// D/2
// J0/2
// etc.

```

Code File: EGPCFort.Prg

```

local HALFMI := 0
local HORN    := 0
local PET     := 0
local PI      := 0           // Constant pi, 3.141592 in ELIPGRID
local POT     := 0
local PROB0   := 0           // Prob. of no hits
local PROB1   := 0           // Prob. of 1 hit
local PROB2   := 0
local RO      := 0
local RDW     := 0
local REVA    := 0           // For rect. grid, transformed A
local REVANG  := 0           // For rect. grid, transformed angle
local REVK    := 0           // For rect. grid, transformed SHAPE
local SER     := 0
local SLING   := 0
local SINGLE  := 0
local SUM0    := 0
local SUM1    := 0
local SUM2    := 0
local T       := 0
local TIN     := 0
local TIZ     := 0
local WINE    := 0
local XHAPE   := 0
local XI      := 0
local XM      := 0
local Y1      := 0
local YI      := 0
local YAM     := 0
local YM      := 0
local ZAP     := 0

/*** Note 08/11/94, JRD ***/
*--- Force full correction level all the time. ---*
m->nElpGrdCor := 3

* Below are assignments made in ELIPGRID.
PI      := 3.141592           // Follows original value.
TIZ     := 0.50000
RDW     := SQRT(3.0)*0.5

IZONK   := IBLANK
A        := A/GDSPAC
SLING    := A
XHAPE    := SHAPE
SINGLE    := ANGLE
SUM1     := 0.0
SUM2     := 0.0
SUM0     := 0.0

*
*   AREAS 1 TO 10 ARE RELATIVE AREAS OF OVERLAP IN THE TRANSFORMED NET
*   35
AREA1    := 0.0
AREA2    := 0.0
AREA3    := 0.0
AREA4    := 0.0
AREA5    := 0.0
AREA6    := 0.0
AREA7    := 0.0
AREA8    := 0.0
AREA9    := 0.0
AREA10   := 0.0

*
*   PROB0 IS THE PROBABILITY OF MISSING THE TARGET

```

Code File: EGPCFort.Prg

```

*   PROB1 IS THE PROBABILITY OF LOCATING THE TARGET ONCE
*   PROB2 IS THE PROBABILITY OF LOCATING THE TARGET TWO OR MORE TIMES
*
PROB0 := 0.0
PROB1 := 0.0
PROB2 := 0.0
*
*   DETERMINES THE GRID TYPE
*
* GO TO (65,40,45),NET
if NET == 2
  ***New Code, 04/04/94, JRD***
  * Handle problem with tri. grid discontinuity near L/G = 0.577.
  * A is the L/G ratio.
  * If ElipGrid correction level is >= 2, consider 4th order linear regression.
  if m->nElpGrdCor >= 2
    if (A > 0.50 .and. A < 0.60) .and. (Shape >= 0.85 .and. Shape < 1.0)
      * Use 4th order linear regression results, not ELIPGRID algorithm.
      return(Prob0_Regr(A,Shape))
    endif
  endif
  ***End New Code, 04/04/94***
  *
  *   HEXAGONAL NET
  *   40
  FIN   := RDW
  * IROT := 30 not needed in this function.
  ZAP   := 6.0
  BALLS := 0.57735
  * GO TO 75
elseif NET == 3
  *
  *   RECTANGULAR NET
  *
  *   45 IF (IROT) 50,50,60
  *
  *   READ SHAPE OF RECTANGULAR GRID
  *
  *   50 READ (IREAD,55) Q
  *   55 FORMAT (F10.5)
  *   60 CALL RECT(SLING,XHAPE,ANGLE,Q,REVK,REVA,REVANG)
  *   Argument SLING is never used by subroutine RECT().
  RECT(XHAPE,ANGLE,Q,QREVK,QREVA,QREVANG)
  SHAPE := REVK
  A      := REVA*SLING
  ANGLE  := REVANG
  * IROT := 90 not needed in this function.
  * GO TO 70
  * 70
  FIN   := 1.000
  ZAP   := 4.0
  BALLS := 0.707107
elseif NET == 1
  *
  *
  *   SQUARE NET
  *
  *   65 IROT=45
  * IROT := 45 not needed in this function.
  * 70
  FIN   := 1.000
  ZAP   := 4.0
  BALLS := 0.707107
endif
* SHAPE restriction below handled by trapping ALL SHAPES < 0.05 in calling

```

GRID1035

Code File: EGPCFort.Prg

```

* code. This is more restrictive than SHAPES < 0.05 and A (L/G) > 2 test below.
* ELIPGRID-PC also traps all L/G ratios > 3.0.
* 75 IF (SHAPE-0.05) 80,95,95
* 80 IF (A-2.0) 95,95,85
* 85 WRITE (IPRIN,90) TARGET
* 90 FORMAT (1H,6HTARGET,A4,45H IS TOO NEEDLE-LIKE AND LONG FOR THIS P
* 1ROGRAM)
* GO TO 20
***

* 95 IF (SHAPE-1.0) 140,115,100 Note: 100 terminates.
* SHAPES > 1.0 or < 0.05 are not allowed to come in to ElipGrid().

if SHAPE == 1.0
*
* CIRCLE
ASQ := A**2 // 115
* IF (A-TIZ) 120,120,125
if A - TIZ <= 0.0
PROB2 := 0.0
PROB1 := PI*ASQ/FIN
PROB0 := 1.0-PROB1
***New Code, 04/07/94, JRD***
* Handle cases where a hit prob. is > 1.0, thus making P(0) negative.
* See relevant ELIPGRID code just above code line 435.
PROB0 := iif(PROB0 < 0.0, 0.0, PROB0)
***End New Code, 04/07/94***
// Top, left STOP in Fig. 7 flowchart
// (Singer and Wickman 1969)
// In JRD notes as STOP 2
* 1st return
return(PROB0)
else
* IF (A-BALLS) 130,135,135 // 125
if A-BALLS < 0
CIM := ACOS(TIZ/A) // 130
PROB2 := ZAP*(ASQ*CIM-TIZ*SQRT(ASQ-0.25))/FIN
PROB1 := PI*ASQ/FIN-2.0*PROB2
PROB0 := 1.0-PROB1-PROB2
***New Code, 04/07/94, JRD***
* Handle cases where a hit prob. is > 1.0, thus making P(0) negative.
* See relevant ELIPGRID code just above code line 435.
PROB0 := iif(PROB0 < 0.0, 0.0, PROB0)
***End New Code, 04/07/94***
* 2nd return
return(PROB0)
else
*
* IF THE RADIUS OF THE CIRCLE IS GREATER THAN 0.7071 THE PROBABILITY
* OF MISSING IS ZERO AND PROB1 AND PROB2 ARE SET EQUAL TO 9. AS
* FLAGS
*
PROB1 := 9.0 // 135
PROB2 := 9.0
PROB0 := 0.0
* 3rd return
return(PROB0)
endif
endif
elseif SHAPE < 1.0
*
* ELLIPSE
*
B := A*SHAPE // 140
*
* B IS THE RADIUS OF THE CIRCLE IN THE TRANSFORMED NET
*

```

Code File: EGPCFort.Prg

```

* IF (A-TIZ) 145,145,150
if A-TIZ <= 0.0
  PROB1 := PI*A*B/FIN          // 145
  PROB2 := 0.0
  PROB0 := 1.0-PROB1
  ***New Code, 04/07/94, JRD***
  * Handle cases where a hit prob. is > 1.0, thus making P(0) negative.
  * See relevant ELIPGRID code just above code line 435.
  PROB0 := if(PROB0 < 0.0, 0.0, PROB0)
  ***End New Code, 04/07/94***

  * 4th return
  return(PROB0)                // Top, right STOP in Fig. 7 flowchart
endif                          // (Singer and Wickman 1969)
                              // In JRD notes as STOP 1

***New Code, 04/06/94, JRD***
* Handle 0.0 angle being incremented to 0.1.
if m->nElpGrdCor < 3
  * IF(ANGLE-0.1) 155,155,160      // 150
  if ANGLE-0.1 <= 0.0
    *
    *   ALPHA IS THE ANGLE IN RADIANs
    *
    ANGLE := ANGLE+0.1          // 155
  endif
else
  * Level 3 correction below does not increment 0.0 to 0.1,
  * but does make sure the angle is positive.
  ANGLE := abs(ANGLE)
endif
***End New Code, 04/06/94***

ALPHA := ANGLE/57.295779        // 160
CNM := 1.0-SHAPE**2
*
*   C,D,DJ1,DJ0,DM1,DM0 ARE DISTANCES BETWEEN CIRCLES IN THE
*   TRANSFORMED NET .
*
C := SQRT(1.0-CNM*COS(ALPHA)**2)

* GO TO (170,165,170),NET      // 164
if NET == 2
  * 165 below.
  Y1 := 3.0+SHAPE**2-2.0*CNM*SIN(ALPHA)**2-CN**4.0*FIN*SIN(ALPHA)*COS(ALPHA)
  D := SQRT(Y1)*0.5000
elseif NET == 1 .or. NET == 3
  D := SQRT(1.0-CNM*SIN(ALPHA)**2)// 170
endif

BSQU := B**2                  // 175
FORN := C*C
HORN := D*D
WINE := FIN*SHAPE
HALFC := C*0.50
HALFD := D*0.50

* IF (B-HALFC) 185,185,180      // 179
if B-HALFC > 0.0
  EQU := ACOS(HALFC/B)         // 180
  AREA1 := 2.0*(BSQU*EQU-HALFC*SQRT(BSQU-HALFC**2))
elseif B-HALFC <= 0.0
  AREA1 := 0.0                 // 185
endif

* IF (B-HALFD) 200,200,195      // 190
if B-HALFD > 0.0

```

Code File: EGPCFort.Prg

```

      HAI := ACOS(HALFD/B)           // 195
      AREA2 := 2.0*(BSQU*HAI-HALFD*SQRT(BSQU-HALFD**2))
    else
      AREA2 := 0.0                  // 200
    endif

* IF (A-BALLS) 210,210,215          // 205
if A-BALLS <= 0.0
  PROB2 := (AREA1+AREA2)/WINE      // 210
  PROB1 := PI*BSQU/WINE-2.0*PROB2
  PROB0 := 1.0-PROB1-PROB2
  * 5th return
  ***New Code, 04/07/94, JRD***
  * Handle cases where a hit prob. is > 1.0, thus making P(0) negative.
  * See relevant ELIPGRID code just above code line 435.
  PROB0 := iif(PROB0 < 0.0, 0.0, PROB0)
  ***End New Code, 04/07/94***
  return(PROB0)                    // Center, left STOP in Fig. 7 flowchart
                                   // (Singer and Wickman 1969)
                                   // In JRD notes as STOP 3
else
  * IF (ANGLE) 220,220,225          // 215
  if ANGLE <= 0.0
    C := C+0.05                    // 220
  endif
  CAROL := C*D                     // 225

  T := ASIN(WINE/CAROL)

  * IF (ANGLE) 235,230,235
  if ANGLE == 0.0
    DJ1 := SQRT(FORN+HORN)          // 230
    DJ0 := 5.0
    *
    * R0 IS THE RADIUS NECESSARY FOR THE TARGET TO BE HIT WITH CERTAINTY
    *
    R0 := DJ1/2.0
  else
    *I:=1.0+(D*COS(T)/C)             // 235
    I := int(1.0+(D*COS(T)/C))      // 235 modified with int()
    * IF (I-1) 240,240,245
    if I-1 <= 0.0
      DJ1 := SQRT((FORN+HORN)-2.0*CAROL*COS(T))
      DJ0 := 5.0
      R0 := DJ1/(2.0*SIN(T))
    else
      XI := I                        // 245
      YI := I-1
      DJ1 := SQRT(XI**2*FORN+HORN-2.0*XI*CAROL*COS(T))
      DJ0 := SQRT(YI**2*FORN+HORN-2.0*YI*CAROL*COS(T))
      R0 := DJ1*DJO/(2.0*D*SIN(T))
    endif
  endif
endif
endif

* 250 IF (B-R0) 260,255,255
if B-R0 >= 0.0
  PROB1 := 9.0                     // 255
  PROB2 := 9.0
  PROB0 := 0.0
  ***New Code, 04/07/94, JRD***
  * Handle cases where a hit prob. is > 1.0, thus making P(0) negative.
  * See relevant ELIPGRID code just above code line 435.
  PROB0 := iif(PROB0 < 0.0, 0.0, PROB0)
  ***End New Code, 04/07/94***

```

Code File: EGPCFort.Prg

```

      * 6th return
      return(PROB0)           // Bot., right STOP in Fig. 7 flowchart
endif                          // (Singer and Wickman 1969)
                              // In JRD notes as STOP 4

HALFJ1 := DJ1*0.50           // 260
HALFJ0 := DJ0*0.50

* Below is CIRCLE A on flowchart Fig. 7. (Singer and Wickman 1969).
* IF (B-HALFJ1) 270,270,265
if B-HALFJ1 > 0.0
  GRO := ACOS(HALFJ1/B)      // 265
  AREA3 := 2.0*(BSQU*GRO-HALFJ1*SQRT(BSQU-HALFJ1**2))
else
  AREA3 := 0.0              // 270
endif

* 275 IF (B-HALFJ0) 285,285,280
if B-HALFJ0 > 0.0
  PET := ACOS(HALFJ0/B)     // 280
  AREA4 := 2.0*(BSQU*PET-HALFJ0*SQRT(BSQU-HALFJ0**2))
else
  AREA4 := 0.0
endif

* 290 M=1.0+(2.0*D*COS(T)/C)
M := int(1.0+(2.0*D*COS(T)/C)) // 290 modified with int().
YM := M-1
XM := M

* IF (M-1) 295,295,300
if M-1 <= 0.0
  DM1 := SQRT(FORN+HORN*4.0-4.0*CAROL*COS(T)) // 295
  DM0 := 5.0
else
  DM1 := SQRT(XM**2*FORN+4.0*HORN-4.0*CAROL*COS(T)) // 300
  DM0 := SQRT(YM**2*FORN+4.0*HORN-4.0*CAROL*COS(T))
endif

HALFM1 := DM1*0.50           // 305
HALFM0 := DM0*0.50

/***
  IF (HALFM1-DJ1) 310,325,310
310 IF (HALFM1-DJ0) 315,325,315
315 IF (HALFM0-DJ1) 320,325,320
320 IF (HALFM0-DJ0) 330,325,330
replaced with below:
***
if HALFM1 == DJ1 .or. ;
  HALFM1 == DJ0 .or. ;
  HALFM0 == DJ1 .or. ;
  HALFM0 == DJ0
  AREA5 := 0.0              // 325
  AREA6 := 0.0
else
  * 330 IF (B-HALFM1) 340,340,335
  if B-HALFM1 > 0.0
    YAM := ACOS(HALFM1/B)   // 335
    AREA5 := 2.0*(BSQU*YAM-HALFM1*SQRT(BSQU-HALFM1**2))
  else
    AREA5 := 0.0           // 340
  endif

  * 345 IF (B-HALFM0) 355,355,350
  if B-HALFM0 > 0.0
    GAM := ACOS(HALFM0/B)   // 350

```

Code File: EGPCFort.Prg

```

      AREA6 := 2.0*(BSQU*GAME-HALFMO*SQRT(BSQU-HALFMO**2))
    else
      AREA6 := 0.0                      // 355
    endif
  endif
endif

* 360 IF (B-DJ1) 370,370,365
if B-DJ1 > 0.0
  SER := ACOS(DJ1/B)                  // 365
  AREA7 := 2.0*(BSQU*SER-DJ1*SQRT(BSQU-DJ1**2))
else
  AREA7 := 0.0                        // 370
endif

* 375 IF (B-DJ0) 385,385,380
if B-DJ0 > 0.0
  AQUAR := ACOS(DJ0/B)                // 380
  AREA8 := 2.0*(BSQU*AQUAR-DJ0*SQRT(BSQU-DJ0**2))
else
  AREA8 := 0.0                        // 385
endif

* 390 IF (B-C) 400,400,395
if B-C > 0.0
  POT := ACOS(C/B)                    // 395
  AREA9 := 2.0*(BSQU*POT-C*SQRT(BSQU-FORN))
else
  AREA9 := 0.0                        // 400
endif

* 405 IF (B-D) 415,415,410
if B-D > 0.0
  TIN := ACOS(D/B)                    // 410
  AREA10 := 2.0*(BSQU*TIN-D*SQRT(BSQU-HORN))
else
  AREA10 := 0.0                       // 415
endif

PROB2 := (AREA1+AREA2+AREA3+AREA4+AREA5+AREA6+AREA7+AREA8+AREA9+AREA10)/WINE
PROB1 := PI*BSQU/WINE-2.0*PROB2-(AREA7+AREA8+AREA9+AREA10)/WINE
PROB0 := 1.0-PROB1-PROB2
***New Code, 04/07/94, JRD***
* Handle cases where a hit prob. is > 1.0, thus making P(0) negative.
* See relevant ELIPGRID code just above code line 435.
PROB0 := iff(PROB0 < 0.0, 0.0, PROB0)
***End New Code, 04/07/94***

* 7th return
return(PROB0)                        // Bot., right STOP in Fig. 7 flowchart
endif                                // 2nd page (Singer and Wickman 1969)
                                     // In JRD notes as STOP 5

* Error return, should never get here.
* 8th return
return(-1)
*** End of Func: ELIPGRID()

*****
Function Prob0_Regr(nLtoG, nShape)
* Determine prob. of missing, P(0), by using a 4th order polynomial regression.
* The regression coefficients were determined using SigmaPlot 5.01 and data
* sets with the values near the discontinuity removed.
* Input:  nLtoG  Semi-major axis to grid size ratio.
*         nShape Semi-minor axis to semi-major axis ratio.
* Output: Prob0  Prob. of missing target.
* Errors:  if nLtoG or nShape out of applicable range, returns 9.

```

Code File: EGPCFort.Prg

```

local nRtnVal := 0
local nB0, nB1, nB2, nB3, nB4      // Regression coefficients
* Check L/G ratio for correct range.
if nLtoG > 0.50 .and. nLtoG < 0.60
  do case
    case nShape >= 0.85 .and. nShape < 0.86
      * Will use regr. coeffs. calculated with nShape == 0.85.
      * Any shape < 0.85 did not appear to need regression.
      nB0 := 0.8736
      nB1 := -5.8080
      nB2 := 39.1737
      nB3 := -95.8914
      nB4 := 71.2386
    case nShape >= 0.86 .and. nShape < 0.88
      * Will use regr. coeffs. calculated with nShape == 0.87.
      nB0 := -1.5907
      nB1 := 13.4531
      nB2 := -16.2801
      nB3 := -26.7985
      nB4 := 39.9151
    case nShape >= 0.88 .and. nShape < 0.92
      * Will use regr. coeffs. calculated with nShape == 0.90.
      nB0 := -8.7963
      nB1 := 71.1939
      nB2 := -187.7169
      nB3 := 195.8430
      nB4 := -66.7013
    case nShape >= 0.92 .and. nShape < 0.94
      * Will use regr. coeffs. calculated with nShape == 0.93.
      nB0 := -19.3100
      nB1 := 156.3713
      nB2 := -443.8610
      nB3 := 533.8017
      nB4 := -231.6841
    case nShape >= 0.94 .and. nShape < 0.96
      * Will use regr. coeffs. calculated with nShape == 0.95.
      nB0 := -27.4195
      nB1 := 222.4814
      nB2 := -644.0422
      nB3 := 800.0227
      nB4 := -362.8194
    case nShape >= 0.96 .and. nShape < 0.98
      * Will use regr. coeffs. calculated with nShape == 0.97.
      nB0 := -35.7606
      nB1 := 290.7372
      nB2 := -851.5507
      nB3 := 1077.1734
      nB4 := -499.9610
    case nShape >= 0.98 .and. nShape < 1.00
      * Will use regr. coeffs. calculated with nShape == 0.99.
      nB0 := -44.0006
      nB1 := 358.3580
      nB2 := -1057.7388
      nB3 := 1353.4032
      nB4 := -637.0739
    otherwise
      * Error: nShape out of range.
      nRtnVal := 9
  endcase
else
  * Error: nLtoG ratio out of range.
  nRtnVal := 9
endif
if nRtnVal != 9
  * Calculate 4th order polynomial.
  nRtnVal := nB0 + nB1 * nLtoG + nB2 * nLtoG^2 + nB3 * nLtoG^3 + nB4 * nLtoG^4

```

Code File: EGPCFort.Prg

```

* Round any neg. values up to 0.0.
nRtnVal := iif(nRtnVal<0.0, 0.0, nRtnVal)
endif
return (nRtnVal)
*** End of Func: ProbO_Regr()

*****
Function RECT(SHAPE,ANGLE,q,REVK,REVA,REVANG)
* This function is taken from Singer's 1972 ELIPGRID program.
* It retains the original algorithm, but is modified to remove
* all goto type statements. Many line numbers have been left in the
* comments as references back to the original code.
* Note the comment below regarding apparent error in the 1972 ELIPGRID code.
*
*
* THIS SUBROUTINE REDUCES THE RECTANGULAR POINT NET TO A SQUARE
* POINT NET WITH AN AFFINE TRANSFORMATION
*
local AQ
local SQK
local TIS
local ALPHA
local COAL
local SIAL
local T

AQ := Q*Q
SQK := SHAPE**2
TIS := AQ*SQK

***New Code, 04/06/94, JRD***
* Handle 0.0 angle being incremented to 0.1.
if m->nElpGrdCor < 3
  * IF (ANGLE-0.1) 5,5,10
  if ANGLE-0.1 <= 0.0
    ANGLE := ANGLE+0.1
  endif
else
  * Level 3 correction below does not increment 0.0 to 0.1,
  * but does make sure the angle is positive.
  ANGLE := abs(ANGLE)
  * Added ANGLE = 90° trap to level 3 correction, 04/17/94.
  ANGLE := iif(ANGLE==90.0, 89.999, ANGLE)
endif
***End New Code, 04/06/94***

ALPHA := ANGLE/57.295779
COAL := COS(ALPHA)**2
SIAL := SIN(ALPHA)**2
T := SQRT(((1.0-TIS)*COAL-(AQ-SQK)*SIAL)**2+4.0*AQ*(1.0-SQK)**2*SIAL*COAL)
REVK := ((1.0+TIS)*COAL+(AQ+SQK)*SIAL-T)/(2.0*Q*SHAPE)
* Below appears to be an error in the original code.
* See (Singer and Wickman 1969, p. 16) for the original math formula.
* REVANG=(ATAN(2.0*Q*(1.0-SQK)*TAN(ALPHA)/((AQ-SQK)*TAN(ALPHA)**2*TIRECT 175
* S-1.0))/2.0)*57.295779
if m->nElpGrdCor == 0
  * Use original formula.
  REVANG := (ATAN(2.0*Q*(1.0-SQK)*TAN(ALPHA)/((AQ-SQK)*TAN(ALPHA)**2 * ;
  TIS-1.0))/2.0)*57.295779
else
  * Next line is corrected formula.
  REVANG := (ATAN(2.0*Q*(1.0-SQK)*TAN(ALPHA)/(1.0-TIS-(AQ-SQK)* ;
  TAN(ALPHA)**2))/2.0)*57.295779
endif
REVA := SQRT(SHAPE/(Q*REVK))
REVANG := ABS(REVANG)

```

Code File: EGPCFort.Prg

```
* The following optional code matches (Singer and Wickman 1969, 16)
* and can be used in place of line RECT 205 above. However, no differences
* in output values were seen when testing Singer's 30 rect. grid examples after
* this code was substituted for line RECT 205 (in ELIPGRD2.FOR).
* if (tan(2.0 * REVANG) >= 0.0) then
*   REVANG = abs(REVANG)
* else
*   REVANG = 90.0 - abs(REVANG)
* endif
RETURN (HIL)
*** End of Func: RECT()

*** End of File: EGPCFort.Prg
```

Code File: EGPCGrph.Prg

```

//=====
// File:          EGPCGrph.Prg
// For:           ELIPGRID-PC, EGPC.Exe.
// Purpose:       Provides code for "Write Cost-based Graph Data" option.
// Author:        Jim Davidson
// Prog Started:  04/22/94
// Last Mod:      09/06/94
// Note:          Functions are arranged in alphabetical order.
// Modifications:
// 08/09/94 Changed name from WotsGrph.Prg to EGPCGrph.Prg.
// 09/02/94 Grid cell area for triangular grid now calculated from rhombus.
//          This adjusts the EPA formula for number of samples for tri. grid.
// 09/06/94 Added "Number Samples" to graphics output data file.
//=====

// Include files.
// Clipper supplied include files.
#include "Directry.Ch"           // File info definitions
#include "Inkey.Ch"              // Key definitions
#include "Set.Ch"                // set() definitions
#include "Setcurs.Ch"            // setcursor() related
#include "Box.Ch"                // Box drawing constants

// ORNL developed include files.
#include "EGPCMax.Ch"            // Hot spot maximums for screen I/O

// Simple graph demo related. Not for external use.
#define GPH_DEMO                 // Optional simple graph demo
#undef GPH_DEMO                  // Add or remove * as 1st char of line
                                // to define or undefine graph demo.

// Defines
#define nPI 3.1415926            // Note that ELIPGRID uses 3.141592
#define nSC_CONVERGED 0          // Status code: Converged
#define nSC_PAST_MAXI 1          // Status code: Past max iterations
#define nSC_ABORTED 2           // Status code: Esc key abort
#define nSC_DATA_RNGE 3          // Status code: Data out of range
#define nSC_UNKNOWN 99          // Status code: Unknown error

// User-defined command
#define DEFAULT <TheParam> TO <DefaultVal> => ;
    IF (<TheParam> == NIL); <TheParam>:=<DefaultVal>; ENDIF

*****
Function GetCostProb(nDesirdCost, nSemiMajor, nShape, nAngle, cGridType, nRecRatio,;
    nSampleArea, nSampleCost, nCostClst, nGSizeFnd, nPrbHtFnd, nNumSampReq)
* Finds the cost-based probability for a given set of parameters.
* Returns results through 4 variables passed in by reference:
*   nCostClst      = Closest cost found to nDesirdCost
*   nGSizeFnd      = Corresponding grid size found
*   nPrbHtFnd      = Corresponding prob. found
*   nNumSampReq     = Corresponding number of samples required
*   Above vars must be passed in by reference, use @.
* Returns status codes:
*   Converged, prob OK = nSC_CONVERGED
*   Failed to converge = nSC_PAST_MAXI
*   Aborted with Esc   = nSC_ABORTED
*   Data out of range  = nSC_DATA_RNGE
*   Unknown error      = nSC_UNKNOWN

#define nMAXC_ITERS 25           // Max cost search iterations

* Below are grid size restrictions to stay in reasonable search.
local nSmaGrid := 0.33334 * nSemiMajor
* Below same as L/G of 0.10.
local nLrgeGrid := 10.0 * nSemiMajor

```

Code File: EGPCGrph.Prg

```

* Initialize flags.
local lConverg      := .F.          // .T. if search converges
local lAborted      := .F.          // .T. if Esc key abort
local lPastMaxIt    := .F.          // .T. if past max iterations
local lOKProb       := .T.          // .F. if problem can't be solved
local nRtnStatus    := 0            // Status code to return

local nGTyp         := 1            // Grid type requested, ELIPGRID form
local nGLong        := 0            // Long side for rec grids
local nSICounter     := 0            // Search iterations counter
local nRACounter     := 0            // Random angle counter
local nProbNoHit     := 1            // Probability of zero hits, P(0)
local nCrntAngle     := 0            // Current angle, used for "random" angle
local nLrgstAngle    := 0            // Largest angle, used for "random" angle
local nProbSum       := 0            // Summing var., used for "random" angle
local nNoHitSmlGrd   := 1            // P(0) for small grid
local nNoHitLrgGrd   := 1            // P(0) for large grid
local nIntrGrid      := 0            // Interpolated grid size
local nNoHitIntGrd    := 1            // P(0) for interpolated grid
local nDiffCost      := 0            // Current diff. from desired cost
local nCurrCost      := 0            // Current cost
local nGridCellArea  := 0            // Area of one grid cell
local nNumSamples    := 0            // Number of samples required

*-----| Find grid size for given cost |-----*
@18,38 say "Current point iteration:   of " + NumTrim(nMAXC_ITS) + " max."

* Get nGTyp, i.e., ELIPGRID grid type code.
nGTyp := iif(cGridType=="S",1,iif(cGridType=="R",3,2))

* Get random largest random angle.
if nAngle == 99.0
  @19,38 say "Random angle iterations: "
  nLrgstAngle := iif(cGridType=="S",45,iif(cGridType=="R",90,30))
endif

* Start searching.
do while ! lConverg .and. ! lAborted .and. ! lPastMaxIt .and. lOKProb
  nSICounter++           // Increment search counter
  @18,63 say str(nSICounter,3)+ " of " + ;
    ltrim(str(nMAXC_ITS)) + " max."

  * GET PROB. AND COST FOR LARGE GRID.
  if nAngle != 99.0
    * Non-random single angle case.
    nNoHitLrgGrd := ElipGrid(nSemiMajor,nShape,nAngle,nLrgGrd, ;
      nGTyp, nRecRatio)
  else
    * Sum up multiple angle results, random case.
    nProbSum := 0.0
    for nCrntAngle = 0 to nLrgstAngle
      nProbNoHit := ElipGrid(nSemiMajor, nShape, nCrntAngle, ;
        nLrgGrd, nGTyp, nRecRatio)
      nProbSum := nProbSum + nProbNoHit
      nRACounter++
      @19,63 say ltrim(str(nRACounter)) + " "
      * Esc key abort, only used with random angles.
      if inkey() == K_ESC
        lAborted := .T.
        exit           // exit for/next loop
      endif
    next nCrntAngle
  if lAborted
    loop
  endif
endif

```

Code File: EGPCGrph.Prg

```

    * Calculate average.
    nNoHitLrgGrd := nProbSum/(nLrgstAngle+1)
endif

* Check if we met cost criteria with large grid.
* First do area calculations.
if cGridType == "R"
    * Rect. grid.
    nGLong := nLrgeGrid * nRecRatio
else
    * Sq. or Tri. grid.
    nGLong := nLrgeGrid
endif

* Required number of samples is approximate.
* Based on (EPA. 1989. "Methods for Eval. the Attainment of Cleanup
* Standards Volume 1: Soils and Solid Media", p. 9-7.
* Calculate grid cell area.
if cGridType == "T"
    * Triangular grid.
    * Grid cell area is now, 09/02/94, area of the rhombus formed
    * from 2 of the equilateral triangles.
    * A = height * base = sin(60°) * base * base = 0.87 * base^2.
    nGridCellArea := 0.866025404 * nLrgeGrid * nLrgeGrid
else
    * Sq. or rec. grid.
    nGridCellArea := nLrgeGrid * nGLong
endif

* This formula is approx. See (EPA 1989, 9-7).
* Ceiling function rounds number of samples up.
nNumSamples := ceiling(nSampleArea/nGridCellArea)

nCurrCost := nNumSamples * nSampleCost

* Can we quit yet?
nDiffCost := abs(nDesirdCost - nCurrCost)
if nDiffCost < nSampleCost
    * Met error criteria with current large grid.
    * Exit grid search.
    nCostClst := nCurrCost
    nGSizeFnd := nLrgeGrid
    nPrbHitFnd := 1.0 - nNoHitLrgGrd
    nNumSampReq := nNumSamples
    lConverg := .T.
    loop
endif

* Will grid size need to be larger than 3 * semi-major axis?
* If so, an L/G ratio > 3.0 would be required.
* If first search with largest grid can't get down to the
* desired cost, no need to search farther.
if nSICounter == 1 .and. (nCurrCost > nDesirdCost)
    * Quit searching.
    lOKProb := .F.
    loop
endif

* GET PROB. FOR SMALL GRID.
if nAngle != 99.0
    * Non-random single angle case.
    nNoHitSmlGrd := ElipGrid(nSemiMajor,nShape,nAngle,nSmlGrid, ;
    nGTyp, nRecRatio)
else
    * Sum up multiple angle results, random case.
    nProbSum := 0.0
    for nCrntAngle = 0 to nLrgstAngle
        nProbNoHit := ElipGrid(nSemiMajor, nShape, nCrntAngle, ;

```

Code File: EGPCGrph.Prg

```

        nSmlGrid, nGTyp, nRecRatio)
    nProbSum := rProbSum + nProbNoHit
    nRACounter++
    @19,63 say ltrim(str(nRACounter)) + "    "
    * Esc key abort, only used with random angles.
    if inkey() == K_ESC
        lAborted := .T.
        exit          // exit for/next loop
    endif
next nCrntAngle
if lAborted
    loop
endif

* Calculate average.
nNoHitSmlGrd := nProbSum/(nLrgstAngle+1)
endif

* Check if we met cost criteria with small grid.
* First do area calculations.
if cGridType == "R"
    * Rect. grid.
    nGLong := nSmlGrid * nRecRatio
else
    * Sq. or Tri. grid.
    nGLong := nSmlGrid
endif

* Required number of samples is approximate.
* Based on (EPA. 1889. "Methods for Eval. the Attainment of Cleanup
* Standards Volume 1: Soils and Solid Media", p. 9-7.
* Calculate grid cell area.
if cGridType == "T"
    * Triangular grid.
    * Grid cell area is now, 09/02/94, area of the rhombus formed
    * from 2 of the equilateral triangles.
    * A = height * base = sin(60°) * base * base = 0.87 * base^2.
    nGridCellArea := 0.866025404 * nSmlGrid * nSmlGrid
else
    * Sq. or rec. grid.
    nGridCellArea := nSmlGrid * nGLong
endif

* This formula is approx. See (EPA 1989, 9-7).
* Ceiling function rounds number of samples up.
nNumSamples := ceiling(nSampleArea/nGridCellArea)

nCurrCost := nNumSamples * nSampleCost

* Can we quit with current small grid?
nDiffCost := abs(nDesirdCost - nCurrCost)
if nDiffCost < nSampleCost
    * Met error criteria with current small grid.
    * Exit grid search.
    nCostClst := nCurrCost
    nGSizeFnd := nSmlGrid
    nPrbHitFnd := 1.0 - nNoHitSmlGrd
    nNumSampReq := nNumSamples
    lConverg := .T.
    loop
endif

* Will grid size need to be smaller than 1/3 * semi-major axis?
* If so, an L/G ratio < 1/3 would be required.
* If first search with smallest grid can't get up to the
* desired cost, no need to search farther.
if nSICounter == 1 .and. (nCurrCost < nDesirdCost)
    * Quit searching.

```

Code File: EGPCGrph.Prg

```

    LOKProb := .F.
  loop
endif

* Bisection method, (Gerald and Wheatley 1989, 7)
nIntrGrid := (nLrgGrid + nSmlGrid)/2

* GET PROB. AND COST FOR INTERPOLATED GRID.
if nAngle != 99.0
  * Non-random single angle case.
  nNoHitIntGrd := ElipGrid(nSemiMajor, nShape, nAngle, nIntrGrid, ;
    nGTyp, nRecRatio)
else
  * Sum up multiple angle results, random case.
  nProbSum := 0.0
  for nCrntAngle = 0 to nLrgstAngle
    nProbNoHit := ElipGrid(nSemiMajor, nShape, nCrntAngle, ;
      nIntrGrid, nGTyp, nRecRatio)
    nProbSum := nProbSum + nProbNoHit
    nRACounter++
    @19,63 say ltrim(str(nRACounter)) + " "
    * Esc key abort, only used with random angles.
    if inkey() == K_ESC
      lAborted := .T.
      exit // exit for/next loop
    endif
  next nCrntAngle
  if lAborted
    loop
  endif

  * Calculate average.
  nNoHitIntGrd := nProbSum/(nLrgstAngle+1)
endif

* Check if we met cost criteria with interpolated grid.
* First do area calculations.
if cGridType == "R"
  * Rect. grid.
  nGLong := nIntrGrid * nRecRatio
else
  * Sq. or Tri. grid.
  nGLong := nIntrGrid
endif

* Required number of samples is approximate.
* Based on (EPA. 1989. "Methods for Eval. the Attainment of Cleanup
* Standards Volume 1: Soils and Solid Media", p. 9-7.
* Calculate grid cell area.
if cGridType == "T"
  * Triangular grid.
  * Grid cell area is now, 09/02/94, area of the rhombus formed
  * from 2 of the equilateral triangles.
  * A = height * base = sin(60°) * base * base = 0.87 * base^2.
  nGridCellArea := 0.866025404 * nIntrGrid * nIntrGrid
else
  * Sq. or rec. grid.
  nGridCellArea := nIntrGrid * nGLong
endif

* This formula is approx. See (EPA 1989, 9-7).
* Ceiling function rounds number of samples up.
nNumSamples := ceiling(nSampleArea/nGridCellArea)

nCurrCost := nNumSamples * nSampleCost

* Can we quit with current interpolated grid?
nDiffCost := abs(nDesirdCost - nCurrCost)

```

Code File: EGPCGrph.Prg

```

if nDiffCost < nSampleCost
  * Met error criteria with current interpolated grid.
  * Exit grid search.
  nCostClst := nCurrCost
  nGSizeFnd := nIntrGrid
  nPrbHtFnd := 1.0 - nNoHitIntGrd
  nNumSampReq := nNumSamples
  lConverg := .T.
  loop
endif

* Update large or small search grid sizes.
* This is a minor difference from linear interpolation and bisection
* methods. They look for sign changes of f(x). In root search
* case, f(x) values will be changing about 0.0.
* We look at whether our current f(x) for the interpolated grid
* is larger than the desired value.
if nCurrCost > nDesirdCost
  nSmalGrid := nIntrGrid
else
  nLrgeGrid := nIntrGrid
endif

* Have we reached max iterations?
if nSICounter == nMAXC_ITERS
  * Failed to converge.
  lPastMaxIt := .T.
  loop
endif
enddo

* Determine return code.
if lConverg
  * Converged, probability OK.
  nRtnStatus := nSC_CONVERGED
elseif lPastMaxIt
  * Failed to converge.
  nRtnStatus := nSC_PAST_MAXI
elseif lAborted
  * Aborted with Esc.
  nRtnStatus := nSC_ABORTED
elseif ! lOKProb
  * Data out of range.
  nRtnStatus := nSC_DATA_RNGE
else
  * Unknown error.
  nRtnStatus := nSC_UNKNOWN
endif

return (nRtnStatus)
*** End of Func: GetCostProb()

*****
Function GetGphDataFile()
* Returns screen output file name entered by user.
* Errors: Checks and warns for valid subdir and file names.
* Returns NIL on Esc key abort.
static cOutFile := "" // Screen output file
local nChoice := 1
local GetList := {}
local cCurrPath := ""
local lDone := .F.
local cRtnVal := cOutFile // Return value, file or NIL
local lOrgReadins := readinsert(.T.) // Insert mode for read = on.

* Default to no outfile flag for Esc key pressed on read.

```

Code File: EGPCGrph.Prg

```

cCurrPath    := diskname() + ":" + dirname()

if empty(cOutFile)
    * Default out file is cCurrPath\Screen.Out.
    * Add trailing \ to path, if needed.
    cOutFile := cCurrPath + iif(right(cCurrPath,1)=="\", "", "\") + "Graph.Dat"
else
    * An out file name has been used.
    * Default out file is cCurrPath\cOutFile. Note that cCurrPath may have
    * changed since cOutFile name created.
    cOutFile := substr(cOutFile, rat("\", cOutFile)+1) // Get just the filename.
    cOutFile := cCurrPath + iif(right(cCurrPath,1)=="\", "", "\") + cOutFile
endif

do while ! lDone
    cls
    MenuBox(2,1,8,67)
    cOutFile := padr(cOutFile,64)
    @04,03 say "Enter graphics data file name:"
    @05,03 get cOutFile pict "a!"
    @06,03 say "Current path: " + cCurrPath
    @07,03 say "Esc = Abort"
    keyboard chr(K_END)
    read
    readinsert(10rgReadIns)

    cOutFile := alltrim(cOutFile)
    if lastkey() != K_ESC
        * First test for valid subdir and valid file name.
        if ! Subdir(ExtractPath(cOutFile)) .or. ;
            ! filevalid(token(cOutFile,":\"))
            * Invalid path or file name.
            Err_MsgBox(10,"E","Error: Invalid path or file name.", ;
                "File.: " + cOutfile)
            loop
        elseif file(cOutFile)
            * File exists.
            * Decide whether to overwrite output file.
            nChoice := AlertBox(8, {"YES, Overwrite It", "Enter New Name"}, ;
                "Warning: Above graph data file exists!", "", ;
                "Overwrite it?","YES, Overwrite It", "Enter New Name"))
            if nChoice == 1
                * Overwrite output file.
                delete file (cOutFile)
                set alternate to (cOutFile) additive
            elseif nChoice == 2
                * Try again.
                loop
            else
                * Esc key. Don't open output file.
                lDone := .T.
                loop
            endif
        else
            * If here, we have valid path and file does not exist.
            * Open file for screen output.
            set alternate to (cOutFile)
        endif
    endif
    lDone := .T.
enddo
if lastkey() == K_ESC
    * Abort.
    cRtnVal := NIL
else
    * Return file path\name.

```

Code File: EGPCGrph.Prg

```

    cRtnVal := cOutFile
endif
return (cRtnVal)
*** End of Func: GetGphDataFile()

*****
Function OutGphData(lWriteHeader,cVerDate,cOutFile,cGridType,nRecRatio, ;
    nShape, nSemiMajor, nAngle, nSampleArea, nSampleCost, nMinCost, ;
    nLoopCost1, nLoopCostMax, nCostStepSize, ;
    nCostClst, nPrbHtFnd, nGSizeFnd, nNumSampReq, nStatus, cDelim)
* Write out header or cost-based graph data.
local cErr      := ""          // Error msg
local nFrstDL   := 18         // First data line

set console off
set alternate on

if lWriteHeader
    * Write the file info header.
    ?? "# Data starts on line: " + NumTrim(nFrstDL)
    ?  "# Output from ORNL/GJ ELIPGRID-PC Program Version: " + cVerDate
    ?  "# File name.: " + cOutFile
    ?  "# Created on: " + dtoc(date())
    ?  "# Current length and area units...: " + ;
        iif(m->cBasicUnit=="F","Feet, ft","Meters, m")
    ?  "# Grid type chosen.....: " + ;
        iif(cGridType=="S","Square",iif(cGridType=="T","Triangle","Rectangle"))+;
        iif(cGridType=="R"," with long/short side ratio "+NumTrim(nRecRatio), "")
    ?  "# Shape of the elliptical hot spot: " + NumTrim(nShape)
    ?  "# Length of semi-major axis.....: " + NumTrim(nSemiMajor) + ;
        iif(m->cBasicUnit=="F"," ft"," m")
    ?  "# Angle of orientation to grid....: " + ;
        iif(nAngle==99.0,"Random", NumTrim(nAngle)+"°")
    ?  "# Total area to sample.....: " + NumTrim(nSampleArea) + ;
        iif(m->cBasicUnit=="F"," ft"," m")
    ?  "# Individual sample cost.....$: " + ltrim(str(nSampleCost,15,2))
    ?  "# Minimum cost for graph data....$: " + ltrim(str(nLoopCost1,15,2)) + ;
        " approx."
    ?  "# Maximum cost allowed this run..$: " + ltrim(str(nLoopCostMax,15,2)) + ;
        " approx."
    ?  "# Step size for cost values.....$: " + ltrim(str(nCostStepSize,15,2))+ ;
        " approx."
    ?  "# Total Cost   Prob. of"
    ?  "# of Sampling  Hitting, P(≥1)  Grid Size Found  Number Samples"
    ?  "#-----"
    lWriteHeader := .F.
else
    * Write a line of data.
    if nStatus == nSC_CONVERGED
        ? trans(nCostClst, "9999999.99") + cDelim + space(5)
        ?? trans(nPrbHtFnd, "9.99999") + cDelim + space(7)
        ?? trans(nGSizeFnd, "999999.999") + cDelim + space(5)
        ?? trans(nNumSampReq, "999,999")
    else
        * Determine error code.
        if nStatus == nSC_PAST_MAXI
            cErr := "Failed to converge."
        elseif nStatus == nSC_ABORTED
            cErr := "Aborted with Esc."
        elseif nStatus == nSC_DATA_RNGE
            * Data out of range.
            cErr := "Data out of range."
        else
            cErr := "Unknown error."
        endif
        ? "## Error in calculating cost: " + cErr
    end
end

```

Code File: EGPCGrph.Prg

```

endif
endif

set alternate off
set console on
return (NIL)
*** End of Func: OutGphData()

*****
Function WriteGphData(cGridType, cVerDate)
* Write cost-based graph data.
#define nMAX_COST_POINTS 250          // Max number of cost-based data values

static nSemiMajor    := 2.82          // Length of semi-major axis
local ntSemiMajor    := nSemiMajor    // Temp value
static nAngle        := 0.0           // Orientation angle of hot spot to grid
local ntAngle        := nAngle        // Temp value
static nRecRatio     := 2.0           // Rectangular grid long/short ratio.
local ntRecRatio     := nRecRatio      // Temp value
static nShape        := 1.0           // Shape, minor/major axis
static nSampleArea   := 7000          // Total area to sample
local ntSampleArea   := nSampleArea   // Temp value
static nSampleCost    := 700          // Cost for one sample
local ntSampleCost   := nSampleCost   // Temp value
static nMinCost       := 200000        // Minimum cost for graph
local ntMinCost       := nMinCost      // Temp value
static nCostIncr      := 10           // Cost increment for graph
local ntCostIncr      := nCostIncr     // Temp value
static cDelim         := " "          // ASCII data delimiter
local ctDelim         := cDelim        // Temp value

static cOutFile       := "Graph.Dat"  // Graph data output file

local nGSizeFnd       := 0            // Grid size found
local nPrbHtFnd       := 0            // Prob. of hit found
local nCurrCost        := 0            // Current cost
local nCostClst        := 0            // Best estimate of cost
local nNumSampReq      := 0            // Number of samples required

* Misc vars.
local nCostStepSize    := 0            // Step size for cost loop
local nLoopCost1       := 0            // Loop cost 1
local nLoopCostMax     := 0            // Max loop cost
local nGraphPoint      := 0            // Graph point counter
local lWriteHeader     := .T.          // Write file header flag
local nStatus          := 0            // Status code for graph data file
local lDone            := .F.          // Loop flag
local lConverge        := .F.          // Convergence flag
local lAborted         := .F.          // Esc key abort
local lPastMaxIt       := .F.          // Exceeded nMAXC_ITERS flag
local lOKProb          := .T.          // Solvable problem specs flag
local nCol             := 0            // Scratch column()
local nKeyPress        := 0            // User key press
local getlist          := {}           // Stops compiler warnings

* For simple graph demo.
#ifdef GPH_DEMO
    local anCostVals    := {}           // Array of cost values
    local anProbVals    := {}           // Array of prob values
#endif

private ntShape        := nShape        // private for F10 key function

if cGridType == NIL
    * Input error: no grid typ passed in
    return (NIL)

```

Code File: EGPCGrph.Prg

```

endif

* Uppcase function argument.
cGridType := upper(cGridType)

* Get screen output file.
cOutFile := GetGphDataFile()
if cOutFile == NIL
    * Esc key abort.
    return (NIL)
endif

* Display screen title.
DispTitle(cGridType,"W",cOutFile,.T.)

do while ! lDone
    * Make sure header is available at beg. of each loop.
    @ 6, 2 say "Shape of the elliptical hot spot..:" ;
    get m->ntShape pict cMAX_Shape ;
    valid ErrorUDF(ntShape <= 1.0 .and. ntShape >= 0.05, ;
        "Shape must be ≥ 0.05 and ≤ 1.0.",len(cMAX_Shape))
    @ 6,49 say "Shape = short axis/long axis."
    @ 7,49 say "F10 calculates axis from area." color(m->C_Help)
    @ 7, 2 say "Length of semi-major axis.....:" ;
    get ntSemiMajor pict cMAX_SemiMajor ;
    valid ErrorUDF(ntSemiMajor > 0.0, ;
        "Length must be > 0.0.",len(cMAX_SemiMajor))
    @row(),col() say iif(m->cBasicUnit=="F","ft","m")
    @ 9,49 say ' 99.0° for "random" angles.'

    if cGridType == "R"
        * Rectangular grid.
        @ 8, 2 say "Angle of orientation to grid.....:" ;
        get ntAngle pict cMAX_Angle ;
        valid ErrorUDF(ntAngle >= 0 .and. ntAngle <= 90.0 .or. ntAngle == 99, ;
            "Angle must be 0° to 90° or 99°=random.",len(cMAX_Angle))
        @ 8,col() say ""
        @ 8,49 say "Angle can be 0° to 90°. Use"
        @ 9, 2 say "Long side/short side ratio.....:" ;
        get ntRecRatio pict cMAX_RecRatio ;
        valid ErrorUDF(ntRecRatio > 1.0, ;
            "Ratio must be > 1.0.",len(cMAX_RecRatio))

    elseif cGridType == "S"
        * Square grid.
        @ 8, 2 say "Angle of orientation to grid.....:" ;
        get ntAngle pict cMAX_Angle ;
        valid ErrorUDF(ntAngle >= 0 .and. ntAngle <= 45.0 .or. ntAngle == 99, ;
            "Angle must be 0° to 45° or 99°=random.",len(cMAX_Angle))
        @ 8,col() say ""
        @ 8,49 say "Angle can be 0° to 45°. Use"
    elseif cGridType == "T"
        * Triangular grid.
        @ 8, 2 say "Angle of orientation to grid.....:" ;
        get ntAngle pict cMAX_Angle ;
        valid ErrorUDF(ntAngle >= 0 .and. ntAngle <= 30.0 .or. ntAngle == 99, ;
            "Angle must be 0° to 30° or 99°=random.",len(cMAX_Angle))
        @ 8,col() say ""
        @ 8,49 say "Angle can be 0° to 30°. Use"
    endif

    @10, 2 say "Total area to sample.....:" ;
    get ntSampleArea pict cMAX_SampleArea ;
    valid ErrorUDF(ntSampleArea>0.0,"Area must be > 0.0",len(cMAX_SampleArea))
    @row(),col() say iif(m->cBasicUnit=="F","ft","m")
    @10,25 say "F10 = Acres" color(m->C_Help)

```

Code File: EGPCGrph.Prg

```

@11, 2 say "Individual sample cost.....$:" ;
  get ntSampleCost pict cMAX_SampleCost ;
  valid ErrorUDF(ntSampleCost>0.0,"Cost must be > 0.0",len(cMAX_SampleCost))
@12, 2 say "Minimum cost for graph data.....$:" ;
  get ntMinCost pict cDESIRD_COST ;
  valid ErrorUDF(ntMinCost >= ntSampleCost, ;
    "Total cost must be > Sample cost.",len(cDESIRD_COST))
@14, 2 say "Graph cost increment, 1-500 times sample cost:" get ntCostIncr ;
  pict "999" valid ErrorUDF(ntCostIncr > 0 .and. ntCostIncr <= 500, ;
    "Cost increment > 0 and < 400.",2)
@15, 2 say "Enter column delimiter, space or comma best..." get ctDelim

@22, 0 say "| "
@22,79 say "| "
@22,1 to 22,78

@23,2 say "Enter = Continue   Esc = Abort" + space(44)   // erase msg

set key K_F10 to F10_Key()
read
set key K_F10 to
nKeyPress := lastkey()

* Abort, Write Data, etc...
do case
  case (nKeyPress == K_ESC)
    * Esc key pressed
    if YN_MsgBox("Abort current data entry session? Y/N")
      lDone := .T.
    * Close output file.
    close alternate
    if filesize(cOutFile) < 22 .and. filesize(cOutFile) > 0
      * No data written to this file, erase.
      * Note: filesize() returns a -1 if no file found.
      * 22 is just the length of the first line of the header.
      * An aborted, new file should be just 1 byte, ASCII 26, Ctrl-Z.
      delete file (cOutFile)
    endif
  endif
endif

case (nKeyPress == K_ENTER .or. nKeyPress == K_CTRL_W)
  * Enter key or Ctrl-W pressed. [Ctrl-W currently not documented.]
  * Save changes to static vars.
  nShape      := m->ntShape
  nSemiMajor   := ntSemiMajor
  nAngle       := ntAngle
  nMinCost     := ntMinCost
  nCostIncr    := ntCostIncr
  cDelim       := ctDelim
  nRecRatio    := ntRecRatio

  * Cost related vars.
  nSampleArea := ntSampleArea
  nSampleCost := ntSampleCost

  *-----| Find probs. for various costs |-----*
  scroll(23,1,23,78)
  setcolor("W/N*")           // Force blinking with *
  @23,2 say "Calculating"
  setcolor(m->C_Normal)
  nCol := col()+1
  @22,nCol say "| "
  @23,nCol say "| "
  @24,nCol say "| "
  @23,52 say "Esc = Stop Calculations..."
  setcursor(SC_NONE)

```

Code File: EGPCGrph.Prg

```

* Get loop cost sizes.
nCostStepSize := nSampleCost * nCostIncr
nLoopCost1    := nMinCost // May later add check on nMinCost prob.
                        // before entering the loop.
// Allow maximum of nMAX_COST_POINTS tries to reach 100% prob. of hit.
nLoopCostMax  := nLoopCost1 + ((nMAX_COST_POINTS-1) * nCostStepSize)

* Write file header, only pass parameters needed.
OutGphData(@WriteHeader,cVerDate,cOutFile,cGridType,nRecRatio, ;
           nShape, nSemiMajor, nAngle, nSampleArea, nSampleCost, nMinCost, ;
           nLoopCost1, nLoopCostMax, nCostStepSize)

* Loop through probabilities for all costs.
@17,38 say "Calculating graph point:  of " + ;
           NumTrim(nMAX_COST_POINTS) + " max."

* For simple graphics demo.
#ifdef GPH_DEMO
    anCostVals := {}
    anProbVals := {}
#endif

for nCurrCost = nLoopCost1 to nLoopCostMax step nCostStepSize

    * Check prob. for current cost.
    @17,63 say str(++nGraphPoint,3)
    nStatus := GetCostProb(nCurrCost, nSemiMajor, nShape, nAngle, ;
                          cGridType, nRecRatio, nSampleArea, nSampleCost, ;
                          @nCostC1st, @nGSizeFnd, @nPrbHtFnd, @nNumSampReq)

    * Write out data for each cost (or failure msg).
    OutGphData(WriteHeader,cVerDate,cOutFile,cGridType,nRecRatio, ;
              nShape, nSemiMajor, nAngle, nSampleArea, nSampleCost, nMinCost, ;
              nLoopCost1, nLoopCostMax, nCostStepSize, ;
              nCostC1st, nPrbHtFnd, nGSizeFnd, nNumSampReq, nStatus, cDelim)

    * For simple graphics demo.
    #ifdef GPH_DEMO
        aadd(anCostVals,nCostC1st)
        aadd(anProbVals,nPrbHtFnd)
    #endif

    * Have we reached approx. 100% yet?
    if nPrbHtFnd > 0.999995
        * Yes, bail out of loop.
        exit
    endif

    * Esc key abort.
    if inkey() == K_ESC .or. nStatus == nSC_ABORTED
        exit
    endif
next nCurrCost
* Close output file.
close alternate

* Clean up calculating msg, etc.
scroll(23,1,23,78)
@22,nCol say "_"
@24,nCol say "_"
tone(440,1)
setcursor(SC_NORMAL)
close alternate
*-----*

* Done msg.

```

Code File: EGPCGrph.Prg

```

    if lastkey() == K_ESC .or. nStatus == nSC_ABORTED
        * Aborted msg.
        Err_MsgBox(7,"M","Calculations aborted.", ;
            "Results in graph data file: " + cOutFile)
    else
        * Finished msg.
        Err_MsgBox(7,"M","Calculations finished.", ;
            "Results in graph data file: " + cOutFile)
    endif

    * Simple graphics demo.
    #ifdef GPH_DEMO
        GraphDemo(anCostVals, anProbVals)
    #endif

    lDone := .T.
    loop
    endcase (LKey == )
enddo
return (NIL)
*** End of Func: WriteGphData()

*** End of standard code.

* Simple graphics demo code.
#ifdef GPH_DEMO
*****
Function GraphDemo(anCostVals, anProbVals)
* Simple graph demo. Not for external use.
* Flipper graphics lib. seems to use lots of memory and lock machine.
* This is probably due to old version of Flipper, version 5.0.
* Currently graph will use no more than last nMAX_G_POINTS available points.
#define nMAX_G_POINTS 50          // Max graph points
local nTR      := 5              // Top screen row
local nNumPoints := len(anCostVals)
local nCurPoint := 0
local nFrstPoint := 0

if ! (len(anProbVals) == nNumPoints)
    * Error.
    Err_MsgBox(7,"E","Error: Data arrays unequal in length.")
endif

cls
if MessageBox(7, "Yes, Display Graph ", "No, Return ", ;
    "Display simple graph demo?") == 1

    if nNumPoints > 0
        * Start Flipper.
        * 2 columns of data times 8 bytes per value.
        flip_init(nNumPoints*2*8)
        set_sayerrr(1)
        * 2 columns.
        initdata(2)
        set_type(1,5)
        if nNumPoints <= nMAX_G_POINTS
            nFrstPoint := 1
        else
            nFrstPoint := nNumPoints - nMAX_G_POINTS
        endif

        for nCurPoint = nFrstPoint to nNumPoints
            store_data(anCostVals[nCurPoint]/1000, 100*anProbVals[nCurPoint])
        next nCurPoint
        plot()
    endif
endif

```

Code File: EGPCGrph.Prg

```
        inkey(0)
        textmode()
    else
        * No points to graph.
        tone(440,.3)
        Err_MsgBox(10,"E","Error: No points to graph.")
    endif
endif
return(NIL)
*** End of Func: GraphDemo()
#endif
*** End of File: EGPCGrph.Prg
```

Code File: EGPCHelp.Prg

```

//=====
// File:          EGPCHelp.Prg
// For:           ELIPGRID-PC, EGPC.Exe.
// Purpose:       Provides help screen code.
// Author:        Jim Davidson
// Prog Started:  10/03/93
// Last Mod:      09/06/94
// Note:          Functions are arranged in alphabetical order.
// 08/09/94 Changed name from HotSHelp.Prg to EGPCHelp.Prg.
//=====

// Include files
#include "Inkey.Ch"           // key definitions
#include "Set.Ch"             // set() function defs.
#include "Setcurs.Ch"         // setcursor() related

*****
Function Help()
* F1 help driver function.
local cTmpScn := savescreen(0,0,24,79)
local cClr := setcolor(m->C_Help)
local nCursor := setcursor(SC_NORMAL)
local nKey := 0
local lDone := .F.
local nScreen := 1
local cMaxScns := "5"
local nMax := val(cMaxScns)

set key K_F1 to                // Stop recursion on F1

do while ! lDone
  cls
  * Display the nth help screen.
  HelpScnW(nScreen, cMaxScns)

  * Turn off any file output.
  set(_SET_EXTRA,.F.)

  @0,0 to 24,79 double
  if empty(set(_SET_EXTRAFILE))
    @01,45 say "F2=Write Help Screens to Help.Scn" color (m->C_Normal)
  else
    @01,45 say "Writing Help Screens to Help.Scn " color (m->C_Normal)
  endif
  @23,01 say ;
  " Press a key to continue... Esc=Return PgUp=Previous" + ;
  " Enter=Next No.=Pg " color (m->C_Normal)
  @23,29 say ""
  inkey(0)

  nKey := lastkey()

  if nKey == K_ESC
    * Return.
    lDone := .T.
  loop
  elseif nKey == K_PGUP .or. nKey == K_UP
    * Go to prev. help screen.
    nScreen--
    nScreen := iif(nScreen<1,nMax,nScreen)
  elseif nKey >= asc("1") .and. nKey <= asc(cMaxScns)
    * Go to page number.
    * Note that asc("1") == 49, asc("2") == 50, etc.
    nScreen := nKey - 48
  elseif nKey == K_F2 .and. empty(set(_SET_EXTRAFILE))
    * Write help screen to "Help.Scn" file. Only turn on if not on already.

```

Code File: EGPCHelp.Prg

```

    set(_SET_EXTRAFILE,"HELP.SCN",.F.)    // .F. overwrite
    ?? "ELIPGRID-PC Program Help Screen(s)"
    set(_SET_EXTRA,.T.)                  // .T. means set extra file on
else
    * Go to next help screen.
    nScreen++
    nScreen := iff(nScreen>nMax,1,nScreen)
endif
enddo

set(_SET_EXTRAFILE,"")                  // Close Help.scn file
setcolor(cClr)
setcursor(nCursor)
restscreen(0,0,24,79,cTmpScn)
set key K_F1 to Help()
return(NIL)
*** End of Func: Help()

*****
Function HelpScnW(nNum,cMax)
* Display help screen number nNum.
local cNum := ltrim(str(nNum))
* Code demonstration exp. date.
local cDemoDate := blidemte()
cDemoDate := substr(cDemoDate,5,2)+"-"+right(cDemoDate,2)+"-"+left(cDemoDate,4)

@01,02 say "ORNL/GJ ELIPGRID-PC Help Screen, " + cNum + " of " + cMax ;
color (m->C_Help)

if ! empty(set(_SET_EXTRAFILE))
    * File output.
    set console off
    ?
    ? "ORNL/GJ ELIPGRID-PC Help Screen, " + cNum + " of " + cMax
    set console on
endif

if nNum == 1
* Display screen 1.
text
    ELIPGRID-PC calculates the probability of detecting an assumed elliptical
    target.

Usage: EGPC [H | M | F | MF]
        EGPC H will give more information on above options.

Key Main Menu Options
"P Probability of Hitting Hot Spot" allows input from an ELIPGRID-style
input file, an SIF input file, or the screen.

"G Grid Size Required, Given Prob." determines a grid spacing that
results in a hit probability very close to a user-specified value.

"S Smallest Hot Spot Hit, Given Grid" determines the length of the
semi-major axis of the smallest hot spot that can be hit, given user-
specified conditions. The result is returned as an area in the
current units.

"C Cost-Based Grid" determines a grid size that meets the user-specified
conditions for a given cost.

"W Write Cost-Based Graph Data" produces ASCII *.DAT files for graphing.
endtext

elseif nNum == 2
text

```

Code File: EGPCHelp.Prg

Input file formats are:

(1) ELIPGRID format, a FORTRAN-style format of column positions.
 Line 1 is the title, format is A80. Data values and formats are:
 SemimajorAxis Shape Angle GridSize GridType Orientation TargetID
 F10.2 F10.2 F10.2 F10.2 I4 I4 A4
 If the grid type is 3, i.e., rectangular, the long/short side ratio
 must follow on the next data line with F10.2 format.
 EOF marker is a Shape > 1.0.

(2) Simplified input format (SIF) removes need for noting column positions.
 Line 1 is the title. Data values are in same order as ELIPGRID format:
 SemimajorAxis Shape Angle GridSize GridType Orientation TargetID
 One or more spaces must separate each data value.
 An asterisk, *, may comment out any line.
 EOF marker is a Shape > 1.0 or no more data lines.

endtext

elseif nNum == 3

text

Example input file formats follow:

ELIPGRID format:

Test41.in input file, 12/21/93. (Note: this is line 1, the title line.)

[Semi-major	Shape	Angle	GridSize	Type	Orient.	TargetID	This line
1000.0	0.38	22.0	800.0	1	0#261		is not part
1250.0	0.50	0.0	1074.57	2	0#104		of an ELIPGRID
1000.0	0.38	22.0	565.69	3	0#261		file.]
2.0							
1250.0	0.50	0.0	1000.0	1	1#104		
9.9	9.9	9.9	9.9	9	9	9	EOF

SIF format:

Test41.in input file, 12/21/93. (Note: this is line 1, the title line.)
 * Semi-major Shape Angle GridSize Type Orient. TargetID

1000.0	0.38	22.0	800.0	1	0	#261	
1250.0	0.50	0.0	1074.57	2	0	#104	(Note skewed columns OK.)
1000.0	0.38	22.0	565.69	3	0	#261	
2.0							
1250.0	0.50	0.0	1000.0	1	1	#104	

endtext

elseif nNum == 4

text

Grid types:

- 1 = Square
- 2 = Triangular (called hexagonal by Singer).
- 3 = Rectangular

Orientation of target to grid:

- 0 = Use angle given by data.
- > 0 = Use average of 0 to 45 degree angles for square grid.
 Use average of 0 to 30 degree angles for triangular grid.
 Use average of 0 to 90 degree angles for rectangular grid.
 These average values are called random by Singer.

Cost calculations:

These calculations depend on an approximate formula found in
 (EPA 1989, 9-7).

$$\text{Number samples required} = \frac{\text{Total Area}}{\text{Grid Cell Area}}. \text{ Since this formula is}$$

only approximate, the resulting cost is approximate.

endtext

Code File: EGPCHelp.Prg

```

elseif nNum == 5
text
  References:

    U.S.EPA. 1989. Methods for Evaluating the Attainment of Cleanup Standards
    Volume 1: Soils and Solid Media, EPA, Washington, DC.

    Gilbert, R.A. 1987. Statistical Methods for Environmental
    Pollution Monitoring. Van Nostrand Reinhold, New York.

    Singer, D.A. 1972. "ELIPGRID, A FORTRAN IV Program for Calculating the
    Probability of Success in Locating Elliptical Targets with Square,
    Rectangular, and Hexagonal Grids," Geocom Programs 4: 1-16.

  Further Information:
  Jim Davidson, ELIPGRID-PC program developer. ORNL/GJ, (303) 248-6259.
endtext
/*** Removed 08/11/94, JRD, program now forces full level 3 correction.
a14,03 say "Current ELIPGRID-PC Correction Level = " + ;
    NumTrim(m->nElpGrdCor) color (m->C_Error)
a15,03 say "Run EGPC Help for correction level information."
***
a17,03 say "Demonstration expiration date: "
aRow(),col() say cDemoDate color(m->C_Error)
endif
return (NIL)
*** End of Func: HelpScrW()

*****
Function ParamHelp(cVerDate)
* Parameter help screen.
set color to W/M
cls
?? repl("=",80)
?? "ORNL/GJ ELIPGRID-PC Program, Version: " + cVerDate
? "Usage: EGPC {H | M | F | MF}"
?
? "      EGPC      = Defaults to color screen and meters for basic unit."
? "      EGPC H[elp] = Help on command line parameters, this screen."
? "      EGPC M[ono] = Monochrome input screens."
? "      EGPC F[ee]t = Use feet for basic unit of length for screen"
? "                  input. File input can be any consistent unit."
? "      EGPC MF      = Monochrome screens and feet for basic screen unit."
?
?
? " Example: EGPC MF"
? "      Use monochrome screen, feet for basic screen unit."
? repl("=",80)
return (NIL)
*** End of Func: ParamHelp()

*** End of File: EGPCHelp.Prg

```

Code File: EGPCScrn.Prg

```

//=====
// File:          EGPCScrn.Prg
// For:           ELIPGRID-PC, EGPC.Exe.
// Purpose:       Provides screen input/output related code.
// Author:        Jim Davidson
// Prog Started:  04/18/94
// Last Mod:      09/02/94
// Note:          Functions are arranged in alphabetical order.
// Modifications:
// 08/09/94 Changed name from HotSScrn.Prg to EGPCScrn.Prg.
// 09/02/94 Grid cell area for triangular grid now calculated from rhombus.
//           This adjusts the EPA formula for number of samples for tri. grid.
//=====

// Include files.
// Clipper supplied include files.
#include "Directory.Ch"           // File info definitions
#include "Inkey.Ch"               // Key definitions
#include "Set.Ch"                 // set() definitions
#include "Setcurs.Ch"             // setcursor() related
#include "Box.Ch"                 // Box drawing constants

// ORNL developed include files.
#include "EGPCMax.Ch"             // ELIPGRID-PC maximums for screen I/O

// Defines
#define nPI 3.1415926             // Note that ELIPGRID uses 3.141592

// User-defined command
#definecommand DEFAULT <TheParam> TO <DefaultVal> => ;
    IF (<TheParam> == NIL); <TheParam>:=<DefaultVal>; ENDIF

*****
Function F10_Key(cProc,nLine,cVar)
* Calculates length of semi-major axis based on area.
* Calculates m² or ft² from acres.
static nHotSptArea := 25.0
static nTotalAcres := 10.0
local nSemiMajor := 0.0
local nNewArea := 0.0
local cTmpScn := savescr(0,0,24,79)
local GetList := {}
set key K_F10 to
if cVar == "NTSEMIAMAJOR" // Check if var is ntSemiMajor.
    * Calculates length of semi-major axis based on area.
    scroll(8,49,11,78)
    @ 8,49 to 11,78
    @ 9,50 say " Convert hot spot area to " + ;
        iif(m->cBasicUnit=="F","ft","m ") color (m->c_Help)
    @ 10,51 say "Enter area:" get nHotSptArea pict cMAX_HotSArea
    @ row(),col() say iif(m->cBasicUnit=="F"," ft²"," m²")
    read
    if lastkey() == K_ENTER
        nSemiMajor := sqrt(nHotSptArea/(nPI * m->ntShape))
        keyboard chr(K_CTRL_Y) + alltrim(str(nSemiMajor)) + chr(K_HOME)
    endif
elseif cVar == "NTSAMPLEAREA"
    * Calculates m² or ft² from total sample acres.
    scroll(11,49,14,78)
    @ 11,49 to 14,78
    @ 12,50 say " Convert acres to " + ;
        iif(m->cBasicUnit=="F","ft²","m² ") + space(7) color (m->c_Help)
    @ 13,51 say "Enter total acres:" get nTotalAcres pict cMAX_Acres
    read
    if lastkey() == K_ENTER
        * Conversion factors from CRC Handbook of Chem./Phs., 1981/82, p. F-282.

```

Code File: EGPCScrn.Prg

```

nNewArea := ;
  if(m->cBasicUnit=="F",nTotalAcres*43560.0,nTotalAcres*4046.8564)
  if nNewArea <= nMAX_SampleArea
    keyboard chr(K_CTRL_Y) + alltrim(str(nNewArea)) + chr(K_HOME)
  else
    tone(440,1)
    @13,51 say "Answer to large: " + NumTrim(nNewArea)
    inkey(3)
  endif
endif
endif
else
  scroll(23,1,23,78)
  @23,2 say "Call F10 from axis or total area fields."
  tone(440,1)
  inkey(2)
endif

set key K_F10 to F10_Key()
restscreen(0,0,24,79,cTmpScn)
return (NIL)
*** End of Func: F10_Key()

*****
Function GetCostGrd(cGridType, cVerDate)
* Searches for a grid size that produces a given cost.
* Only searches for grids with L/G ratios between
* 0.10 and 3.0, i.e., the grid size is between L/3 and 10*L.
* Currently uses a modification of the bisection method for root finding.
* See "Applied Numerical Analysis", 4th Ed., by Gerald and Wheatley p. 7.
* Input:  cGridType = "S", "R", or "T" for square, rectangular,
*         or triangular grids.
*         cVerDate = Version date.
* Returns: NIL
* Error:   Aborts if cGridType == NIL.
*
* The specified cost of sampling will be matched by the calculated grid
* cost to within ± < sample cost.

#define nMAXC_ITERS 25          // Max cost search iterations

static nHotSptArea := 25.0      // Hot spot area
local ntHotSptArea := nHotSptArea // Temp value
static nSemiMajor := 2.82      // Length of semi-major axis
local ntSemiMajor := nSemiMajor // Temp value
static nAngle := 0.0          // Orientation angle of hot spot to grid
local ntAngle := nAngle       // Temp value
static nRecRatio := 2.0        // Rectangular grid long/short ratio.
local ntRecRatio := nRecRatio // Temp value
static nShape := 1.0           // Shape, minor/major axis
static nSampleArea := 7000     // Total area to sample
local ntSampleArea := nSampleArea // Temp value
static nSampleCost := 700      // Cost for one sample
local ntSampleCost := nSampleCost // Temp value
static nDesirdCost := 200000    // Desired cost
local ntDesirdCost := nDesirdCost // Temp value
static cOutFile := "Screen.Out" // Screen output file

local nGTyp := 1              // Grid type requested
local nGSizeFnd := 0          // Grid size found
local nGLong := 0             // Long side for rec grids
local nCol := 0               // Scratch column()
local nSICounter := 0         // Search iterations counter
local nRACounter := 0         // Random angle counter
local nPrbHtFnd := 0          // Prob. of hit found
local nPrbNoHit := 1          // Probability of zero hits, P(0)
local nCrntAngle := 0         // Current angle, used for "random" angle

```

Code File: EGPCScr.Prg

```

local nLrgstAngle := 0          // Largest angle, used for "random" angle
local nProbSum    := 0          // Summing var., used for "random" angle
local nSmlGrid    := 0          // Small grid size
local nNoHitSmlGrd := 1         // P(0) for small grid
local nLrgGrid    := 0          // Large grid size
local nNoHitLrgGrd := 1         // P(0) for large grid
local nIntrGrid   := 0          // Interpolated grid size
local nNoHitIntGrd := 1         // P(0) for interpolated grid
local nDiffCost   := 0          // Current diff. from desired cost
local nCurrCost   := 0          // Current cost
local nCostClst   := 0          // Best estimate of cost
local nGridCellArea := 0        // Area of one grid cell
local nNumSamples := 0          // Number of samples required

* Misc vars.
local lOutFile    := .F.        // Use output file flag
local lWriteHeader := .F.        // Write file header flag
local cStatus     := "OK"       // Status msg for output file data
local lDone       := .F.        // Loop flag
local lConverge   := .F.        // Convergence flag
local lAborted    := .F.        // Esc key abort
local lPastMaxIt  := .F.        // Exceeded nMAXC_ITERS flag
local lOKProb     := .T.        // Solvable problem specs flag
local nKeyPress   := 0          // User key press
local getlist     := {}         // Stops compiler warnings

private ntShape    := nShape     // private for F10 key function

if cGridType == NIL
  * Input error: no grid typ passed in.
  return (NIL)
endif

* Uppcase function argument.
cGridType := upper(cGridType)

* Get screen output file.
cOutFile := GetScnOutFile(@lOutFile, @lWriteHeader)

* Display screen title.
DispTitle(cGridType,"C",cOutFile, lOutFile)

do while ! lDone
  @ 6, 2 say "Shape of the elliptical hot spot...";
  get m->ntShape pict cMAX_Shape ;
  valid ErrorUDF(ntShape <= 1.0 .and. ntShape >= 0.05, ;
    "Shape must be ≥ 0.05 and ≤ 1.0.",len(cMAX_Shape))
  @ 6,49 say "Shape = short axis/long axis."
  @ 7,49 say "F10 calculates axis from area." color(m->c_Help)
  @ 7, 2 say "Length of semi-major axis.....";
  get ntSemiMajor pict cMAX_SemiMajor ;
  valid ErrorUDF(ntSemiMajor > 0.0, ;
    "Length must be > 0.0.",len(cMAX_SemiMajor))
  @row(),col() say iif(m->cBasicUnit=="F","ft","m")
  @ 9,49 say ' 99.0° for "random" angles.'

  if cGridType == "R"
    * Rectangular grid.
    @ 8, 2 say "Angle of orientation to grid.....";
    get ntAngle pict cMAX_Angle ;
    valid ErrorUDF(ntAngle >= 0 .and. ntAngle <= 90.0 .or. ntAngle == 99, ;
      "Angle must be 0° to 90° or 99°=random.",len(cMAX_Angle))
    @ 8,col() say ""
    @ 8,49 say "Angle can be 0° to 90°. Use"
    @ 9, 2 say "Long side/short side ratio.....";
    get ntRecRatio pict cMAX_RecRatio ;

```

Code File: EGPCScrn.Prg

```

    valid ErrorUDF(ntRecRatio > 1.0, ;
    "Ratio must be > 1.0.",len(cMAX_RecRatio))

elseif cGridType == "S"
    * Square grid.
    @ 8, 2 say "Angle of orientation to grid.....:" ;
    get ntAngle pict cMAX_Angle ;
    valid ErrorUDF(ntAngle >= 0 .and. ntAngle <= 45.0 .or. ntAngle == 99,;
    "Angle must be 0° to 45° or 99°=random.",len(cMAX_Angle))
    @ 8,col() say ""
    @ 8,49 say "Angle can be 0° to 45°. Use"
elseif cGridType == "T"
    * Triangular grid.
    @ 8, 2 say "Angle of orientation to grid.....:" ;
    get ntAngle pict cMAX_Angle ;
    valid ErrorUDF(ntAngle >= 0 .and. ntAngle <= 30.0 .or. ntAngle == 99,;
    "Angle must be 0° to 30° or 99°=random.",len(cMAX_Angle))
    @ 8,col() say ""
    @ 8,49 say "Angle can be 0° to 30°. Use"
endif

@10, 2 say "Total area to sample.....:" ;
get ntSampleArea pict cMAX_SampleArea ;
valid ErrorUDF(ntSampleArea>0.0,"Area must be > 0.0",len(cMAX_SampleArea))
@row(),col() say if(m->cBasicUnit=="F","ft","m")
@10,25 say "F10 = Acres" color(m->c_Help)
@11, 2 say "Individual sample cost.....$:" ;
get ntSampleCost pict cMAX_SampleCost ;
valid ErrorUDF(ntSampleCost>0.0,"Cost must be > 0.0",len(cMAX_SampleCost))
@12, 2 say "Desired cost of grid.....$:" ;
get ntDesirdCost pict cDESIRD_COST ;
valid ErrorUDF(ntDesirdCost >= ntSampleCost, ;
    "Total cost must be ≥ Sample cost.",len(cDESIRD_COST))
@11,49 say "Program will search for cost"
@12,49 say "with error < ± 1 sample cost."

@22, 0 say "|"
@22,79 say "|"
@22,1 to 22,78

@23,2 say "Enter = Continue   Esc = Abort" + space(44) // erase msg

set key K_F10 to F10_Key()
read
set key K_F10 to
nKeyPress := lastkey()

* Abort, Write Data, etc...
do case
case (nKeyPress == K_ESC)
    * Esc key pressed
    if YN_MsgBox("Abort current data entry session? Y/N")
        if lOutFile
            * Close out file.
            close alternate
        endif
        lDone := .T.
    endif
case (nKeyPress == K_ENTER .or. nKeyPress == K_CTRL_W)
    * Enter key or Ctrl-W pressed. [Ctrl-W currently not documented.]
    * Save changes to static vars.
    scroll(13,1,21,78)
    nShape      := m->ntShape
    nSemiMajor  := ntSemiMajor
    nAngle      := ntAngle
    nDesirdCost := ntDesirdCost

```

Code File: EGPCScrn.Prg

```

nRecRatio := ntRecRatio

if cGridType == "S"
  * Sqare grid.
  nGTyp := 1
  nLrgstAngle := 45 // For "random" angle
elseif cGridType == "R"
  * Rect. grid.
  nGTyp := 3
  nLrgstAngle := 90 // For "random" angle
elseif cGridType == "T"
  * Tri. grid.
  nGTyp := 2
  nLrgstAngle := 30 // For "random" angle
endif

* Cost related vars.
nSampleArea := ntSampleArea
nSampleCost := ntSampleCost

*-----| Find grid size for given cost |-----*
* Below are grid size restrictions to stay in reasonable search.
nSmlGrid := 0.33334 * nSemiMajor
* Below same as L/G of 0.10.
nLrgGrid := 10.0 * nSemiMajor

scroll(23,1,23,78)
setcolor("W+M") // Force blinking with *
@23,2 say "Calculating"
setcolor(m->C_Normal)
nCol := col()+1
@22,nCol say "┌"
@23,nCol say "│"
@24,nCol say "└"
@23,52 say "Esc = Stop Calculations..."
setcursor(SC_NONE)

* Initialize counters
nSICounter := 0 // Search iterations
nRACounter := 0 // Random angle iterations
@14,38 say "Cost search iterations.: "
if nAngle == 99.0
  @15,38 say "Random angle iterations: "
endif
@17,38 say "Cost search is usually less"
@18,38 say "than 16 iterations."

* Initialize flags.
lConverg := .F. // .T. if search converges
lAborted := .F. // .T. if Esc key abort
lPastMaxIt := .F. // .T. if past max iterations
lOKProb := .T. // .F. if problem can't be solved

* Start searching.
do while ! lConverg .and. ! lAborted .and. ! lPastMaxIt .and. lOKProb
  nSICounter++ // Increment search counter
  @14,64 say NumTrim(nSICounter)+ "/" + ;
  ltrim(str(nMAXC_ITERS)) + " maximum."

  * GET PROB. AND COST FOR LARGE GRID.
  if nAngle != 99.0
    * Non-random single angle case.
    nNoHitLrgGrd := ElipGrid(nSemiMajor,nShape,nAngle,nLrgGrid, ;
      nGTyp, nRecRatio)
  else
    * Sum up multiple angle results, random case.

```

Code File: EGPCScrn.Prg

```

nProbSum := 0.0
for nCrntAngle = 0 to nLrgstAngle
  nProbNoHit := ElipGrid(nSemiMajor, nShape, nCrntAngle, ;
    nLrgGrid, nGTyp, nRecRatio)
  nProbSum := nProbSum + nProbNoHit
  nRACounter++
  @15,64 say ltrim(str(nRACounter))
  * Esc key abort, only used with random angles.
  if inkey() == K_ESC
    lAborted := .T.
    exit // exit for/next loop
  endif
next nCrntAngle
if lAborted
  loop
endif

* Calculate average.
nNoHitLrgGrd := nProbSum/(nLrgstAngle+1)
endif

* Check if we met cost criteria with large grid.
* First do area calculations.
if cGridType == "R"
  * Rect. grid.
  nGLong := nLrgGrid * nRecRatio
else
  * Sq. or Tri. grid.
  nGLong := nLrgGrid
endif

* Required number of samples is approximate.
* Based on (EPA. 1989. "Methods for Eval. the Attainment of Cleanup
* Standards Volume 1: Soils and Solid Media", p. 9-7.
* Calculate grid cell area.
if cGridType == "T"
  * Triangular grid.
  * Grid cell area is now, 09/02/94, area of the rhombus formed
  * from 2 of the equilateral triangles.
  * A = height * base = sin(60°) * base * base = 0.87 * base^2.
  nGridCellArea := 0.866025404 * nLrgGrid * nLrgGrid
else
  * Sq. or rec. grid.
  nGridCellArea := nLrgGrid * nGLong
endif

* This formula is approx. See (EPA 1989, 9-7).
* Ceiling function rounds number of samples up.
nNumSamples := ceiling(nSampleArea/nGridCellArea)

nCurrCost := nNumSamples * nSampleCost

* Can we quit yet?
nDiffCost := abs(nDesirdCost - nCurrCost)
if nDiffCost < nSampleCost
  * Met error criteria with current large grid.
  * Exit grid search.
  nCostClst := nCurrCost
  nGSizeFnd := nLrgGrid
  nPrbHitFnd := 1.0 - nNoHitLrgGrd
  lConverg := .T.
  loop
endif

* Will grid size need to be larger than 3 * semi-major axis?
* If so, an L/G ratio > 3.0 would be required.
* If first search with largest grid can't get down to the
* desired cost, no need to search farther.

```

Code File: EGPCScrn.Prg

```

if nSICounter == 1 .and. (nCurrCost > nDesirdCost)
  * Quit searching.
  LOKProb := .F.
  loop
endif

* GET PROB. FOR SMALL GRID.
if nAngle != 99.0
  * Non-random single angle case.
  nNoHitSmlGrd := ElipGrid(nSemiMajor,nShape,nAngle,nSmlGrid, ;
    nGTyp, nRecRatio)
else
  * Sum up multiple angle results, random case.
  nProbSum := 0.0
  for nCrntAngle = 0 to nLrgstAngle
    nProbNoHit := ElipGrid(nSemiMajor, nShape, nCrntAngle, ;
      nSmlGrid, nGTyp, nRecRatio)
    nProbSum := nProbSum + nProbNoHit
    nRACounter++
    @15,64 say ltrim(str(nRACounter))
    * Esc key abort, only used with random angles.
    if inkey() == K_ESC
      lAborted := .T.
      exit // exit for/next loop
    endif
  next nCrntAngle
  if lAborted
    loop
  endif

  * Calculate average.
  nNoHitSmlGrd := nProbSum/(nLrgstAngle+1)
endif

* Check if we met cost criteria with small grid.
* First do area calculations.
if cGridType == "R"
  * Rect. grid.
  nGLong := nSmlGrid * nRecRatio
else
  * Sq. or Tri. grid.
  nGLong := nSmlGrid
endif

* Required number of samples is approximate.
* Based on (EPA. 1889. "Methods for Eval. the Attainment of Cleanup
* Standards Volume 1: Soils and Solid Media", p. 9-7.
* Calculate grid cell area.
if cGridType == "T"
  * Triangular grid.
  * Grid cell area is now, 09/02/94, area of the rhombus formed
  * from 2 of the equilateral triangles.
  * A = height * base = sin(60°) * base * base = 0.87 * base^2.
  nGridCellArea := 0.866025404 * nSmlGrid * nSmlGrid
else
  * Sq. or rec. grid.
  nGridCellArea := nSmlGrid * nGLong
endif

* This formula is approx. See (EPA 1989, 9-7).
* Ceiling function rounds number of samples up.
nNumSamples := ceiling(nSampleArea/nGridCellArea)

nCurrCost := nNumSamples * nSampleCost

* Can we quit with current small grid?
nDiffCost := abs(nDesirdCost - nCurrCost)
if nDiffCost < nSampleCost

```

Code File: EGPCScrn.Prg

```

    * Met error criteria with current small grid.
    * Exit grid search.
    nCostClt := nCurrCost
    nGSizeFnd := nSmlGrid
    nPrbHtFnd := 1.0 - nNoHitSmlGrd
    lConverg := .T.
  loop
endif

* Will grid size need to be smaller than 1/3 * semi-major axis?
* If so, an L/G ratio < 1/3 would be required.
* If first search with smallest grid can't get up to the
* desired cost, no need to search farther.
if nSlCounter == 1 .and. (nCurrCost < nDesirdCost)
  * Quit searching.
  lOKProb := .F.
  loop
endif

* Bisection method, (Gerald and Wheately 1989, 7)
nIntrGrid := (nLrgeGrid + nSmlGrid)/2

* GET PROB. AND COST FOR INTERPOLATED GRID.
if nAngle != 99.0
  * Non-random single angle case.
  nNoHitIntGrd := ElipGrid(nSemiMajor,nShape,nAngle,nIntrGrid, ;
    nGTyp, nRecRatio)
else
  * Sum up multiple angle results, random case.
  nProbSum := 0.0
  for nCrntAngle = 0 to nLrgstAngle
    nProbNoHit := ElipGrid(nSemiMajor, nShape, nCrntAngle, ;
      nIntrGrid, nGTyp, nRecRatio)
    nProbSum := nProbSum + nProbNoHit
    nRACounter++
    @15,64 say ltrim(str(nRACounter))
    * Esc key abort, only used with random angles.
    if inkey() == K_ESC
      lAborted := .T.
      exit // exit for/next loop
    endif
  next nCrntAngle
  if lAborted
    loop
  endif

  * Calculate average.
  nNoHitIntGrd := nProbSum/(nLrgstAngle+1)
endif

* Check if we met cost criteria with interpolated grid.
* First do area calculations.
if cGridType == "R"
  * Rect. grid.
  nGLong := nIntrGrid * nRecRatio
else
  * Sq. or Tri. grid.
  nGLong := nIntrGrid
endif

* Required number of samples is approximate.
* Based on (EPA. 1989. "Methods for Eval. the Attainment of Cleanup
* Standards Volume 1: Soils and Solid Media", p. 9-7.
* Calculate grid cell area.
if cGridType == "T"
  * Triangular grid.
  * Grid cell area is now, 09/02/94, area of the rhombus formed

```

Code File: EGPCScrn.Prg

```

    * from 2 of the equilateral triangles.
    * A = height * base = sin(60°) * base * base = 0.87 * base^2.
    nGridCellArea := 0.866025404 * nIntrGrid * nIntrGrid
  else
    * Sq. or rec. grid.
    nGridCellArea := nIntrGrid * nGLong
  endif
  * This formula is approx. See (EPA 1989, 9-7).
  * Ceiling function rounds number of samples up.
  nNumSamples := ceiling(nSampleArea/nGridCellArea)

  nCurrCost := nNumSamples * nSampleCost

  * Can we quit with current interpolated grid?
  nDiffCost := abs(nDesirdCost - nCurrCost)
  if nDiffCost < nSampleCost
    * Met error criteria with current interpolated grid.
    * Exit grid search.
    nCostClt := nCurrCost
    nGSizeFnd := nIntrGrid
    nPrbHitFnd := 1.0 - nNoHitIntGrd
    lConverg := .T.
    loop
  endif

  * Update large or small search grid sizes.
  * This is a difference from linear interpolation and bisection
  * methods. They look for sign changes of f(x). In root search
  * case, f(x) values will be changing about 0.0.
  * We look at whether our current f(x) for the interpolated grid
  * is larger than the desired value.
  if nCurrCost > nDesirdCost
    nSmlGrid := nIntrGrid
  else
    nLrgeGrid := nIntrGrid
  endif

  * Have we reached max iterations?
  if nSICounter == nMAXC_ITERS
    * Failed to converge.
    lPastMaxit := .T.
    loop
  endif
enddo

* Clean up calculating msg.
scroll(23,1,23,78)
@22,nCol say "_"
@24,nCol say "_"
tone(440,1)
setcursor(SC_NORMAL)
*-----*

* Display results.
setcolor(m->C_Help)
if lPastMaxit .or. lAborted
  * Failed to converge msg. or Esc key aborted.
  scroll(17,1,21,78)
  @17,1 to 21,78 double
  if lPastMaxit
    @18,2 say " Failed to converge." color(m->C_Error)
  elseif lAborted
    @18,2 say " Calculations aborted..." color(m->C_Error)
  endif
  @19,2 say " Last interpolated grid estimate: " + ;
  ltrim(str(nIntrGrid,12,4))

```

Code File: EGPCScrn.Prg

```

@20,2 say " Last calculated cost.....: $" + ;
  ltrim(trans(nNumSamples*nSampleCost,cMAX_TotalCost))

elseif ! LOKProb
  * User input data require grid size out of range.
  scroll(17,1,21,78)
  @17,1 to 21,78 double
  @18,2 say " Data out of range." color(m->C_Error)
  @19,2 say " The input data require a grid size that is out of " + ;
    "the search range of the"
  @20,2 say " program. Will only search for grid sizes between" + ;
    " L/3 s Grid s 3 * L."
else
  * Found grid msg.
  scroll(14,1,21,78)
  @13,1 to 21,78 double
  if cGridType == "R"
    * Rect. grid.
    nGLong := nGSizeFnd * nRecRatio
    @14,2 say " Grid size found, long side      = " + ;
      ltrim(str(nGLong,10,3))
    @row(),col() say iif(m->cBasicUnit=="F"," ft", " m")
    @15,2 say " Grid size found, short side    = " + ;
      ltrim(str(nGSizeFnd,10,3))
  else
    * Sq. or Tri. grid.
    nGLong := nGSizeFnd
    @14,2 say " Grid size found                  = " + ;
      ltrim(str(nGSizeFnd,10,3))
  endif

  @row(),col() say iif(m->cBasicUnit=="F"," ft", " m")

  @14,49 say "Grid search iterations.: " + NumTrim(nSICounter)
  if nAngle == 99.0
    @15,49 say "Random angle iterations: " + NumTrim(nRACounter)
  endif

  @16,2 say " Probability of hitting hot spot = " + ;
    ltrim(str(100*nPrbHtFnd,6,1)) + "%"
endif

if nSampleArea > 0 .and. lConverg
  * Required number of samples is approximate.
  * Based on (EPA. 1989. "Methods for Eval. the Attainment of Cleanup
  * Standards Volume 1: Soils and Solid Media", p. 9-7.
  * Calculate grid cell area.
  if cGridType == "T"
    * Triangular grid.
    * Grid cell area is now, 09/02/94, area of the rhombus formed
    * from 2 of the equilateral triangles.
    * A = height * base = sin(60°) * base * base = 0.87 * base^2.
    nGridCellArea := 0.866025404 * nGSizeFnd * nGSizeFnd
  else
    * Sq. or rec. grid.
    nGridCellArea := nGSizeFnd * nGLong
  endif

  * This formula is approx. See (EPA 1989, 9-7).
  * Ceiling function rounds number of samples up.
  nNumSamples := ceiling(nSampleArea/nGridCellArea)

  @18,2 say " Required number of samples          = " + ;
    ltrim(trans(nNumSamples,cMAX_Samples))
  @19,3 say ;
    "Required number of samples is approximate. " color "W/RB"
  @20,2 say " Total cost for above number of samples = $" + ;

```

Code File: EGPCScrn.Prg

```

        ltrim(trans(nNumSamples*nSampleCost,cMAX_TotalCost))
    endif

    if lOutFile .and. lConverg
        * Write to output file.
        * Pass lWriteHeader by reference, WriteData will update it.
        WriteData(@lWriteHeader,cVerDate,cOutFile,cGridType,nRecRatio, ;
            nSemiMajor,nGSizeFnd,nShape,nAngle,nPrbHtFnd,nSampleArea, ;
            nNumSamples,nSampleCost)
    elseif lOutFile .and. (lAborted .or. ! lConverg)
        * Write data for abort or failed to converge.
        nGSizeFnd := nIntrGrid
        nPrbHtFnd := 1.0 - nNoHitIntGrd
        cStatus := if(lAborted," ABRT"," FAIL")
        WriteData(@lWriteHeader,cVerDate,cOutFile,cGridType,nRecRatio, ;
            nSemiMajor,nGSizeFnd,nShape,nAngle,nPrbHtFnd,nSampleArea, ;
            nNumSamples,nSampleCost,cStatus)
    endif
    setcolor(m->C_Normal)
otherwise
    * Will loop back.
    lDone := .F.
endcase (LKey == )
enddo
return (NIL)
*** End of Func: GetCostGrd()

*****
Function GetGridSiz(cGridType, cVerDate)
* Searches for a grid size that produces a given probability.
* Uses a modification of the linear interpolation method for nonlinear equation
* root finding for most searching. Uses a modification of the bisection
* method when searching for sizes producing very small or large hot spot
* miss probabilities. See "Applied Numerical Analysis", 4th Ed.,
* by Gerald and Wheatley pp. 5-10.
* Only searches for grids with L/G ratios between
* 0.10 and 3.0, i.e., the grid size is between L/3 and 10*L.
* Input:  cGridType = "S", "R", or "T" for square, rectangular,
*          or triangular grids.
*          cVerDate = Version date.
* Returns: NIL
* Error:  Aborts if cGridType == NIL.
* The specified prob. of a hit will be matched by the chosen grid to within
*  $\pm$  nERR_CRITERIA. This corresponds to, e.g.,
* 90.0%  $\pm$  0.05%. Or 89.95% < calculated value < 90.05%.
#define nERR_CRITERIA 0.000499999 // Worked better than 0.0005 in some cases
#define nMAX_ITERS 25 // Max search iterations

static nHotSptArea := 25.0 // Hot spot area
local ntHotSptArea := nHotSptArea // Temp value
static nSemiMajor := 2.82 // Length of semi-major axis
local ntSemiMajor := nSemiMajor // Temp value
static nAngle := 0.0 // Orientation angle of hot spot to grid
local ntAngle := nAngle // Temp value
static nRecRatio := 2.0 // Rectangular grid long/short ratio.
local ntRecRatio := nRecRatio // Temp value
static nShape := 1.0 // Shape, minor/major axis
static nSampleArea := 0 // Total area to sample
local ntSampleArea := nSampleArea // Temp value
static nSampleCost := 0 // Cost for one sample
local ntSampleCost := nSampleCost // Temp value
static nDesirdProb := 95.0 // Desired probability
local ntDesirdProb := nDesirdProb // Temp value
static cOutFile := "Screen.Out" // Screen output file

local nGTyp := 1 // Grid type requested

```

Code File: EGPCScrn.Prg

```

local nGSizeFnd      := 0           // Grid size found
local nGLong         := 0           // Long side for rec grids
local nCol           := 0           // Scratch column()
local nSICounter     := 0           // Search iterations counter
local nRACounter     := 0           // Random angle counter
local nDesirdNoHit   := 0           // Desired prob. of missing
local nPrbHtFnd      := 0           // Prob. of hit found
local nProbNoHit     := 1           // Probability of zero hits, P(0)
local nCrntAngle     := 0           // Current angle, used for "random" angle
local nLrgstAngle    := 0           // Largest angle, used for "random" angle
local nPrbSum        := 0           // Summing var., used for "random" angle
local nSmelGrid      := 0           // Small grid size
local nNoHitSmlGrd   := 1           // P(0) for small grid
local nDiffSml       := 0           // Small grid diff from desired P(0)
local nLrgGrid       := 0           // Large grid size
local nNoHitLrgGrd   := 1           // P(0) for large grid
local nDiffLrg       := 0           // Large grid diff from desired P(0)
local nIntrGrid      := 0           // Interpolated grid size
local nNoHitIntGrd   := 1           // P(0) for interpolated grid
local nDiffInt       := 0           // Interpolated grid diff from desired P(0)
local nGridCellArea  := 0           // Area of one grid cell
local nNumSamples    := 0           // Number of samples required

* Misc vars.
local lOutFile       := .F.         // Use output file flag
local lWriteHeader    := .F.         // Write file header flag
local cStatus        := "OK"        // Status msg for output file data
local lDone          := .F.         // Loop flag
local lConverge       := .F.         // Convergence flag
local lAborted        := .F.         // Esc key abort
local lPastMaxit      := .F.         // Exceeded nMAX_ITERS flag
local lOKProb         := .T.         // Solvable problem specs flag
local nKeyPress       := 0           // User key press
local getlist         := {}          // Stops compiler warnings

private ntShape       := nShape      // private for F10 key function

if cGridType == NIL
  * Input error: no grid typ passed in.
  return (NIL)
endif

* Upcase function argument.
cGridType := upper(cGridType)

* Get screen output file.
cOutFile := GetScnOutFile(@lOutFile, @lWriteHeader)

* Display screen title.
DispTitle(cGridType,"G",cOutFile, lOutFile)

do while ! lDone
  @ 6, 2 say "Shape of the elliptical hot spot..:" ;
  get m->ntShape pict cMAX_Shape ;
  valid ErrorUDF(ntShape <= 1.0 .and. ntShape >= 0.05, ;
    "Shape must be > 0.05 and <= 1.0.",len(cMAX_Shape))
  @ 6,49 say "Shape = short axis/long axis."
  @ 7,49 say "F10 calculates axis from area." color(m->c_Help)
  @ 7, 2 say "Length of semi-major axis.....:" ;
  get ntSemiMajor pict cMAX_SemiMajor ;
  valid ErrorUDF(ntSemiMajor > 0.0, ;
    "Length must be > 0.0.",len(cMAX_SemiMajor))
  @row(),col() say iif(m->cBasicUnit=="F","ft","m")
  @ 9,49 say ' 99.0' for "random" angles.'

  if cGridType == "R"

```

Code File: EGPCScrn.Prg

```

* Rectangular grid.
@ 8, 2 say "Angle of orientation to grid.....:" ;
get ntAngle pict cMAX_Angle ;
valid ErrorUDF(ntAngle >= 0 .and. ntAngle <= 90.0 .or. ntAngle == 99,;
  "Angle must be 0° to 90° or 99°=random.",len(cMAX_Angle))
@ 8,col() say ""
@ 8,49 say "Angle can be 0° to 90°. Use"
@ 9, 2 say "Long side/short side ratio.....:" ;
get ntRecRatio pict cMAX_RecRatio ;
valid ErrorUDF(ntRecRatio > 1.0, ;
  "Ratio must be > 1.0.",len(cMAX_RecRatio))

elseif cGridType == "S"
* Square grid.
@ 8, 2 say "Angle of orientation to grid.....:" ;
get ntAngle pict cMAX_Angle ;
valid ErrorUDF(ntAngle >= 0 .and. ntAngle <= 45.0 .or. ntAngle == 99,;
  "Angle must be 0° to 45° or 99°=random.",len(cMAX_Angle))
@ 8,col() say ""
@ 8,49 say "Angle can be 0° to 45°. Use"
elseif cGridType == "T"
* Triangular grid.
@ 8, 2 say "Angle of orientation to grid.....:" ;
get ntAngle pict cMAX_Angle ;
valid ErrorUDF(ntAngle >= 0 .and. ntAngle <= 30.0 .or. ntAngle == 99,;
  "Angle must be 0° to 30° or 99°=random.",len(cMAX_Angle))
@ 8,col() say ""
@ 8,49 say "Angle can be 0° to 30°. Use"
endif

@10, 2 say "Desired probability of hitting.....:" ;
get ntDesirdProb pict cDESIRD_PROB ;
valid ErrorUDF(ntDesirdProb >= 10.0 .and. ntDesirdProb<=99.9,;
  "Prob. must be 10% to 99.9%.",len(cDESIRD_PROB))
@10,col() say "%"
@10,49 say "Use 10% to 99.9%."
@11,49 say "    Leave area and sample cost"
@12,49 say "    at 0 if cost not desired."
@11, 2 say "Total area to sample.....:" ;
get ntSampleArea pict cMAX_SampleArea
@row(),col() say iif(m->cBasicUnit=="F","ft","m")
@11,25 say "F10 = Acres" color(m->C_Help)
@12, 2 say "Individual sample cost.....$:" ;
get ntSampleCost pict cMAX_SampleCost
@22, 0 say "
@22,79 say "
@22,1 to 22,78

@23,2 say "Enter = Continue   Esc = Abort" + space(44) // erase msg

set key K_F10 to F10_Key()
read
set key K_F10 to
nKeyPress := lastkey()

* Abort, Write Data, etc...
do case
case (nKeyPress == K_ESC)
* Esc key pressed
if YN_MsgBox("Abort current data entry session? Y/N")
if !OutFile
* Close out file.
close alternate
endif
!Done := .T.
endif

```

Code File: EGPCScrn.Prg

```

case (nKeyPress == K_ENTER .or. nKeyPress == K_CTRL_W)
  * Enter key or Ctrl-W pressed. [Ctrl-W currently not documented.]
  * Save changes to static vars.
  scroll(13,1,21,78)
  nShape      := m->ntShape
  nSemiMajor  := ntSemiMajor
  nAngle      := ntAngle
  nDesirdProb := ntDesirdProb
  nRecRatio   := ntRecRatio

  if cGridType == "S"
    * Square grid.
    nGTyp      := 1
    nLrgstAngle := 45      // For "random" angle
  elseif cGridType == "R"
    * Rect. grid.
    nGTyp      := 3
    nLrgstAngle := 90      // For "random" angle
  elseif cGridType == "T"
    * Tri. grid.
    nGTyp      := 2
    nLrgstAngle := 30      // For "random" angle
  endif

  * Cost related vars.
  nSampleArea := ntSampleArea
  nSampleCost := ntSampleCost

  *-----| Calculate grid size for desired prob. |-----*
  nDesirdProb := iif(nDesirdProb==100.0,99.95,nDesirdProb)
  // Above left in code in case 100% again used as valid %.
  nDesirdNoHit := 1.0 - (nDesirdProb*0.01)
  * Below approx. same as L/G <= 3.0 restriction.
  nSmalGrid := 0.33334 * nSemiMajor
  * Below same as L/G of 0.10.
  nLrgeGrid := 10.0 * nSemiMajor

  scroll(23,1,23,78)
  setcolor("W+/W*")      // Force blinking
  @23,2 say "Calculating"
  setcolor(m->C_Normal)
  nCol := col()+1
  @22,nCol say "┌"
  @23,nCol say "│"
  @24,nCol say "└"
  @23,52 say "Esc = Stop Calculations..."
  setcursor(SC_NONE)

  * Keep calculating until error is less than nERR_CRITERIA.
  nSICounter := 0      // Search iterations
  nRACounter := 0      // Random angle iterations
  @14,38 say "Grid search iterations.: "
  if nAngle == 99.0
    @15,38 say "Random angle iterations: "
  endif
  @17,38 say "Grid search is usually less"
  @18,38 say "than 16 iterations."

  lConverg := .F.      // .T. if search converges
  lAborted := .F.      // .T. if Esc key abort
  lPastMaxIt := .F.    // .T. if past max iterations
  lOKProb := .T.       // .F. if problem can't be solved
  do while ! lConverg .and. ! lAborted .and. ! lPastMaxIt .and. ! lOKProb
    nSICounter++      // Increment search counter
    @14,64 say NumTrim(nSICounter)+ "/" + ;
    ltrim(str(nMAX_ITERS)) + " maximum."

```

Code File: EGPCScrn.Prg

```

* GET PROB. FOR LARGE GRID.
if nAngle != 99.0
  * Non-random single angle case.
  nNoHitLrgGrd := ElipGrid(nSemiMajor,nShape,nAngle,nLrgeGrid, ;
    nGTyp, nRecRatio)
else
  * Sum up multiple angle results, random case.
  nProbSum := 0.0
  for nCrntAngle = 0 to nLrgstAngle
    nProbNoHit := ElipGrid(nSemiMajor, nShape, nCrntAngle, ;
      nLrgeGrid, nGTyp, nRecRatio)
    nProbSum := nProbSum + nProbNoHit
    nRACounter++
    @15,64 say ltrim(str(nRACounter))
    * Esc key abort, only used with random angles.
    if inkey() == K_ESC
      lAborted := .T.
      exit // exit for/next loop
    endif
  next nCrntAngle
  if lAborted
    loop
  endif

  * Calculate average.
  nNoHitLrgGrd := nProbSum/(nLrgstAngle+1)
endif

* Check if we met error criteria with large grid.
nDiffLrg := abs(nDesirdNoHit - nNoHitLrgGrd)
if nDiffLrg < nERR_CRITERIA
  * Met error criteria with current large grid.
  * Exit grid search.
  nGSizeFnd := nLrgeGrid
  nPrbHitFnd := 1.0 - nNoHitLrgGrd
  lConverg := .T.
  loop
endif

* GET PROB. FOR SMALL GRID.
if nAngle != 99.0
  * Non-random single angle case.
  nNoHitSmlGrd := ElipGrid(nSemiMajor,nShape,nAngle,nSmlGrid, ;
    nGTyp, nRecRatio)
else
  * Sum up multiple angle results, random case.
  nProbSum := 0.0
  for nCrntAngle = 0 to nLrgstAngle
    nProbNoHit := ElipGrid(nSemiMajor, nShape, nCrntAngle, ;
      nSmlGrid, nGTyp, nRecRatio)
    nProbSum := nProbSum + nProbNoHit
    nRACounter++
    @15,64 say ltrim(str(nRACounter))
    * Esc key abort, only used with random angles.
    if inkey() == K_ESC
      lAborted := .T.
      exit // exit for/next loop
    endif
  next nCrntAngle
  if lAborted
    loop
  endif

  * Calculate average.
  nNoHitSmlGrd := nProbSum/(nLrgstAngle+1)
endif

```

Code File: EGPCScrn.Prg

```

* Will grid size need to be smaller than 1/3 semi-major axis?
* If so, an L/G ratio > 3.0 would be required.
* If first search with smallest grid can't get the
* desired prob. of hitting, no need to search farther.
if nSlCounter == 1 .and. (nNoHitSmlGrd > nDesirdNoHit)
  * Quit searching.
  LOKProb := .F.
  loop
endif

* Check if we met error criteria with small grid.
nDiffSml := abs(nDesirdNoHit - nNoHitSmlGrd)
if nDiffSml < nERR_CRITERIA
  * Met error criteria with current small grid.
  * Exit grid search.
  nGSizeFnd := nSmlGrid
  nPrbHtFnd := 1.0 - nNoHitSmlGrd
  lConverg := .T.
  loop
endif

* Get interpolated (or average for small/large probs.) grid size.
if nDesirdNoHit >= 0.04 .and. nDesirdNoHit <= 0.50
  * Formula based on a modification to the linear interpolation
  * method, (Gerald and Wheatley 1989, 10)
  * 0.04 and 0.50 limits set by trial and error.
  nIntrGrid := nLrgGrid - (nLrgGrid - nSmlGrid) * ;
    (nNoHitLrgGrd - nDesirdNoHit) / (nNoHitLrgGrd - nNoHitSmlGrd)
else
  * Bisection method, (Gerald and Wheatley 1989, 7)
  * Works better than linear interp. on ends of prob. curve.
  nIntrGrid := (nLrgGrid + nSmlGrid)/2
end

* GET PROB. FOR INTERPOLATED GRID.
if nAngle != 99.0
  * Non-random single angle case.
  nNoHitIntGrd := ElipGrid(nSemiMajor,nShape,nAngle,nIntrGrid, ;
    nGTyp, nRecRatio)
else
  * Sum up multiple angle results, random case.
  nProbSum := 0.0
  for nCrntAngle = 0 to nLrgstAngle
    nProbNoHit := ElipGrid(nSemiMajor, nShape, nCrntAngle, ;
      nIntrGrid, nGTyp, nRecRatio)
    nProbSum := nProbSum + nProbNoHit
    nRACounter++
    @15,64 say ltrim(str(nRACounter))
    * Esc key abort, only used with random angles.
    if inkey() == K_ESC
      lAborted := .T.
      exit // exit for/next loop
    endif
  next nCrntAngle
  if lAborted
    loop
  endif

  * Calculate average.
  nNoHitIntGrd := nProbSum/(nLrgstAngle+1)
endif

* Check if we met error criteria with interpolated grid.
nDiffInt := abs(nDesirdNoHit - nNoHitIntGrd)
if nDiffInt < nERR_CRITERIA
  * Met error criteria with current interpolated grid.

```

Code File: EGPCScrn.Prg

```

    * Exit grid search.
    nGSizeFnd := nIntrGrid
    nPrbHtFnd := 1.0 - nNoHitIntGrd
    lConverg := .T.
    loop
  endif

  * Update large or small search grid sizes.
  * This is a difference from linear interpolation and bisection
  * methods. They look for sign changes of f(x). In root search
  * case, f(x) values will be changing about 0.0.
  * We look at whether our current f(x) for the interpolated grid
  * is larger than the desired value.
  if nNoHitIntGrd > nDesirdNoHit
    nLrgeGrid := nIntrGrid
  else
    nSmalGrid := nIntrGrid
  endif

  * Have we reached max iterations?
  if nSICounter == nMAX_ITERS
    * Failed to converge.
    lPastMaxIt := .T.
    loop
  endif
enddo

* Clean up calculating msg.
scroll(23,1,23,78)
@22,nCol say "_"
@24,nCol say "_"
tone(440,1)
setcursor(SC_NORMAL)
* In case rec. grid, get long side.
nGLong := nGSizeFnd * nRecRatio
*-----*

* Display results.
setcolor(m->C_Help)
if lPastMaxIt.or. lAborted
  * Failed to converge msg. or Esc key aborted.
  scroll(17,1,21,78)
  @17,1 to 21,78 double
  if lPastMaxIt
    @18,2 say " Failed to converge." color(m->C_Error)
  elseif lAborted
    @18,2 say " Calculations aborted..." color(m->C_Error)
  endif
  @19,2 say " Last interpolated grid estimate.: " + ;
    ltrim(str(nIntrGrid,12,4))
  @20,2 say " Last calculated prob. of hitting: " + ;
    ltrim(str(100*(1-nNoHitIntGrd),12,4)) + "%"
elseif lOKProb
  * Problem specs. require grid size < 1/3 semi-major axis.
  scroll(17,1,21,78)
  @17,1 to 21,78 double
  @18,2 say " Data out of range." color(m->C_Error)
  @19,2 say " The input data require a grid size that is out of " + ;
    "the search range of the"
  @20,2 say " program. Will only search for grid sizes > 1/3 " + ;
    "length of semi-major axis."
else
  * Found grid msg.
  scroll(14,1,21,78)
  @13,1 to 21,78 double
  if cGridType == "R"

```

Code File: EGPCScrn.Prg

```

    * Rect. grid.
    nGLong := nGSizeFnd * nRecRatio
    @14,2 say " Grid size found, long side      = " + ;
        ltrim(str(nGLong,10,3))
    @row(),col() say iif(m->cBasicUnit=="F"," ft", " m")
    @15,2 say " Grid size found, short side     = " + ;
        ltrim(str(nGSizeFnd,10,3))
  else
    * Sq. or Tri. grid.
    nGLong := nGSizeFnd
    @14,2 say " Grid size found                  = " + ;
        ltrim(str(nGSizeFnd,10,3))
  endif

  @row(),col() say iif(m->cBasicUnit=="F"," ft", " m")

  @14,49 say "Grid search iterations.: " + NumTrim(nSICounter)
  if nAngle == 99.0
    @15,49 say "Random angle iterations: " + NumTrim(nRACounter)
  endif

  @16,2 say " Probability of hitting hot spot = " + ;
      ltrim(str(100*nPrbHtFnd,6,1)) + "%"
  endif

if nSampleArea > 0 .and. lConverg
  * Required number of samples is approximate.
  * Based on (EPA. 1989. "Methods for Eval. the Attainment of Cleanup
  * Standards Volume 1: Soils and Solid Media", v. 9-7.
  * Calculate grid cell area.
  if cGridType == "T"
    * Triangular grid.
    * Grid cell area is now, 09/02/94, area of the rhombus formed
    * from 2 of the equilateral triangles.
    * A = height * base = sin(60°) * base * base = 0.87 * base^2.
    nGridCellArea := 0.866025404 * nGSizeFnd * nGSizeFnd
  else
    * Sq. or rec. grid.
    nGridCellArea := nGSizeFnd * nGLong
  endif
  * This formula is approx. See (EPA 1989, 9-7).
  * Ceiling function rounds number of samples up.
  nNumSamples := ceiling(nSampleArea/nGridCellArea)

  @18,2 say " Required number of samples          = " + ;
      ltrim(trans(nNumSamples,cMAX_Samples))
  @19,3 say ;
      "Required number of samples is approximate. " color "W/RB"
  @20,2 say " Total cost for above number of samples = $" + ;
      ltrim(trans(nNumSamples*nSampleCost,cMAX_TotalCost))
endif

if lOutFile .and. lConverg
  * Write to output file.
  * Pass lWriteHeader by reference, WriteData will update it.
  WriteData(@lWriteHeader,cVerDate,cOutFile,cGridType,nRecRatio, ;
      nSemiMajor,nGSizeFnd,nShape,nAngle,nPrbHtFnd,nSampleArea, ;
      nNumSamples,nSampleCost)
elseif lOutFile .and. (lAborted.or. lConverg)
  * Write data for abort or failed to converge.
  nGSizeFnd := nIntrGrid
  nPrbHtFnd := 1.0 - nNoHitIntGrd
  cStatus := iif(lAborted," ABRT"," FAIL")
  WriteData(@lWriteHeader,cVerDate,cOutFile,cGridType,nRecRatio, ;
      nSemiMajor,nGSizeFnd,nShape,nAngle,nPrbHtFnd,nSampleArea, ;
      nNumSamples,nSampleCost,cStatus)

```

Code File: EGPCScr.Prg

```

endif
setcolor(m->C_Normal)
otherwise
* Will loop back.
lDone := .F.
endcase (LKey == )
enddo
return (NIL)
*** End of Func: GetGridSiz()

*****
Function GetProbHit(cGridType, cVerDate)
* Calculates probability of missing/hitting hot spot based on
* Singer's 1972 ELIPGRID algorithm.
* Input:  cGridType = "S", "R", or "T" for square, rectangular,
*         or triangular grids.
*         cVerDate = Version date.
* Returns: NIL
*         Aborts if cGridType == NIL.

static nHotSptArea := 25.0          // Hot spot area
local ntHotSptArea := nHotSptArea  // Temp value
static nSemiMajor := 2.82          // Length of semi-major axis
local ntSemiMajor := nSemiMajor    // Temp value
static nAngle := 99.0              // Orientation angle of hot spot to grid
local ntAngle := nAngle            // Temp value
static nRecRatio := 2.0             // Rectangular grid long/short ratio
local ntRecRatio := nRecRatio      // Temp value
static nGSize := 10.0              // G in Gilbert, grid spacing
// Short side, if rectangular grid

local ntGSize := nGSize            // Temp value
static nShape := 1.0               // Shape, minor/major axis
static nSampleArea := 0            // Total area to sample
local ntSampleArea := nSampleArea  // Temp value
static nSampleCost := 0            // Cost for one sample
local ntSampleCost := nSampleCost  // Temp value
static cOutFile := "Screen.Out"    // Screen output file

local nGTyp := 1                   // Grid type
local nGLong := nGSize              // Long side for rec grids
local nProbNoHit := 0               // Probability of zero hits, P(0)
local nProbOfAHit := 0              // Probability of at least 1 hit, 1-P(0)
local nCrntAngle := 0               // Current angle, used for "random" angle
local nLrgstAngle := 0              // Largest angle, used for "random" angle
local nProbSum := 0                 // Summing var., used for "random" angle
local nGridCellArea := 0            // Area of one grid cell
local nNumSamples := 0              // Number of samples required

* Misc vars.
local lOutFile := .F.               // Use output file flag
local lWriteHeader := .F.           // Write file header flag
local nChoice := 0                  //
local lDone := .F.                  // Loop flag
local nLKey :=                      //
local getlist := ()                 // Stops compiler warnings

private ntShape := nShape            // private for F10 key function

if cGridType == NIL
return (NIL)
endif

* Uppcase function argument.
cGridType := upper(cGridType)

* Get screen output file.

```

Code File: EGPCScrn.Prg

```

cOutFile := GetScnOutFile(@lOutFile, @lWriteHeader)

* Display screen title.
DispTitle(cGridType,"P",cOutFile, lOutFile)

do while ! lDone
  @ 6, 2 say "Shape of the elliptical hot spot..:" ;
  get m->ntShape pict cMAX_Shape ;
  valid ErrorUDF(ntShape <= 1.0 .and. ntShape >= 0.05, ;
    "Shape must be > 0.05 and < 1.0.",len(cMAX_Shape))
  @ 6,49 say "Shape = short axis/long axis."
  @ 7,49 say "F10 calculates axis from area." color(m->c_Help)
  @ 7, 2 say "Length of semi-major axis.....:" ;
  get ntSemiMajor pict cMAX_SemiMajor ;
  valid ErrorUDF(ntSemiMajor > 0.0, ;
    "Length must be > 0.0.",len(cMAX_SemiMajor))
  @row(),col() say iif(m->cBasicUnit=="F","ft","m")
  @ 9,49 say " 99.0° for average of"
  @10,49 say " multiple, "random", angles."

  if cGridType == "R"
    * Rectangular grid.
    @ 8, 2 say "Angle of orientation to grid.....:" ;
    get ntAngle pict cMAX_Angle ;
    valid ErrorUDF(ntAngle >= 0 .and. ntAngle <= 90.0 .or. ntAngle == 99,;
      "Angle must be 0° to 90° or 99°=random.",len(cMAX_Angle))
    @ 8,col() say ""
    @ 8,49 say "Angle can be 0° to 90°. Use"
    @ 9, 2 say "Length of short side of rect. grid:" ;
    get ntGSize pict cMAX_GSize ;
    valid ErrorUDF(ntGSize >= ntSemiMajor/3, ;
      "Grid size must be > Semimajor axis/3.0.",len(cMAX_GSize))
    @row(),col() say iif(m->cBasicUnit=="F","ft","m")
    @10, 2 say "Long side/short side ratio.....:" ;
    get ntRecRatio pict cMAX_RecRatio ;
    valid ErrorUDF(ntRecRatio > 1.0, ;
      "Ratio must be > 1.0.",len(cMAX_RecRatio))
  elseif cGridType == "S"
    * Square grid.
    @ 8, 2 say "Angle of orientation to grid.....:" ;
    get ntAngle pict cMAX_Angle ;
    valid ErrorUDF(ntAngle >= 0 .and. ntAngle <= 45.0 .or. ntAngle == 99,;
      "Angle must be 0° to 45° or 99°=random.",len(cMAX_Angle))
    @ 8,col() say ""
    @ 8,49 say "Angle can be 0° to 45°. Use"
    @ 9, 2 say "Length of any side of square grid.:" ;
    get ntGSize pict cMAX_GSize ;
    valid ErrorUDF(ntGSize >= ntSemiMajor/3, ;
      "Grid size must be > Semimajor axis/3.0.",len(cMAX_GSize))
    @row(),col() say iif(m->cBasicUnit=="F","ft","m")
  elseif cGridType == "T"
    * Triangular grid.
    @ 8, 2 say "Angle of orientation to grid.....:" ;
    get ntAngle pict cMAX_Angle ;
    valid ErrorUDF(ntAngle >= 0 .and. ntAngle <= 30.0 .or. ntAngle == 99,;
      "Angle must be 0° to 30° or 99°=random.",len(cMAX_Angle))
    @ 8,col() say ""
    @ 8,49 say "Angle can be 0° to 30°. Use"
    @ 9, 2 say "Length of side of triangular grid.:" ;
    get ntGSize pict cMAX_GSize ;
    valid ErrorUDF(ntGSize >= ntSemiMajor/3, ;
      "Grid size must be > Semimajor axis/3.0.",len(cMAX_GSize))
    @row(),col() say iif(m->cBasicUnit=="F","ft","m")
  endif

  @11,49 say " Leave area and sample cost"

```

Code File: EGPCScrn.Prg

```

@12,49 say "      at 0, if cost not desired."
@11, 2 say "Total area to sample.....:" ;
  get ntSampleArea pict cMAX_SampleArea
  @row(),col() say iif(m->cBasicUnit=="F","ft","m")
@11,25 say "F10 = Acres" color(m->C_Help)
@12, 2 say "Individual sample cost.....$:" ;
  get ntSampleCost pict cMAX_SampleCost
@22, 0 say " "
@22,79 say " "
@22,1 to 22,78

@23,2 say "Enter = Continue   Esc = Abort" + space(44)  // erase msg

set key K_F10 to F10_Key()
read
set key K_F10 to
nLKey := lastkey()

* Abort, Write Data, etc...
do case
  case (nLKey == K_ESC)
    * Esc key pressed
    if YN_MsgBox("Abort current data entry session? Y/N")
      if !OutFile
        * Close out file.
        close alternate
      endif
      lDone := .T.
    endif
  case (nLKey == K_ENTER .or. nLKey == K_CTRL_W)
    * Enter key or Ctrl-W pressed. [Ctrl-W currently not documented.]
    * Save changes to static vars.
    scroll(13,1,21,78)
    nShape      := m->ntShape
    nSemiMajor  := ntSemiMajor
    nAngle      := ntAngle
    nGSize      := ntGSize
    nGLong      := nGSize      // Will correct for rec grid below
    nRecRatio   := ntRecRatio

    if cGridType == "S"
      * Square grid.
      nGTyp      := 1
    elseif cGridType == "R"
      * Rect. grid.
      nGTyp      := 3
      nGLong     := nGSize * nRecRatio
    elseif cGridType == "T"
      * Tri. grid.
      nGTyp      := 2
    endif

    * Cost related vars.
    nSampleArea := ntSampleArea
    nSampleCost := ntSampleCost

    *-----| Calculate probability of no hit, P(0) |-----*
    if nAngle != 99.0
      * Calculate for a single angle.
      nProbNoHit := ElipGrid(nSemiMajor,nShape,nAngle,nGSize,nGTyp, ;
        nRecRatio)
    else
      scroll(13,1,21,78)
      @14,02 say 'Calculating average for multiple angles, i.e., "random".'
      * Calculate for average of multiple angles, i.e., "random" choice
      * in Singer's 1972 ELIPGRID.

```

Code File: EGPCScrn.Prg

```

    if nGTyp == 1
        nLrgstAngle := 45
    elseif nGTyp == 2
        * For triangular grid (hexagon).
        nLrgstAngle := 30
    elseif nGTyp == 3
        * For rectangular grid.
        nLrgstAngle := 90
    endif
    * Sum up multiple angles results.
    nProbSum := 0.0
    for nCrntAngle = 0 to nLrgstAngle
        nProbNoHit := ElipGrid(nSemiMajor,nShape,nCrntAngle,nGSize, ;
            nGTyp, nRecRatio)
        nProbSum := nProbSum + nProbNoHit
    next nCrntAngle

    * Calculate average.
    nProbNoHit := nProbSum/(nLrgstAngle+1)
endif
*-----*

* Display results.
setcolor(m->c_Help)
scroll(14,1,21,78)
@13,1 to 21,78 double
nProbOfAHit := 100 * (1.0 - nProbNoHit)
@14,2 say " Probability of hitting at least once      = " + ;
    str(nProbOfAHit,6,1)+"%"
@16,2 say " Probability of NOT hitting hot spot      = " + ;
    str(100*nProbNoHit,6,1) + "%"

* If applicable, display cost.
if nSampleArea > 0
    * Required number of samples is approximate.
    * Based on (EPA. 1989. "Methods for Eval. the Attainment of Cleanup
    * Standards Volume 1: Soils and Solid Media", p. 9-7.
    * Calculate grid cell area.
    if cGridType == "T"
        * Triangular grid.
        * Grid cell area is now, 09/02/94, area of the rhombus formed
        * from 2 of the equilateral triangles.
        * A = height * base = sin(60°) * base * base = 0.87 * base^2.
        nGridCellArea := 0.866025404 * nGSize * nGSize
    else
        * Sq. or rec. grid.
        nGridCellArea := nGSize * nGLong
    endif
    * This formula is approx. See (EPA 1989, 9-7).
    * Ceiling function rounds number of samples up.
    nNumSamples := ceiling(nSampleArea/nGridCellArea)

    @18,2 say " Required number of samples              = " + ;
        ltrim(trans(nNumSamples,cMAX_Samples))
    @19,3 say ;
        "Required number of samples is approximate. " color "W/RB"
    @20,2 say " Total cost for above number of samples = $" + ;
        ltrim(trans(nNumSamples*nSampleCost,cMAX_TotalCost))
endif

if lOutFile
    * Write to output file.
    * Pass lWriteHeader by reference, WriteData will update it.
    WriteData(@lWriteHeader,cVerDate,cOutFile,cGridType,nRecRatio, ;
        nSemiMajor,nGSize,nShape,nAngle,nProbOfAHit/100,nSampleArea, ;
        nNumSamples,nSampleCost)

```

Code File: EGPCScrn.Prg

```

endif
setcolor(m->C_Normal)

otherwise
* Will loop back.
(Done := .F.
endcase (nLKey == )
enddo
set alternate to
return (NIL)
*** End of Func: GetProbHit()

*****
Function GetSmallestArea(cGridType, cVerDate)
* Searches for smallest hot spot size that produces a given probability.
* Currently uses a modification of the bisection method for root finding.
* See "Applied Numerical Analysis", 4th Ed., by Gerald and Wheatley p. 7.
* Only searches for hot spots with L/G ratios between
* 0.10 and 3.0, i.e., the hot spot semi-major axis size, L,
* is between 0.1 * G and 3 * G, where G is given grid size.
* Input:   cGridType = "S", "R", or "T" for square, rectangular,
*          or triangular grids.
*          cVerDate = Version date.
* Returns: NIL
* Error:   Aborts if cGridType == NIL.
* The specified prob. of a hit will be matched by the chosen hot spot to within
*  $\pm < \text{nERR\_CRITERIA}$ . This corresponds to, e.g.,
* 90.0%  $\pm < 0.05\%$ . Or 89.95% < calculated value < 90.05%.

static nAngle      := 0.0           // Orientation angle of hot spot to grid
local ntAngle      := nAngle        // Temp value
static nRecRatio   := 2.0           // Rectangular grid long/short ratio.
local ntRecRatio   := nRecRatio      // Temp value
static nGSize      := 10.0          // G in Gilbert, grid spacing
// Short side, if rectangular grid

local ntGSize      := nGSize        // Temp value
static nShape      := 1.0           // Shape, minor/major axis
static nSampleArea := 0             // Total area to sample
local ntSampleArea := nSampleArea   // Temp value
static nSampleCost := 0             // Cost for one sample
local ntSampleCost := nSampleCost    // Temp value
static nDesirdProb := 95.0          // Desired probability
local ntDesirdProb := nDesirdProb    // Temp value
static cOutFile    := "Screen.Out"  // Screen output file

local nGTyp        := 1            // Grid type requested
local nHS_L_Fnd    := 0            // Hot spot L value found, semi-major axis
local nGLong       := 0            // Long side for rec grids
local nCol         := 0            // Scratch column()
local nSICounter   := 0            // Search iterations counter
local nRACounter   := 0            // Random angle counter
local nDesirdNoHit := 0            // Desired prob. of missing
local nPrbHitFnd   := 0            // Prob. of hit found
local nProbNoHit   := 1            // Probability of zero hits, P(0)
local nCrntAngle   := 0            // Current angle, used for "random" angle
local nLrgstAngle  := 0            // Largest angle, used for "random" angle
local nProbSum     := 0            // Summing var., used for "random" angle
local nSmlHS_L     := 0            // Small trial hot spot semimaj axis len.
local nNoHitSmlHSL := 1            // P(0) for small hot spot
local nDiffSml     := 0            // Small hot spot diff from desired P(0)
local nLrgHS_L     := 0            // Large hot spot size
local nNoHitLrgHSL := 1            // P(0) for large hot spot
local nDiffLrg     := 0            // Large hot spot diff from desired P(0)
local nIntrHS_L    := 0            // Interpolated hot spot size
local nNoHitIntHSL := 1            // P(0) for interpolated hot spot
local nDiffInt     := 0            // Interpolated hspot diff from desired P(0)

```

Code File: EGPCScrn.Prg

```

local nGridCellArea := 0          // Area of one grid cell
local nNumSamples   := 0          // Number of samples required

* Misc vars.
local lOutFile      := .F.        // Use output file flag
local lWriteHeader   := .F.        // Write file header flag
local cStatus       := "OK"       // Status msg for output file data
local lDone         := .F.        // Loop flag
local lConverge      := .F.        // Convergence flag
local lAborted       := .F.        // Esc key abort
local lPastMaxIt     := .F.        // Exceeded nMAX_ITERS flag
local lOKProb        := .T.        // Solvable problem specs flag
local nKeyPress      := 0          // User key press
local getList        := {}         // Stops compiler warnings

private ntShape      := nShape     // private for F10 key function

if cGridType == NIL
  * Input error: no grid typ passed in.
  return (NIL)
endif

* Uppcase function argument.
cGridType := upper(cGridType)

* Get screen output file.
cOutFile := GetScnOutFile(@lOutFile, @lWriteHeader)

* Display screen title.
DispTitle(cGridType,"S",cOutFile, lOutFile)

* Get input data.
do while ! lDone
  @ 6, 2 say "Shape of the elliptical hot spot..." ;
  get m->ntShape pict cMAX_Shape ;
  valid ErrorUDF(ntShape <= 1.0 .and. ntShape >= 0.05, ;
    "Shape must be > 0.05 and < 1.0.",len(cMAX_Shape))
  @ 6,49 say "Shape = short axis/long axis."
  @ 8,49 say ' 99.0° for "random" angles.'

  if cGridType == "R"
    * Rectangular grid.
    @ 7, 2 say "Angle of orientation to grid.....:" ;
    get ntAngle pict cMAX_Angle ;
    valid ErrorUDF(ntAngle >= 0 .and. ntAngle <= 90.0 .or. ntAngle == 99, ;
      "Angle must be 0° to 90° or 99°=random.",len(cMAX_Angle))
    @ 7,col() say ""
    @ 7,49 say "Angle can be 0° to 90°. Use"
    @ 8, 2 say "Length of short side of rect. grid:" ;
    get ntGSize pict cMAX_GSize ;
    valid ErrorUDF(ntGSize >= 0, ;
      "Grid size must be > 0.0.",len(cMAX_GSize))
    @row(),col() say iif(m->cBasicUnit=="F","ft","m")
    @ 9, 2 say "Long side/short side ratio.....:" ;
    get ntRecRatio pict cMAX_RecRatio ;
    valid ErrorUDF(ntRecRatio > 1.0, ;
      "Ratio must be > 1.0.",len(cMAX_RecRatio))

  elseif cGridType == "S"
    * Square grid.
    @ 7, 2 say "Angle of orientation to grid.....:" ;
    get ntAngle pict cMAX_Angle ;
    valid ErrorUDF(ntAngle >= 0 .and. ntAngle <= 45.0 .or. ntAngle == 99, ;
      "Angle must be 0° to 45° or 99°=random.",len(cMAX_Angle))
    @ 7,col() say ""
    @ 7,49 say "Angle can be 0° to 45°. Use"

```

Code File: EGPCScr.Prg

```

@ 8, 2 say "Length of any side of square grid.:" ;
  get ntGSize pict cMAX_GSize ;
  valid ErrorUDF(ntGSize >= 0, ;
    "Grid size must be > 0.0.",len(cMAX_GSize))
  @row(),col() say iif(m->cBasicUnit=="F","ft","m")

elseif cGridType == "T"
  * Triangular grid.
  @ 7, 2 say "Angle of orientation to grid.....:" ;
  get ntAngle pict cMAX_Angle ;
  valid ErrorUDF(ntAngle >= 0 .and. ntAngle <= 30.0 .or. ntAngle == 99,;
    "Angle must be 0° to 30° or 99°=random.",len(cMAX_Angle))
  @ 7,col() say ""
  @ 7,49 say "Angle can be 0° to 30°. Use"
  @ 8, 2 say "Length of side of triangular grid.:" ;
  get ntGSize pict cMAX_GSize ;
  valid ErrorUDF(ntGSize >= 0, ;
    "Grid size must be > 0.0.",len(cMAX_GSize))
  @row(),col() say iif(m->cBasicUnit=="F","ft","m")
endif

@10, 2 say "Desired probability of hitting.....:" ;
  get ntDesirdProb pict cDESIRD_PROB ;
  valid ErrorUDF(ntDesirdProb >= 10.0 .and. ntDesirdProb<=99.9,;
    "Prob. must be 10% to 99.9%.",len(cDESIRD_PROB))
  @10,col() say "%"
  @10,49 say "Use 10% to 99.9%." ///, %Error <±.05% removed 03/31/94
  @11,49 say "    Leave area and sample cost"
  @12,49 say "    at 0, if cost not desired."
  @11, 2 say "Total area to sample.....:" ;
  get ntSampleArea pict cMAX_SampleArea
  @row(),col() say iif(m->cBasicUnit=="F","ft","m")
  @11,25 say "F10 = Acres" color(m->C_Help)
  @12, 2 say "Individual sample cost.....$:" ;
  get ntSampleCost pict cMAX_SampleCost
  @22, 0 say "|"
  @22,79 say "|"
  @22,1 to 22,78

@23,2 say "Enter = Continue   Esc = Abort" + space(44)  // erase msg

set key K_F10 to F10_Key()
read
set key K_F10 to
nKeyPress := lastkey()

* Abort, Write Data, etc...
do case
  case (nKeyPress == K_ESC)
    * Esc key pressed
    if YN_MsgBox("Abort current data entry session? Y/N")
      if lOutFile
        * Close out file.
        close alternate
      endif
      lDone := .T.
    endif
  case (nKeyPress == K_ENTER .or. nKeyPress == K_CTRL_W)
    * Enter key or Ctrl-W pressed. [Ctrl-W currently not documented.]
    * Save changes to static vars.
    scroll(13,1,21,78)
    nShape      := m->ntShape
    nAngle      := ntAngle
    nDesirdProb := ntDesirdProb
    nRecRatio   := ntRecRatio

```

Code File: EGPCScrn.Prg

```

if cGridType == "S"
  * Square grid.
  nGTyp := 1
  nLrgstAngle := 45          // For "random" angle
elseif cGridType == "R"
  * Rect. grid.
  nGTyp := 3
  nLrgstAngle := 90          // For "random" angle
elseif cGridType == "T"
  * Tri. grid.
  nGTyp := 2
  nLrgstAngle := 30          // For "random" angle
endif

* Cost related vars.
nSampleArea := ntSampleArea
nSampleCost := ntSampleCost

*-----| Find smallest area hit with given prob. |-----*
nDesirdProb := iff(nDesirdProb==100.0,99.95,nDesirdProb)
// Above left in code in case 100% again used as valid %.
nDesirdNoHit := 1.0 - (nDesirdProb*0.01)
* Below same as L/G <= 3.0 restriction.
nSmalHS_L := 0.1 * nGSize
* Below same as L/G of 0.10.
nLrgeHS_L := 3.0 * nGSize

scroll(23,1,23,78)
setcolor("W+/W*")          // Force blinking
@23,2 say "Calculating"
setcolor(m->C_Normal)
nCol := col()+1
@22,nCol say "┌"
@23,nCol say "│"
@24,nCol say "└"
@23,52 say "Esc = Stop Calculations..."
setcursor(SC_NONE)

* Keep calculating until error is less than nERR_CRITERIA.
nSICounter := 0              // Search iterations
nRACounter := 0              // Random angle iterations
@14,38 say "Search iterations.....: "
if nAngle == 99.0
  @15,38 say "Random angle iterations: "
endif
@17,38 say "Hot spot search is usually less"
@18,38 say "than 16 iterations."

lConverg := .F.              // .T. if search converges
lAborted := .F.              // .T. if Esc key abort
lPastMaxIt := .F.            // .T. if past max iterations
lOKProb := .T.               // .F. if problem can't be solved
do while ! lConverg .and. ! lAborted .and. ! lPastMaxIt .and. lOKProb
  nSICounter++               // Increment search counter
  @14,64 say NumTrim(nSICounter)+ "/" + ;
    ltrim(str(nMAX_ITERS)) + " maximum."

  * GET PROB. FOR HOT SPOT.
  if nAngle != 99.0
    * Non-random single angle case.
    nNoHitLrgHSL := ElipGrid(nLrgeHS_L, nShape, nAngle, nGSize, ;
      nGTyp, nRecRatio)
  else
    * Sum up multiple angle results, random case.
    nProbSum := 0.0
    for nCrntAngle = 0 to nLrgstAngle

```

Code File: EGPCScrn.Prg

```

        nProbNoHit := ElipGrid(nLrgHS_L, nShape, nCrntAngle, ;
            nGSize, nGTyp, nRecRatio)
        nProbSum := nProbSum + nProbNoHit
        nRACounter++
        @15,64 say ltrim(str(nRACounter))
        * Esc key abort, only used with random angles.
        if inkey() == K_ESC
            lAborted := .T.
            exit // exit for/next loop
        endif
    next nCrntAngle
    if lAborted
        loop
    endif

    * Calculate average.
    nNoHitLrgHSL := nProbSum/(nLrgstAngle+1)
endif

* Will hot spot size need to be larger than 3*G?
* If so, an L/G ratio > 3.0 would be required.
* If first search with largest hot spot can't get the
* desired prob. of hitting, no need to search farther.
if nSICounter == 1 .and. (nNoHitLrgHSL > nDesirdNoHit)
    * Quit searching.
    lOKProb := .F.
    loop
endif

* Check if we met error criteria with large hot spot.
nDiffLrg := abs(nDesirdNoHit - nNoHitLrgHSL)
if nDiffLrg < nERR_CRITERIA
    * Met error criteria with current large hot spot.
    * Exit search.
    nHS_L_Fnd := nLrgHS_L
    nPrbHtFnd := 1.0 - nNoHitLrgHSL
    lConverg := .T.
    loop
endif

* GET PROB. FOR SMALL HOT SPOT.
if nAngle != 99.0
    * Non-random single angle case.
    nNoHitSmlHSL := ElipGrid(nSmlHS_L, nShape, nAngle, nGSize, ;
        nGTyp, nRecRatio)
else
    * Sum up multiple angle results, random case.
    nProbSum := 0.0
    for nCrntAngle = 0 to nLrgstAngle
        nProbNoHit := ElipGrid(nSmlHS_L, nShape, nCrntAngle, ;
            nGSize, nGTyp, nRecRatio)
        nProbSum := nProbSum + nProbNoHit
        nRACounter++
        @15,64 say ltrim(str(nRACounter))
        * Esc key abort, only used with random angles.
        if inkey() == K_ESC
            lAborted := .T.
            exit // exit for/next loop
        endif
    next nCrntAngle
    if lAborted
        loop
    endif

    * Calculate average.
    nNoHitSmlHSL := nProbSum/(nLrgstAngle+1)

```

Code File: EGPCScrn.Prg

```

endif

* Check if we met error criteria with small hot spot.
nDiffSml := abs(nDesirdNoHit - nNoHitSmlHSL)
if nDiffSml < nERR_CRITERIA
    * Met error criteria with current small hot spot.
    * Exit search.
    nHS_L_Fnd := nSmlHS_L
    nPrbHitFnd := 1.0 - nNoHitSmlHSL
    lConverg := .T.
    loop
endif

* Get interpolated hot spot size.
* Bisection method, (Gerald and Wheatly 1989, 7)
nIntrHS_L := (nLrgHS_L + nSmlHS_L)/2

* GET PROB. FOR INTERPOLATED HOT SPOT SIZE.
if nAngle != 99.0
    * Non-random single angle case.
    nNoHitIntHSL := ElipGrid(nIntrHS_L, nShape, nAngle, nGSize, ;
        nGTyp, nRecRatio)
else
    * Sum up multiple angle results, random case.
    nProbSum := 0.0
    for nCrntAngle = 0 to nLrgstAngle
        nProbNoHit := ElipGrid(nIntrHS_L, nShape, nCrntAngle, ;
            nGSize, nGTyp, nRecRatio)
        nProbSum := nProbSum + nProbNoHit
        nRACounter++
        @15,64 say ltrim(str(nRACounter))
        * Esc key abort, only used with random angles.
        if inkey() == K_ESC
            lAborted := .T.
            exit // exit for/next loop
        endif
    next nCrntAngle
    if lAborted
        loop
    endif

    * Calculate average.
    nNoHitIntHSL := nProbSum/(nLrgstAngle+1)
endif

* Check if we met error criteria with interpolated hot spot.
nDiffInt := abs(nDesirdNoHit - nNoHitIntHSL)
if nDiffInt < nERR_CRITERIA
    * Met error criteria with current interpolated hot spot.
    * Exit search.
    nHS_L_Fnd := nIntrHS_L
    nPrbHitFnd := 1.0 - nNoHitIntHSL
    lConverg := .T.
    loop
endif

* Update large or small search hot spot sizes.
* This is a difference from linear interpolation and bisection
* methods. They look for sign changes of f(x). In root search
* case, f(x) values will be changing about 0.0.
* We look at whether our current f(x) for the interpolated grid
* is smaller than the desired value.
if nNoHitIntHSL < nDesirdNoHit
    nLrgHS_L := nIntrHS_L
else
    nSmlHS_L := nIntrHS_L

```

Code File: EGPCScr.Prg

```

endif

* Have we reached max iterations?
if nSICounter == nMAX_ITERS
    * Failed to converge.
    lPastMaxIt := .T.
    loop
endif
enddo

* Clean up calculating msg.
scroll(23,1,23,78)
a22,nCol say "-"
a24,nCol say "-"
tone(440,1)
setcursor(SC_NORMAL)
if cGridType == "R"
    * In case rec. grid, get long side.
    // nGLong := nHS_L_Fnd * nRecRatio
    //ERROR in 05/09/94 Beta - //
    nGLong := nGSize * nRecRatio
endif
*-----*

* Display results.
setcolor(m->C_Help)
if lPastMaxIt .or. lAborted
    * Failed to converge msg. or Esc key aborted.
    scroll(17,1,21,78)
    a17,1 to 21,78 double
    if lPastMaxIt
        a18,2 say " Failed to converge." color(m->C_Error)
    elseif lAborted
        a18,2 say " Calculations aborted..." color(m->C_Error)
    endif
    a19,2 say " Last interpolated hot spot estimate: " + ;
        ltrim(str(nIntrHS_L,12,4))
    a20,2 say " Last calculated prob. of hitting...: " + ;
        ltrim(str(100*(1-nNoHitIntHSL),12,4)) + "%"
elseif ! lOKProb
    * Problem specs. requires hot spot > 3 * grid size.
    scroll(17,1,21,78)
    a17,1 to 21,78 double
    a18,2 say " Data out of range." color(m->C_Error)
    a19,2 say " The input data require a hot spot that is out of " + ;
        "the search range of the"
    a20,2 say " program. Will only search for hot spot ≤ 3 * " + ;
        "length of grid size."
else
    * Found grid msg.
    scroll(14,1,21,78)
    a13,1 to 21,78 double
    * Move over to line up = signs.
    a14,2 say " Area of smallest hot spot hit = " + ;
        ltrim(str(nPI * nHS_L_Fnd^2 * nShape,10,1)) + ;
        iif(m->cBasicUnit=="F"," ft"," m")
    if cGridType == "R"
        * Rect. grid.
        a15,2 say " Given grid size, long side = " + ;
            ltrim(str(nGLong,10,3))
        arow(), col() say iif(m->cBasicUnit=="F"," ft"," m")
        a16,2 say " Given grid size, short side = " + ;
            ltrim(str(nGSize,10,3))
    else
        * Sq. or Tri. grid.
        a16,2 say " Grid size = " + ;

```

Code File: EGPCScrn.Prg

```

        ltrim(str(nGSize,10,3))
    endif
    @row(), col() say if(m->cBasicUnit=="F"," ft", " m")
    * Move over to line up = signs.
    @17,2 say " Given probability of hitting = " + ;
        ltrim(str(nDesirdProb,6,1)) + "%"

    @14,49 say "Search iterations.....: " + NumTrim(nSICounter)
    if nAngle == 99.0
        @15,49 say "Random angle iterations: " + NumTrim(nRACounter)
    endif
endif

if nSampleArea > 0 .and. lConverg
    * Required number of samples is approximate.
    * Based on (EPA. 1889. "Methods for Eval. the Attainment of Cleanup
    * Standards Volume 1: Soils and Solid Media", p. 9-7.
    * Calculate grid cell area.
    if cGridType == "T"
        * Triangular grid.
        * Grid cell area is now, 09/02/94, area of the rhombus formed
        * from 2 of the equilateral triangles.
        * A = height * base = sin(60°) * base * base = .87 * base^2.
        nGridCellArea := 0.866025404 * nGSize * nGSize
    elseif cGridType == "S"
        * Sq. grid.
        nGridCellArea := nGSize * nGSize
    else
        * Rec. grid.
        // nGridCellArea := nHS_L_Fnd * nGLong
        //ERROR in 05/09/94 Beta
        nGridCellArea := nGSize * nGLong
    endif
    * This formula is approx. See (EPA 1989, 9-7).
    * Ceiling function rounds number of samples up.
    nNumSamples := ceiling(nSampleArea/nGridCellArea)

    @18,2 say " Required number of samples          = " + ;
        ltrim(trans(nNumSamples,cMAX_Samples))
    @19,3 say ;
        "Required number of samples is approximate. " color "W/RB"
    @20,2 say " Total cost for above number of samples = $" + ;
        ltrim(trans(nNumSamples*nSampleCost,cMAX_TotalCost))
endif

if lOutFile .and. lConverg
    * Write to output file.
    * Pass lWriteHeader by reference, WriteData will update it.
    WriteData(@lWriteHeader,cVerDate,cOutFile,cGridType,nRecRatio, ;
        nHS_L_Fnd,nGSize,nShape,nAngle,nPrbHtFnd,nSampleArea, ;
        nNumSamples,nSampleCost)
elseif lOutFile .and. ((Aborted .or. ) lConverg)
    * Write data for abort or failed to converge.
    nHS_L_Fnd := nIntrHS_L
    nPrbHtFnd := 1.0 - nWohitIntHSL
    cStatus := if(lAborted," ABRT"," FAIL")
    WriteData(@lWriteHeader,cVerDate,cOutFile,cGridType,nRecRatio, ;
        nHS_L_Fnd,nGSize,nShape,nAngle,nPrbHtFnd,nSampleArea, ;
        nNumSamples,nSampleCost,cStatus)
endif
setcolor(m->c_Normal)
otherwise
    * Will loop back.
    lDone := .F.
endcase (lKey == )
enddo

```

Code File: EGPCScrn.Prg

```

return (NIL)
*** End of Func: GetSmallestArea()

*****
Function WriteData(lWriteHeader,cVerDate,cOutFile,cGridType,nRecRatio, ;
  nSemiMajor,nGSize,nShape,nAngle,nProbHit,nSampleArea, ;
  nNumSamples,nSampleCost, cStatus)
* Write header optionally based on lWriteHeader, then write a line of
* data to cOutFile.
* lWriteHeader is passed in by reference, then updated by WriteData().
default cStatus to "OK"

set console off
set alternate on
if lWriteHeader
  ?? "Output from ORNL/GJ ELIPGRID-PC Program Version: " + cVerDate
  ? "File name.: " + cOutFile
  ? "Created on: " + dtoc(date())
  ? "Input file: From screen"
  ?
  ? "Grid Type      Semi-major Axis   Gridspace   Shape" + ;
  ? "  Angle  Prob. Hitting Area   Samples   Cost/      Total"
  ? "          in Rel. Units (L/G) in " + ;
  ?   iif(m->cBasicUnit=="F","Feet ", "Meters") + "      " + ;
  ?     deg      1.0-P(0)      " + ;
  ?   iif(m->cBasicUnit=="F","ft", "m" ) + ;
  ?     (rounded up) Sample   Sample Cost "
  lWriteHeader := .F.
endif

* Write a line of data.
if cGridType == "S"
  ? "Square      " + space(8)
elseif cGridType == "R"
  ? "Rectangular, " + trans(nRecRatio,cMAX_RecRatio) + "/1 "
elseif cGridType == "T"
  ? "Triangular  " + space(8)
endif

* Write data fields.
?? trans(nSemiMajor/nGSize,cMAX_LtoG) + space(8) + ;
trans(nGSize,cMAX_GSize) + space(6) + ;
trans(nShape,cMAX_Shape) + space(3) + ;
iif(nAngle==99,"Random", " "+trans(nAngle,cMAX_Angle+"")) + ;
space(4) + ;
trans(nProbHit,cMAX_ProbHit)

if cStatus != "OK"
  * Write "ABRT" or "FAIL".
  ?? cStatus
elseif nSampleArea > 0
  * Write cost related info.
  ?? trans(nSampleArea,cMAX_SampleArea) + " " + ;
trans(nNumSamples,cMAX_Samples) + " " + ;
trans(nSampleCost,cMAX_SampleCost) + " " + ;
trans(nNumSamples * nSampleCost,cMAX_TotalCost)
endif

set alternate off
set console on
return (NIL)
*** End of Func: WriteData()

*** End of File: EGPCScrn.Prg

```

APPENDIX F

EGGRAPH SOURCE CODE

APPENDIX F
EGGRAPH SOURCE CODE

The first page of this appendix contains sample make and link files for EGGRAPH.
The remaining pages contain all the main code and subroutines in one file.

Sample Make and Link Files for EGGRAPH

```
// File....: EGGraph.rmk
// Purpose.: Make file for EGGraph program, G.Exe.
// Compiler: Clipper 5.2
// Author...: Jim Davidson
// Started..: 05/05/94
// Last Mod: 08/26/94
// Compiler Switches below:
// /A = Automatic declaration of publics/privates as memvars.
// /B = Include debugging info., delete this switch for final exe.
// /N = No automatic main proc., must be used for file-wide var declarations.
// /Q = Quiet, suppress line number display.
// /W = Warn of ambiguous var references.
// /V = Treat all ambiguous var references as dynamic vars, not as fields.
```

```
"e:\EGGraph.OBJ": "C:\CLIPPER2\EDITOR\EGGraph\EGGraph.PRG"
    e:\clipper C:\CLIPPER2\EDITOR\EGGraph\EGGraph /A/W/Q/V/W /Oe:\ /Te:\ /le:\
```

```
"e:\G.EXE": "e:\EGGraph.OBJ"
    e:\blinker @C:\CLIPPER2\EDITOR\EGGraph\EGGraph.LNK
```

```
# File....: EGGraph.Lnk
# Purpose.: Blinker response file for EGGraph Program with Flipper Libraries
# Compiler: Clipper 5.2
# Author...: Jim Davidson
# Started..: 05/05/94
# Last Mod: 08/26/94
blinker incremental off
blinker message noblink
# Below is obj source file
file e:\EGGraph
output e:\g
lib e:\clipper
lib e:\ct
lib e:\extend
lib e:\terminal
#lib e:\dbfntx
#lib e:\cld
search e:\flip5
search e:\clip50
```

Code File: EGGraph.Prg

```
//=====
// Program...: EGGraph.Prg, G.exe.
// Purpose...: Simple graphics demo program for ELIPGRID-PC.
// Version...: 1.0
// Author...: Jim Davidson
// Started...: 04/28/94
// Last Mod.: 09/06/94
//
// Files....: EGGraph.prg           This file only.
//
// Notes....: Compiler = Clipper 5.2d
//            Linker   = Blinker 3.0
// Modifications:
// 09/06/94 Added "UseDefaultF := .F." for error msg "No data values found"
//            to force file selection box to pop-up.
//=====
// Version Info
#define VER_DATE "09/06/94"
#define in() inkey(0)

// Include files
#include "Inkey.Ch"           // key definitions
#include "Colors.Ch"         // Color definitions
#include "Directry.ch"       // File info definitions
#include "Box.Ch"            // Box drawing constants

// Generic defines (may not all be used)
#define BELL1 chr(7)         // Error Bell
#define BELL2 chr(7) + chr(7) // Printing done bell
#define CR_LF chr(13) + chr(10)
#define SC_NORMAL 1          // Normal cursor (underline)
#define LINESIZE 80          // Buffer line size

// User-defined commands
#define DEFAULT <TheParam> TO <DefaultVal> => ;
    IF (<TheParam> == NIL); <TheParam>:=<DefaultVal>; ENDIF

* Currently graph will use no more than last nMAX_G_POINTS available points.
#define nMAX_G_POINTS 50    // Max graph points

* Below for colors
static C_Normal := C_WHT_BLU // Normal screen colors
static C_HighLight := C_CYN_BLU // Current subdir color
static C_Help := C_WHT_MAG // Help screens
static C_Error := C_WHT_RED // Error screens

* HP LJet IV (111?) commands.
#define HP_RESET chr(K_ESC) + "E"
* Below defined on HP LJ IV PCL Typeface List printout from test menu.
* Font is line printer internal font 48 for HP LJ IV.
#define HP_LINE_PRINTER_148 chr(K_ESC)+"(11U"+chr(K_ESC)+"(s0p16.67h8.5v0s0b0T"

*=====| Main Module |=====
Function Main()
* Main module of program
** Define main()'s local vars.

local lDone := .F. // Main() loop done flag
local cInFile := "Graph.Dat" // Default input file
local cInPath := "" // Input file path
local nHandle := 0 // File handle
local nKeyPress := 0 // Inkey() value of a keypress
local nTR := 5 // Top screen row
local cGrphData := ""
local cCurLine := ""
local nCurLine := 0
```

Code File: EGGraph.Prg

```

local nCurPoint      := 0
local nNumLines      := 0
local nNumPoints     := 0
local nLineNum       := 0
local nDataLines     := 0
local nFrstPoint     := 0
local nXMin          := -1          // Graph X-axis min value, -1 = auto min
local nYMin          := -1          // Graph Y-axis min value, -1 = auto min
local nXInc          := -1          // Graph X-axis inc, -1 = auto inc
local nXMax          := -1          // Graph X-axis max value, -1 = auto max
local nMaxGValue      := 0          // Graph x of max data value
local nLMargin       := 32
local nCh            := 0
local lBdFile        := .F.
local lUseDefaultF    := .T.        // Use default file flag
local cDOSCmdLine    := ""         // DOS command line params
local cDOSChar1      := ""
local anCostVals     := {}
local anProbVals     := {}
local acText         := {}

local GetList        := {}         // Define to stop compiler warnings

local HP_HorzSize    := 640        // See HP_Units() info in Flipper
local HP_VertSize    := 480        // manual, p. 7-10.
local HP_LeftX       := 231
local HP_LeftY       := 4454
local HP_RightX      := 7800
local HP_RightY      := 10130
local HP_HorzOrg     := 1150
local HP_VertOrg     := -500
local HP_PerCent     := .80
* Make static below so will not have to open fonts all the time.
* (If decide to call repetively.)
* Font1 and 2 are positive font handles from font_open().
static Font1        := -1          // Font1 used below is "RMN7_25.@"
static Font2        := -1          // Font2 used below is "SS2_22.@"
static Font3        := -1          // Font3 used below is "RMNB_15.@"

cls

* Get DOS command line parameters.
cDOSCmdLine := upper(dosparam())
cDOSChar1   := left(alltrim(cDOSCmdLine),1)

if "/" $ cDOSCmdLine .or. "?" $ cDOSCmdLine .or. cDOSChar1 == "H"
  * Help param. passed.
  ParamHelp(VER_DATE)
  quit
endif

If "." $ cDOSCmdLine
  * Get file name.
  cInFile := ltrim(cDOSCmdLine)
  if ! file(cInFile)
    Err_MsgBox(3,"E","ELIPGRID-PC Graph Program", ;
      "Error: File not found.", ;
      "File.: " + cInFile)
    lUseDefaultF := .F.
  endif
endif
else
  if "/" $ cDOSCmdLine .or. cDOSChar1 == "F" .or. ! file(cInFile)
    * Use file selector.
    lUseDefaultF := .F.
  endif
  if "/" $ cDOSCmdLine

```

Code File: EGGraph.Prg

```

* Monochrome param. passed.
* Black on white for LCD screens.
C_Normal      := C_BLK_WHT      // Normal screen colors
C_HighLight   := C_WHT_BLK      // Current subdir color
C_Help        := C_WHT_BLK      // Help screens
C_Error       := C_WHT_BLK      // Error screens
endif
endif

* Get drive and path program starts in.
cInPath := alltrim(diskname() + ":" + "\" + curdir())

set escape on
set scoreboard off
set bell off
set confirm on
set wrap on

do while ! lDone
    setcolor(C_Normal)          // Reset in case looping back
    cls

    * Get file name if not using default, Graph.dat.
    if ! lUseDefault
        cInPath := padr(cInPath,64)

        @0,0 to 4,79 double
        @01, 2 say "Program....: EGGraph.Exe"
        @01,61 say "Version: " + VER_DATE
        @02, 2 say "Purpose....: Simple demo of a graph program for ELIPGRID-PC."
        @03, 2 say "Note.....: File selector only lists *.DAT files."
        @03,63 say "Esc key to Exit" color(C_Help)

        * === Get subdir for input file === *
        nTR := 6
        @nTR,01 to nTR+4,67
        @nTR+1, 2 say " Enter drive and path of ELIPGRID-PC graph data file." ;
            color(C_Help)
        @nTR+3, 2 say " Do NOT enter file name." color(C_Help)

        * Get desired drive and path
        keyboard chr(K_END)
        @nTR+2, 3 get cInPath picture "@!"
        read
        cInPath := alltrim(cInPath)
        set key K_CTRL_F1 to

        if lastkey() == K_ESC
            lDone := .T.
            loop
        endif

        * Check for subdir
        if !SubDir(cInPath)
            Err_MsgBox(nTR+5,"E","Subdir path: " + alltrim(cInPath), ;
                "Could not be found! Try again.", ;
                "Be sure you did not enter the file name.")
            loop
        endif

        * Put a "\" on end if not there already.
        cInPath := cInPath + iif(right(cInPath,1)=="\" , "", "\")

        *==== Get Input File =====*
        * Pop up Get file box
        scroll(nTR+5,0,nTR+19,79)

```

Code File: EGGraph.Prg

```

cInFile := GetFileBox(nTR+5,5,nTR+15,NIL, cInPath + "*.DAT")

if lastkey() == K_ESC .or. empty(cInFile)
  loop
endif
endif

* Put a "\" on end if not there already.
cInPath := cInPath + iif(right(cInPath,1)=="\"","", "\")
scroll(nTR,0,24,79)
@nTR+1, 2 say "Input file.....: " + cInPath + cInFile

* Open the file
nHandle := fopen(cInPath + cInFile)
if (ferror() != 0)
  Err_MsgBox(10, "E", "Error opening: " + cInPath + cInFile)
  fclose(nHandle)
  loop
endif

* Get the data.
nLineNum := 0
nDataLines := 0
lBadFile := .F.
do while freadln(nHandle, @cCurLine, LINESIZE) .and. inkey() != K_ESC
  * Check first line to verify correct file format.
  if nLineNum == 0
    if ! (left(cCurLine,22) == "# Data starts on line:")
      Err_MsgBox(10, "E", ;
        "Error.....: Incorrect file format in line 1.,";
        "First 22 chars are: " + left(cCurLine,22), ;
        "Should read.....: # Data starts on line:")
      lBadFile := .T.
      exit
    endif
  endif

  nLineNum++
  @nTR+2,2 say "Processing line.: " + str(nLineNum,7)
  cCurLine := ltrim(cCurLine)
  if left(cCurLine,1) == "#"
    * Comment line, skip over or get data.
    if nLineNum >= 6 .and. nLineNum <= 11
      aadd(acText, substr(cCurLine,3))
    endif
  else
    * Line with a data point, add to data arrays.
    aadd(anCostVals, val(cCurLine))
    aadd(anProbVals, val(substr(cCurLine,at(" ",cCurLine))))
    nNumPoints++
  endif
enddo
fclose(nHandle)

* Loop back if bad file format.
if lBadFile
  if lUseDefaultF
    lDone := .T.
  endif
endif
loop
endif

* Loop back if no data points found.
if nNumPoints < 1
  Err_MsgBox(10, "E", "Error: No data values found.", ;
    "File.: " + cInPath + cInFile)

```

Code File: EGGraph.Prg

```

    (UseDefaultF := .F.           // Force file selection box to pop-up.
    loop
endif

* Use last nMAX_G_POINTS for graph.
if nNumPoints <= nMAX_G_POINTS
    nFrstPoint := 1
else
    nFrstPoint := nNumPoints - nMAX_G_POINTS + 1
endif

* Data buffer bytes = Number of points * 2 values/point * 8 bytes/point.
flip_init((nNumPoints-nFrstPoint+1) * 2 * 8)

* Below sets display of error msgs on.
set_sayerrrr(1)
* Open font files.
if Font1 < 0
    * Only open if not already opened.
    Font1 := font_open("RMN7_25.@" )
    Font2 := font_open("SS2_22.@" )
    Font3 := font_open("RMN8_15.@" )
endif
* Set data buffer to 2 cols.
initdata(2)
* Below sets the way Y col 1 will be displayed, 5 = point graph type.
set_type(1,5)
* Set Y col 1 to point pattern, 9 = filled box.
set_style(1,9)

* Get data into data buffer.
for nCurPoint = nFrstPoint to nNumPoints
    store_data(anCostVals[nCurPoint]/1000, 100*anProbVals[nCurPoint])
next i
* Max graph value.
nMaxGValue := anCostVals[nNumPoints]/1000

* Set the graph colors, Color macros in "Colors.Ch"
*      Graph   Grid   Graph Scale   Graph   Text
*      frame,  lines  scale, titles,  title,  legend  Inside
*      grf_colors(DLGREEN, DLYCAN, DLYELLOW, DLYELLOW, DLWHITE, DBLUE)
*
* Set graph title font.
set_g_font(Font1)

* Set graph label font.
set_l_font(Font2)

* Set axis tick labels.
set_s_font(Font3)

* Set x axis grid on, pattern = 0, line
set_grid(0,1,0)
* Set y axis grid on, pattern = 0, line
set_grid(1,1,0)
g_label("PROBABILITY OF HIT \. COST")
x_label("COST, $K")
y_label("PROBABILITY OF HIT, %")

* Display the graph.
plot()

* Below sets new current font to BIOS type 2.
font_new(2)
font_color(2, DLWHITE,DBLACK)
say_text(atx(70,2),aty(97,2), "Displays up to")

```

Code File: EGGraph.Prg

```

say_text(atx(70,2),aty(100,2), "last 30-50 points.")
set cursor on
say_text(atx(2,2),aty(97,2), "Press a key to exit...")
say_text(atx(2,2),aty(100,2), "F2 prints to HP LaserJet III/IV.")
inkey(0)
* Return to text mode.
textmode()
setcolor(C_Normal)
cls

*** Shall we PRINT the graph? ***
if lastkey() == K_F2 .and. AlertBox(7,(" Yes "," No, abort print "), ;
  "Are you attached to an On Line", "HP LaserJet III or IV printer?") == 1
  cls

  * Shall we print data file text info?
  nCh := AlertBox(7, ;
    ("Yes, print both", "No, just print the graph", "Abort print"), ;
    "Do you want to print BOTH graph and text comments?")

  if nCh == 1 .or. nCh == 2
    * Graph related info.
    cls
    * Shall we scale the X-axis?
    MenuBox(0,1,2,78)
    SayCenter(1, ;
      "Enter Graph Related Information. -1 = Automatic Scaling. " + ;
      "Esc = Abort")
    @ 5,2 say "Enter = lowest cost for graph, -1 = auto minimum:" ;
    get nXMin pict "99999" ;
    valid ErrorUDF(nXMin < nMaxGValue, ;
      "Min X must be < " + (trim(str(nMaxGValue,8,1)) + " K",5)
    @row(),col()+1 say "$K"

    * If scaling X-axis, what max shall we use?
    @ 7,2 say "Enter = max cost for graph, -1 = auto maximum...:" ;
    get nXMax pict "99999" when nXMin > 0 ;
    valid ErrorUDF(nXMax > 0, "X maximum must be > 0 K",5)
    @row(),col()+1 say "$K"

    * If scaling X-axis, what increment shall we use?
    @ 9,2 say "Enter Cost increment in $K, -1 = auto increment.:" ;
    get nXInc pict "999" when nXMin > 0 ;
    valid ErrorUDF(nXInc < nMaxGValue, "X increment must be < " + ;
      (trim(str(nMaxGValue,8,1)) + " K",3)

    * Shall we scale the Y-axis?
    @11,2 say "Enter = lowest probability for graph, -1 = auto scaling:" ;
    get nYMin pict "99"
    read
    if lastkey() == K_ESC
      * Abort print.
      lDone := .T.
      loop
    endif
  endif

  if nCh == 1
    * Print text comments.
    cls
    MenuBox(0,1,2,78)
    SayCenter(1,"Enter Text Related Information. Esc = Abort")
    @ 5,2 say "Enter left margin for text, 0-60:" ;
    get nLMargin pict "99" ;
    valid ErrorUDF(nLMargin >= 0 .and. nLMargin <= 60, ;
      "Must be >= 0 and <= 60.", 2)

```

Code File: EGGraph.Prg

```

read
if lastkey() == K_ESC
    * Abort print.
    lDone := .T.
    loop
endif

cls
MenuBox(7,10,9,70)
@8,12 say "Sending data to printer. Please wait..."
set print on
set device to print
set console off
set margin to nLMargin
* Send line printer font command.
?? HP_LINE_PRINTER_148

* Establish first text line.
@40,0 say " "
?
? "Input File: " + cInPath + cInFile
? "Print Date: " + dtoc(date())
? "Print Time: " + ltrim(ampm(time()))
for nCurLine = 1 to len(acText)
    ? acText[nCurLine]
next nCurLine

set margin to
set print off
set device to screen
set console on
elseif nCh == 2
    * Just print graph.
    cls
    MenuBox(7,10,9,70)
    @8,12 say "Sending data to printer. Please wait..."
else
    * Abort print.
    lDone := .T.
    loop
endif

* Now do actual graphics printing.
*
* Set the default orientation to 1 = portrait.
* Flipper Manual p. 7-9.
* This command seems needed even though HP III defaults to portrait.
hp_setup(1)

* Opens an HP LJet III for re-direction of Flipper commands.
* Flipper Manual p. 7-5.
hp_open('LJIII')

* Set up the units to plot from screen to print device.
* Values used were determined by trial-and-error after talking
* with a Flipper rep.
* Flipper Manual p. 7-10.
* Scale the graph.
HP_RghtX *= HP_PerCent
HP_RghtY *= (HP_PerCent+.1)
hp_units( (HP_HorzSize, HP_VertSize, HP_LeftX, HP_LeftY, ;
           HP_RghtX, HP_RghtY, HP_HorzOrg, HP_VertOrg) )

* Below sets the way Y col 1 will be displayed, 5 = point graph type.
set_type(1,5)

```

Code File: EGGraph.Prg

```

* Set Y col 1 to point pattern, 9 = filled box.
set_style(1,9)

* Set graph title font.
set_g_font(Font1)

* Set graph label font.
set_l_font(Font2)

* Set manual x-axis scaling (or leave as auto).
if nXMin >= 0
    set_xman(1)                // Turn on manual scaling
    set_xmin(nXMin)            // Set X-axis min
    if nXInc > 0
        * When setting X increment, the X-axis max must also be set.
        * See Flipper Manual, p. 3-36.
        set_xinc(nXInc)
        if nXMax <= 0
            * This will put X-axis max just beyond largest X value.
            nXMax := nXMin + Ceiling((nMaxGValue - nXMin)/c) * nXInc
        end
        set_xmax(nXMax)
    endif
endif

* Set manual Y-axis scaling (or leave as auto).
if nYMin >= 0
    set_yman(1)                // Turn on manual scaling
    set_ymin(nYMin)            // Set Y-axis min
endif

* Set x axis grid on, pattern = 0, line
set_grid(0,1,0)
* Set y axis grid on, pattern = 0, line
set_grid(1,1,0)
g_label("PROBABILITY OF HIT vs COST")
x_label("COST, $K")
y_label("PROBABILITY OF HIT, %")

* Below forces a half-tone graph background
grf_colors(,,,,, DBLUE)

* Plot lines, points, legends, etc.
plot()
/// LJ3_Eject() removed, 05/05/94, HP_RESET below does eject.
HP_Close()

* Reset the internal LJ font, if needed, also eject page.
set print on
set console off
?? HP_RESET
set print off
set console on
endif
* Return to text mode.
textmode()
* Will exit program after enddo.
exit
enddo

close all
set color to
cls
? "Type EXIT if you want to return to ELIPGRID-PC program..."
* return to DOS
return (0)                // Return 0 to DOS ErrorLabel
*** End of Func: Main()

```

Code File: EGGraph.Prg

```

*****
Function AlertBox(nTR, acOptions, nLin1, nLin2, nLin3)
* Substitute for alert() function. Alert() does not obey color settings.
* AlertBox obeys current color setting. Alert() is hard to read on LCD screens.
* Lines 2 and 3 are optional.
* Returns: Esc = 0, else number of array element of acOptions chosen.
local cTmpScn      := ""
local lDispMsg     := .T.
local nMaxLineWdth := 0
local nPrmptWdth   := 0
local nWidth       := 0
local cOrgClr      := ""
local nLC          := 0
local nBR          := 0
local nRC          := 0
local nLines       := 0
local nCurRow     := 0
local nCurCol     := 0
local nNumOps      := len(acOptions)
local nCurOp      := 1
local nOpCol       := 0
local nRtnVal      := 0

* Set box color.
cOrgClr := setcolor(C_Error)

* Get current cursor pos.
nCurRow := row()
nCurCol := col()

if (valtype(nLin3) == "C")
    * 3 lines to display
    nBR := nTR + 4 + 3           // 4 lines for misc. + 3 msg lines
    nMaxLineWdth := max( max(len(nLin1), len(nLin2)), len(nLin3) )
    nLines := 3
elseif (valtype(nLin2) == "C")
    * 2 lines to display
    nBR := nTR + 4 + 2           // 4 lines for misc. + 2 msg lines
    nMaxLineWdth := max(len(nLin1), len(nLin2))
    nLines := 2
elseif (valtype(nLin1) == "C")
    * 1 line to display
    nBR := nTR + 4 + 1
    nMaxLineWdth := len(nLin1)
    nLines := 1
else
    * Incorrect params. passed
    lDispMsg := .F.
endif

* Display message.
if (lDispMsg)
    * Get total width of the prompts plus inner spacing.
    for nCurOp = 1 to nNumOps
        nPrmptWdth := nPrmptWdth + len(acOptions[nCurOp])
    next nCurOp
    nPrmptWdth := nPrmptWdth + 3 * (nNumOps-1)

    * Determine overall width of box.
    nMaxLineWdth := max(nMaxLineWdth, nPrmptWdth)
    nWidth := 4 + nMaxLineWdth
    nLC := (79 - nWidth)/2           // center
    nRC := nLC + nWidth - 1
    cTmpScn := savescreen(nTR, nLC, nBR+1, nRC+1)
    MenuBox(nTR, nLC, nBR, nRC)
    if (nLines >= 1)

```

Code File: EGGraph.Prg

```

    @nTR+2, nLC + 2 say nLin1
endif
if (nLines >= 2)
    @nTR+3, nLC + 2 say nLin2
endif
if (nLines == 3)
    @nTR+4, nLC + 2 say nLin3
endif

* Display and get desired menu option.
* ... - 1 to nOptions

    nOpCol := nOpCol + len(acOptions[nCurOp-1]) + 3
endif
    @nBR-1, nOpCol prompt acOptions[nCurOp]
next nCurOp
tone(440,0.3)
menu to nRtnVal

restscreen(nTR, nLC, nBR+1, nRC+1, cTmpScn)
else
    @0,0
    @0,0 say " AlertBox() error: Check parameters.  Press a key to return... "
    inkey(0)
endif (lDispMsg )
setcolor( cOrgClr )
@nCurRow, nCurCol say ""
return (nRtnVal)
*** End of Func: AlertBox()

*****
Function Ceiling(nNum)
* Returns the next integer >= nNum on the number line.
* Examples: Ceiling(3.01) == 4
*           Ceiling(3)    == 3
*           Ceiling(-3.99) == -3
*           Ceiling(-3)   == -3
local nRtnVal := 0
if nNum % int(nNum) == 0
    * Already an integer.
    nRtnVal := nNum
else
    * If pos, trunc and add 1, else just truncate.
    nRtnVal := iif(nNum >= 0, int(nNum) + 1, int(nNum))
endif
return (nRtnVal)
*** End of Func: Ceiling()

*****
Function Err_MsgBox(nTR, cType, nLin1, nLin2, nLin3)
* Generic error or msg box. Defaults to error box.
* Displays up to 3 lines + Press key msg and waits for keypress.
* Returns: NIL
local cTmpScn      := ""
local lDispMsg     := .T.
local nMaxLineWdth := 0
local nWidth       := 0
local cOrgClr      := ""
local nLC          := 0
local nBR          := 0
local nRC          := 0
local nLines       := 0
local nCurRow      := 0
local nCurCol      := 0

```

Code File: EGGraph.Prg

```

default cType to "E"                // Default to error box

* Set box color.
if upper(cType) == "E"
    cOrgClr := setcolor(C_Error)
else
    cOrgClr := setcolor(C_Help)
endif

* Get current cursor pos.
nCurRow := row()
nCurCol := col()

if (valtype(nLin3) == "C")
    * 3 lines to display
    nBR := nTR + 4 + 3                // 4 lines for misc. + 3 msg lines
    nMaxLineWdth := max( max(len(nLin1), len(nLin2)), len(nLin3) )
    nLines := 3
elseif (valtype(nLin2) == "C")
    * 2 lines to display
    nBR := nTR + 4 + 2                // 4 lines for misc. + 2 msg lines
    nMaxLineWdth := max(len(nLin1), len(nLin2))
    nLines := 2
elseif (valtype(nLin1) == "C")
    * 1 line to display
    nBR := nTR + 4 + 1
    nMaxLineWdth := len(nLin1)
    nLines := 1
else
    * Incorrect params. passed
    lDispMsg := .F.
endif

* Display message.
if (lDispMsg)
    nMaxLineWdth := max( nMaxLineWdth, len("Press a key to continue...") )
    nWidth := 4 + nMaxLineWdth        // 2 lines/blanks + largest line
    nLC := (79 - nWidth)/2            // center
    nRC := nLC + nWidth - 1
    cTmpScn := savescreen(nTR, nLC, nBR+1, nRC+1)
    MenuBox(nTR, nLC, nBR, nRC)
    if (nLines >= 1)
        @nTR+2, nLC + 2 say nLin1
    endif
    if (nLines >= 2)
        @nTR+3, nLC + 2 say nLin2
    endif
    if (nLines == 3)
        @nTR+4, nLC + 2 say nLin3
    endif
    @nBR-1, nLC + 2 say "Press a key to continue..."
    tone(440,1)
    inkey(0)
    restscreen(nTR, nLC, nBR+1, nRC+1, cTmpScn)
else
    @0,0
    @0,0 say " Err_MsgBox() error: Check parameters.  Press a key to return... "
    inkey(0)
endif (lDispMsg)
setcolor( cOrgClr )
@nCurRow, nCurCol say ""
return (NIL)
*** End of Func: Err_MsgBox()

*****
Function ErrorUDF(lPassTest, cErrorMsg, nFldLen)

```

Code File: EGGraph.Prg

```

* Generic error routine for @ say/get valid clauses.
* lPassTest: Logic flag for--pass test?
* cErrorMsg: Message to display
* nFldLen...: Length of get field--as picture specifies for numeric.
* Returns...: .F. if lPassTest == .F., else just returns .T.
local CurGetName := readvar()           // Name of current get variable
local nTR        := row() + 1           // Current row + 1 for error box
local nBR        := nTR + 3
local nLC        := col() - nFldLen     // Current col is end of get field
local nRC        := nLC + len(cErrorMsg) + 1
local cTmpScr    := savescreen(nTR, nLC, nBR, nRC)
local cCurClr   := setcolor(C_Error)
local lRtnVal    := .F.

if ! lPassTest
    * Invalid input failed valid test, display error box.
    scroll(nTR, nLC, nBR, nRC)
    @nTR,nLC to nBR,nRC
    @nTR+1, nLC+1 say cErrorMsg
    @nTR+2, nLC+1 say "Press a key..."
    tone(440,1)
    inkey(0)
    restscreen(nTR, nLC, nBR, nRC, cTmpScr)
    lRtnVal := .F.
else
    lRtnVal := .T.
endif
setcolor(cCurClr)
return (lRtnVal)
*** End of Func: ErrorUDF()

*****
Function ExtrctPath( PathFileN )
* Extract path from pathfilename
* Example: ExtrctPath("D:\file.ext") ==> "D:\"
* Based on Environ.prg fuction FilePath() supplied by Nantucket
local BkSlshPos, Path
BkSlshPos := rat("\", PathFileN)
if ( BkSlshPos == 0 )
    Path := ""
else
    Path := substr(PathFileN, 1, BkSlshPos)
endif
return ( Path )
*** End of Func: ExtrctPath()

*****
Function FReadLn(Handle, Line, LineSize)
* From Ref(Clipper), 4/88, p.9
local More, Where, StrLen, TabSpaces, CharPos
local Buffer := space(LineSize)
local NumRead := fread(Handle, @Buffer, LineSize)
if (NumRead != 0)
    Where := at(chr(13) + chr(10), Buffer)
    * Did we find a new line
    if (Where != 0)
        * yes, so return
        Line := substr(Buffer, 1, Where)
        * Reposition to just after new line
        fseek(Handle, -NumRead + Where + 1, 1)
    else
        * no, so return all we read
        Line := substr(Buffer,1,NumRead)
    endif
    More := .T.
else

```

Code File: EGGraph.Prg

```

More := .F.
endif
return (More)

*** End of Func: FReadLn()

*****
Function GetFileBox(nTR, nLC, nBR, nRC, cDirSpec, lDispBox, cColor, nInitFile)
* Pop-up file selector, all params. are optional
* Parameter defaults:
*   nTR top row ==> to 0
*   nLC left col ==> to 0
*   nBR bot row ==> to maxrow
*   nRC right col ==> to nLC + 38
*   cDirSpec ==> "*.*)"
*   lDispBox ==> .T.
*   cColorVar ==> "W+/n,n/W"
*   nInitFile ==> 1
* Returns:
*   if Enter key ==> File name
*   if Esc key ==> NIL
*   if error ==> NIL

local cOrgClr := ""
local cFileName := NIL
local cTmpScrn := ""
local i // Scratch
local aDrctry := {} // Array of dir info
local acFileNames := {} // Array of file names
local nFileChoice := 0

* If any param. not passed, below assigns defaults as needed.
default nTR to 0
default nLC to 0
default nBR to maxrow()
default nRC to nLC + 38
default cDirSpec to "*.*)"
default lDispBox to .T.
default cColor to (C_Help)
default nInitFile to 1

cTmpScrn := savescreen(nTR,nLC,nBR+1,nRC+1) // + 1 for shadow

if (!SubDir(cDirSpec))
  Err_MsgBox(15,"E","No .DAT files found in current subdir.")
  return (NIL)
endif

cOrgClr := setcolor(cColor)
scroll(nTR,nLC,nBR,nRC)
if (lDispBox)
  MenuBox(nTR,nLC,nBR,nRC)
endif
@nTR,nLC+2 say " Choose Input File... "

aDrctry := directory(cDirSpec)
* Sort array according to file name.
asort(aDrctry,,, ([FrstName,NextName| FrstName[F_NAME] < NextName[F_NAME]))

* Fill an array with file info to display.
acFileNames := {}
for i = 1 to len(aDrctry)
  add(acFileNames, ;
    pedl(aDrctry[i,F_NAME],13) + ;
    pedl(numtrim(aDrctry[i,F_SIZE]),8) + ;
    pedl(dtoc(aDrctry[i,F_DATE]), 9) + ;

```

Code File: EGGraph.Prg

```

        padl(substr(aDrctry[i,F_TIME],1,5),6) )
next i

* Display files and get choice.
nFileChoice := achoice(nTR+1,nLC+1,nBR-1,nRC-1, acFileNames,,,nInitFile)
if (nFileChoice != 0)
    * Is 0 if Esc key exit
    cFileName := aDrctry[nFileChoice,F_NAME]
    nInitFile := nFileChoice
endif
setcolor(cOrgClr)
restscreen(nTR,nLC,nBR+1,nRC+1,cTmpScn)
return (cFileName)
*** End of Func: GetFileBox()

*****
Function MenuBox(nTR,nLC,nBR,nRC, cSides, lShadow)
* Draw box sides for a menu.
* cSides defaults to double top, single sides.
* cSides could be defined constants from from Box.Ch.
* lShadow defaults to .T.
local cOrgColor := setcolor()
default cSides to B_DOUBLE_SINGLE
default lShadow to .T.
if lShadow
    set color to
    scroll(nTR+1,nLC+1,nBR+1,nRC+1)
    setcolor(cOrgColor)
    scroll(nTR,nLC,nBR,nRC)
endif
dispbox(nTR,nLC,nBR,nRC, cSides)
return (NIL)
*** End of Func: MenuBox()

*****
Function NumTrim( Num )
* Returns Num in str form trimmed
local NumStr := alltrim(str(Num))
return (NumStr)
*** End of Func: NumTrim()

*****
Function ParamHelp(cVerDate)
* Parameter help screen.
set color to W+/M
cls
?? repli("-",80)
?? "ORNL/GJ ELIPGRID-PC Simple Graph Program, Version: " + cVerDate
? "Usage: G [Filename.Ext] | [/H | ?] | [/F [/M]]"
?
? '      G      = Use "Graph.Dat" as default input file.'
? '      G Tr2.A = Use file "Tr2.A" as default input file.'
? "      G /F    = Use file selector for input files."
? "      G /H    = Help on command line parameters, this screen."
? "      G /M    = Monochrome non-graphics screens."
?
? " Example: G /F/M"
? "           Use file selector and monochrome screens."
?
? "Quick parameter options:"
? "      G H      = Help"
? "      G F      = File selector"
? repli("-",80)
return (NIL)
*** End of Func: ParamHelp()

```

Code File: EGGraph.Prg

```

*****
Function SayCenter(nRow, cMsg)
* Displays cMsg on centered nRow.
local nCol := (80-len(cMsg))/2
aRow,nCol say cMsg
return (NIL)
*** End of Func: SayCenter()

*****
Function Subdir( TestSubdir )
* Returns .T. if TestSubdir exists, .F. otherwise
* The directory() command will return an empty array
* if the TestSubdir does not exist.
local RtnVal := .F.
local aDirctry := {}

TestSubdir := alltrim(TestSubdir)
aDirctry := directory(TestSubdir, "D") // D to include all subdirs
if len(aDirctry) > 0
    RtnVal := .T.
endif
return (RtnVal)
*** End of Func: Subdir()

*** End of File: EGGraph.Prg

```

INTERNAL DISTRIBUTION

- | | |
|------------------------|----------------------------------|
| 1. B. A. Berven | 32. G. H. Stevens |
| 2. B. Coleman | 33. J. E. Wilson |
| 3 - 22. J. R. Davidson | 34. Central Research Library |
| 23. P. V. Egidi | 35 - 36. Laboratory Records |
| 24. D. K. Halford | 37. Laboratory Records - RC |
| 25 - 30. C. A. Little | 38. ORNL Patent Section |
| 31. P. T. Owen | 39. ORNL Technical Library, Y-12 |

EXTERNAL DISTRIBUTION

40. Jim Berger, 1299 Bethel Valley Road, Oak Ridge, TN 37830
41. James A. Bowers, Westinghouse Savannah River Company, PO Box 616, Aiken, SC 29802
42. C. C. Britton, Mesa State College, PO Box 2647, Grand Junction, CO 81502
43. Rich Engelder, RUST Geotech, Inc., PO Box 14000, Grand Junction, CO 81502
44. Richard O. Gilbert, Pacific Northwest Laboratory, ISB 1 Building, PO Box 999, Richland, WA 99352
45. Carl Gogolak, U.S. Department of Energy, Environmental Measurements Lab., 376 Hudson Street, New York, NY 10014-3621
46. Tim LeGore, Westinghouse Hanford Company, PO Box 1970, Richland, WA 99352
47. Don Mackenzie, USDOE/HQ, EM-442, Quince Orchard Bldg., Washington, DC 20585-0002
48. David E. Mathes, USDOE, Office of ER, EM-451 (GTN) Room D-427, Washington, DC 20545
49. T. J. Novotny, Mesa State College, PO Box 2647, Grand Junction, CO 81502
50. Donald A. Singer, U.S. Geological Survey, 345 Middlefield Road, Menlo Park, CA 94025
51. Andrew Wallo, III, USDOE, Air, Water & Radiation Division, EH-232, 1000 Independence Avenue, SW, Washington, DC 20585
52. Office of Assistant Manager, Energy Research and Development, Oak Ridge Operations Office, P.O. Box 2001, Oak Ridge, TN 37831-8600
- 53 - 54. Office of Scientific and Technical Information, U.S. Department of Energy, P.O. Box 62, Oak Ridge, TN 37831

END DATE
12-1-94