This program, written in FORTRAN, demonstrates the application of the statistical approach to the modeling of diffraction. It predicts the spatial distribution of diffracted energy as it passes through a single infinite slit, and arrives at an observation screen some distance away. The following describes the variables used throughout the program. This particular version is for the ***splitting rays approach****.*

* NY is an array that divides the observation screen into strips. Each element in this array serves as a
* "bin" which keeps track of the number of bundles reaching that strip.

* LAMDA is the wavelength of the entering monochromatic radiation.

* W is the width of the rectangular slit.

* Z is the distance from the aperture to the observation screen.

* R is the change in vertical location that the entering energy bundle undergoes between its point of entry
* and its arrival to the observation screen.

* DIST is the total distance traveled by energy bundle from aperture before being intercepted by the
* observation screen.

* YO is the coordinate of the bottom edge of the rectangular aperture.

* Y is the coordinate of the entering energy bundle in the plane containing the aperture.

* YSCREEN is the coordinate of the energy bundle when it strikes the observation plane. Note that the
* YSCREEN coordinate is such that YSCREEN=0 occurs at the aperture center.

* YSCMIN, YSCMAX determine the minimum and maximum values of Y for which the number of
* energy bundles striking at the observation screen will be recorded.

* H is the number of strips into which the observation screen will be divided.

* INCREM is the width of each of the strips on the observation screen.

* NUMRAYS is the number of energy bundles to be directed from the aperture to the observation screen.

* DEL1,DEL2 are the distances from the point of entry to the two aperture edges.

* RANMAR calls a random number from the subroutine rmarin(ij, kl).

* XVAL(ERFX) is the subroutine which determines the values of x when erf(x) is known, and returns this
* value to the main program.

* ERFX1,ERFX2 are the arguments sent to XVAL for the determination of x where erf(x) is known.

* SD1,SD2 are the standard deviations for the diffraction angles calculated for each of the two edges.

* K is the wavenumber, given by 2*PI/LAMDA.

* PHI1,PHI2 are the two diffraction angles due to each aperture side.

* YDIV is the variable used to determine the "bin" into which to store energy bundles incident to the
* observation screen.

PROGRAM DIFFRACT1

```
C$    NOEXTENSIONS NOWARNINGS

*     Initialize arrays, Note that array NY is of size H, where H is the number of strips into which
*     observation screen is divided.

      DIMENSION NY(1000)

*     Initialize variables used in program

      DOUBLE PRECISION LAMDA, W,Z,Y,R1,YO,DEL1,DEL2,RANMAR,ARG2
     *PI,ERFX1,ERFX2,SD1,SD2,K,PHI1,PHI2,PHIDIFF1,YSCREEN1,ARG1,
     *YSCMIN,YSCMAX,INCREM,NY,DIST1,PHIDIFF2,R2,DIST2,YSCREEN2

      REAL XVAL

      INTEGER N,YDIV,H,NUMRAYS

      DATA LAMDA,W,Z,PI/100.0,60.0,60.25,3.141593/

      H=1000
      YO=-W/2
      YSCMAX=80.0
      YSCMIN=-80.0
      INCREM=(YSCMAX-YSCMIN)/H
      NUMRAYS=2000000
      K=(2*PI)/LAMDA

*     Open file that will store output from execution of code.
      OPEN(15,FILE='sr100',STATUS='OLD')

*     Initialize the values in the matrix containing the number of energy bundles arriving at screen, which is
*     divided into H strips.

      DO 5 I=1,H
      NY(I)=0
5     CONTINUE

*     Assign seeds for random number generator.
      I=1
      J=3
      CALL RMARIN(I,J)

*     Begin Monte-Carlo solution.  This loop causes NUMRAYS rays to be fired in randomly and uniformly
*     from aperture.

      DO 10 J=1,NUMRAYS

*     Calculate point of entry of the current energy bundle.

       Y=YO + RANMAR()*W

*     Calculate distance from point of entry of the current energy bundle and the two aperture edges.

       DEL1=(W/2)+ABS(Y)
```

```
      DEL2=(W/2)-ABS(Y)

*     Calculate the standard deviation for distribution of diffraction angles.

      SD1=ATAN(1.0/(2.0*DEL1*K))
      SD2=ATAN(1.0/(2.0*DEL2*K))

*     Calculate the angle of diffraction caused by each of the aperture edges.  The subprogram XVAL must
*     be called in order to determine the value of x, where erf(x) is known.

      ERFX1= 2.0*RANMAR()-1.0
      ARG1=XVAL(ERFX1)
      PHI1=ATAN(ARG1*SQRT(2.0)*SD1)

*     Calculate the angle of diffraction caused by the other aperture edge.

      ERFX2= 2.0*RANMAR()-1.0
      ARG2=XVAL(ERFX2)
      PHI2=ATAN(ARG2*SQRT(2.0)*SD2)

*     Calculate the angle of diffraction caused by each edge.

       PHIDIFF1=PHI1
       PHIDIFF2=PHI2

*     Calculate the change in vertical location of the entering energy bundle, R.

      R1=TAN(PHIDIFF1)*Z
      R2=TAN(PHIDIFF2)*Z

*     Calculate the total distance traveled by the energy bundle before reaching the screen, DIST.

      DIST1=SQRT(Z**2+ R1**2)
      DIST2=SQRT(Z**2+ R2**2)

*     Calculate the coordinate at which each energy bundle arrives at the observation screen.

      YSCREEN1=Y+R1
      YSCREEN2=Y+R2

*     Increment counter for correct strip on observation screen for ray1.

     IF (ABS(YSCREEN1).LE.YSCMAX) THEN
      IF (YSCREEN1.LT.0) THEN
         YDIV =INT(YSCREEN1/INCREM)-1
         YDIV=((H/2)+1)+YDIV
       ELSE IF (YSCREEN1.GE.0) THEN
         YDIV =INT(YSCREEN1/INCREM)+1
         YDIV=YDIV+(H/2)
       END IF
     NY(YDIV)=NY(YDIV)+1
     END IF
```

```
*     Increment counter for correct strip on observation screen for ray2.

     IF (ABS(YSCREEN2).LE.YSCMAX) THEN
       IF (YSCREEN2.LT.0) THEN
         YDIV =INT(YSCREEN2/INCREM)-1
         YDIV=((H/2)+1)+YDIV
       ELSE IF (YSCREEN2.GE.0) THEN
         YDIV =INT(YSCREEN2/INCREM)+1
         YDIV=YDIV+(H/2)
       END IF
     NY(YDIV)=NY(YDIV)+1
     END IF

10     CONTINUE
*      After all rays have been fired, store the results in an output file.
       DO 40 I=1,H
       WRITE(15,*)NY(I),(YSCMIN+INCREM*(I-1)),'   -',(YSCMIN+INCREM*(I))
40     CONTINUE

       END


*    Subroutine that determines x, when erf(x) is known

      FUNCTION XVAL(ERFX)

*   This subprogram is used to solve for x, knowing erf(x), which is sent from the main program.  A
*    truncated infinite series is used to approximate the error function, which can be solved for x
*    using the bisection method.

*     Initialize all variables

      INTEGER NUM,I,NEG
      DOUBLEPRECISION SD,A,B,TOL,P,PO,PI,CHK,ERFX,FP,FA,FB
      REAL XVAL
      DATA TOL/0.0001/, PI/3.141593/, NUM/1000/

      IF (ERFX.LT.0) THEN
        NEG=1
        ERFX=ABS(ERFX)
      ELSE IF (ERFX.GT.0) THEN
        NEG=2
      END IF

     IF (ERFX.LE.0.992869) THEN

*     Initialize endpoints (x=a, lower endpoint; x=b, upper endpoint)
      A=-1
      B=2.0

*     Initialize counter
      I=0

*     Begin loop to continue for a maximum of NUM iterations
      DO WHILE (I.LT.NUM)
        CHK= ABS((B-A)/2)
```

```
*    Determine midpoint
       P= A + ((B-A)/2)

*    Solve error function for current midpoint and endpoints using a truncated (15 terms) infinite series
 *   approximating erf(x).

      FP=(2/SQRT(PI))*(P-(P**3)/(3*1)+(P**5)/(5*2)-(P**7)/(7*6)
     *+(P**9)/(9*24)-(P**11)/(11*120)+(P**13)/(13*720)-
     *(P**15)/(15*5040) +(P**17)/(17*40320)-(P**19)/(19*362880)
     *+(P**21)/(21*3628800)-(P**23)/(23*39916800))-ERFX

      FA=(2/SQRT(PI))*(A-(A**3)/(3*1)+(A**5)/(5*2)-(A**7)/(7*6)
     *+(A**9)/(9*24)-(A**11)/(11*120)+(A**13)/(13*720)-
     *(A**15)/(15*5040) + (A**17)/(17*40320) -(A**19)/(19*362880)
     *+(A**21)/(21*3628800)-(A**23)/(23*39916800))-ERFX

      FB=(2/SQRT(PI))*(B-(B**3)/(3*1)+(B**5)/(5*2)-(B**7)/(7*6)
     *+(B**9)/(9*24)-(B**11)/(11*120)+(B**13)/(13*720)-
     *(B**15)/(15*5040) + (B**17)/(17*40320) -(B**19)/(19*362880)
     *+(B**21)/(21*3628800)-(B**23)/(23*39916800))-ERFX

        IF (FP.EQ.0) THEN
          XVAL=P
          GO TO 10
        ELSE IF (CHK.LT.TOL) THEN
          XVAL=(A+B)/2
          GO TO 10
        ELSE IF ((FA*FP).LT.0) THEN
          B=P
        ELSE IF ((FB*FP).LT.0) THEN
          A=P
        END IF

      END DO

       ELSE
        XVAL=2
        END IF

10    IF (NEG.EQ.1) THEN
       XVAL=-XVAL
       ERFX=-ERFX
      END IF

      END

*    Subroutine that generates Random Numbers

       SUBROUTINE RMARIN(ij, kl)

C This is the initialization routine for the random number generator ranmar(). NOTE: The seed variables
C can have values between:
C    0 <= IJ <= 31328
C    0 <= KL <= 30081

C The random number sequences created by these two seeds are of sufficient length to complete an entire
```

```
C calculation with. For example, if sveral different groups are working on different parts of the same
C calculation, each group could be assigned its own IJ seed. This would leave each group with 30000
C choices for the second seed. That is to say, this random number generator can create 900 million different
C subsequences -- with each subsequence having a length of approximately 10^30.
C Use IJ = 1802 & KL = 9373 to test the random number generator.  The subroutine ranmar should be used
C to generate 20000 random numbers.  Then display the next six random numbers generated multiplied by
C 4096*4096.  If the random number generator is working properly, the random numbers should be:
C          6533892.0  14220222.0  7275067.0
C          6172232.0  8354498.0   10633180.0

C   implicit real*8 (a-h, o-z)
      real*8 u(97), c, cd, cm, s, t
      integer ii, i, j, ij, jj, k, kl, l, m, i97, j97
      logical test
      common /raset1/ u, c, cd, cm, i97, j97, test
      test = .false.
c
      if( IJ .lt. 0  .or.  IJ .gt. 31328  .or.
     S    KL .lt. 0  .or.  KL .gt. 30081 ) then
        write (*, *) ' The first random number seed must have a'
        write (*, *) ' value between 0 and 31328.'
        write (*, *)
        write (*, *) ' The second seed must have a value between 0'
        write (*, *) ' and 30081.'
        write (*, *) 'Stopping...'
        stop
      endif
c
      i = mod(IJ/177, 177) + 2
      j = mod(IJ    , 177) + 2
      k = mod(KL/169, 178) + 1
      l = mod(kl,     169)
c
      do 2 ii = 1, 97
        s = 0.0
        t = 0.5
        do 3 jj = 1, 24
          m = mod(mod(i*j, 179)*k, 179)
          i = j
          j = k
          k = m
          l = mod(53*l+1, 169)
          if (mod(l*m, 64) .ge. 32) then
            s = s + t
          endif
          t = 0.5 * t
 3      continue
        u(ii) = s
 2      continue
c
      c = 362436.0 / 16777216.0
      cd = 7654321.0 / 16777216.0
      cm = 16777213.0 /16777216.0
c
      i97 = 97
      j97 = 33
```

```
c
      test = .true.
c
      return
      end
      real*8 function ranmar()
c
c This is the random number generator proposed by George Marsaglia
c in Florida State University Report: FSU-SCRI-87-50
c
c     implicit real*8 (a-h, o-z)
      real*8 u(97), uni, c, cd, cm
      integer i97, j97
      logical test
      common /raset1/ u, c, cd, cm, i97, j97, test
c
      if(.not.test) then
        write (*, *)
        write (*, *) 'ranmar error #1: must call the'
        write (*, *) 'initialization routine rmarin before'
        write (*, *) 'calling ranmar.'
        write (*, *) 'Stopping...'
        stop
      endif
c
      uni = u(i97) - u(j97)
      if( uni .lt. 0.0 ) uni = uni + 1.0
      u(i97) = uni
      i97 = i97 - 1
      if(i97 .eq. 0) i97 = 97
      j97 = j97 - 1
      if(j97 .eq. 0) j97 = 97
      c = c - cd
      if( c .lt. 0.0 ) c = c + cm
      uni = uni - c
      if( uni .lt. 0.0 ) uni = uni + 1.0
c
      ranmar = uni
c
      return
      end
```

This portion of a program, written in FORTRAN, demonstrates the application of the statistical approach to the modeling of diffraction. It predicts the spatial distribution of diffracted energy as it passes through a single infinite slit, and arrives at an observation screen some distance away. The subroutines called can be found with the first code (for the splitting rays approach) in Appendix A. This particular version is for the ***summing angles approach****.*

```
PROGRAM DIFFRACT2
C$    NOEXTENSIONS NOWARNINGS
*     Initialize variables used in program

*     Initialize arrays
      DIMENSION NY(1000)

*     Note that array NY is of size H, where H is the number of strips into which observation screen is
*     divided.

      DOUBLE PRECISION LAMDA, W,Z,Y,R1,YO,DEL1,DEL2,RANMAR,
     *PI,ERFX1,ERFX2,SD1,SD2,K,PHI1,PHI2,PHIDIFF,YSCREEN,ARG1,
     *YSCMIN,YSCMAX,INCREM,NY,DIST,R,ARG2
      REAL XVAL
      INTEGER N,YDIV,H,NUMRAYS

      DATA LAMDA,W,Z,PI/100.0,60.0,60.25,3.141593/
      H=1000
      YO=-W/2
      YSCMAX=80.0
      YSCMIN=-80.0
      INCREM=(YSCMAX-YSCMIN)/H
      NUMRAYS=2000000
      K=(2*PI)/LAMDA

*     Open file that will store output from execution of code.
      OPEN(15,FILE='sumr100',STATUS='OLD')

*     Initialize the values in the matrix containing the number of energy bundles arriving at screen, which is
*     divided into H strips.

      DO 5 I=1,H
      NY(I)=0
5     CONTINUE

*     Assign seeds for random number generator.
      I=1
      J=3
      CALL RMARIN(I,J)

*     Begin Monte-Carlo solution.  This loop causes NUMRAYS rays to be fired in randomly and uniformly
*     from aperture.

      DO 10 J=1,NUMRAYS

*     Calculate point of entry of the current energy bundle.
       Y=YO + RANMAR()*W

*     Calculate distance from point of entry of the current energy bundle and the two aperture edges.
       DEL1=(W/2)+ABS(Y)
```

```
      DEL2=(W/2)-ABS(Y)

*     Calculate the standard deviation for distribution of diffraction angles.

      SD1=ATAN(1.0/(2.0*DEL1*K))
      SD2=ATAN(1.0/(2.0*DEL2*K))

*     Calculate the angle of diffraction caused by each of the aperture edges.  The subprogram XVAL must
*     be called in order to determine the value of x, where erf(x) is known.

      ERFX1= 2.0*RANMAR()-1.0
      ARG1=XVAL(ERFX1)
      PHI1=ATAN(ARG1*SQRT(2.0)*SD1)

*     Calculate the angle of diffraction caused by the other aperture edge.

      ERFX2= 2.0*RANMAR()-1.0
      ARG2=XVAL(ERFX2)
      PHI2=ATAN(ARG2*SQRT(2.0)*SD2)

*     Calculate the angle of diffraction due to the presence of both sides of the aperture by summing the two
*     angles.

      PHIDIFF=PHI1+PHI2

*     Calculate the change in vertical location of the entering energy bundle, R.

      R=TAN(PHIDIFF)*Z

*     Calculate the total distance traveled by the energy bundle before reaching the screen, DIST.

      DIST=SQRT(Z**2+ R**2)

*     Calculate the coordinate at which each energy bundle arrives at the observation screen.

      YSCREEN=Y+R

*     Increment counter for correct strip on observation screen.

     IF (ABS(YSCREEN).LE.YSCMAX) THEN
      IF (YSCREEN.LT.0) THEN
      YDIV =INT(YSCREEN/INCREM)-1
       YDIV=((H/2)+1)+YDIV
      ELSE IF (YSCREEN.GE.0) THEN
      YDIV =INT(YSCREEN/INCREM)+1
       YDIV=YDIV+(H/2)
      END IF
      NY(YDIV)=NY(YDIV)+1
      END IF

10    CONTINUE
*     After all rays have been fired, store the results in an output file.
       DO 40 I=1,H
       WRITE(15,*)NY(I),(YSCMIN+INCREM*(I-1)),'   -',(YSCMIN+INCREM*(I))
40    CONTINUE
     END
```

This portion of a program, written in FORTRAN, demonstrates the application of the statistical approach to the modeling of diffraction. It predicts the spatial distribution of diffracted energy as it passes through a single infinite slit, and arrives at an observation screen some distance away. The subroutines called can be found with the first code (for splitting rays approach) in Appendix A.. This particular version is for the ***closest edge effect*** *approach.*

```
PROGRAM DIFFRACT3
C$    NOEXTENSIONS NOWARNINGS


*     Initialize arrays, note that array NY is of size H, where H is the number of strips into which
*     observation screen is divided.
      DIMENSION NY(1000)

*     Initialize variables used in program

      DOUBLE PRECISION LAMDA, W,Z,Y,R1,YO,DEL1,DEL2,RANMAR,ARG2
     *PI,ERFX1,ERFX2,SD1,SD2,K,PHI1,PHI2,PHIDIFF1,YSCREEN1,ARG1,
     *YSCMIN,YSCMAX,INCREM,NY,DIST1,PHIDIFF2,R2,DIST2,YSCREEN2
      REAL XVAL
      INTEGER N,YDIV,H,NUMRAYS

      DATA LAMDA,W,Z,PI/100.0,60.0,60.25,3.141593/
      H=1000
      YO=-W/2
      YSCMAX=80.0
      YSCMIN=-80.0
      INCREM=(YSCMAX-YSCMIN)/H
      NUMRAYS=2000000
      K=(2*PI)/LAMDA

*     Open file that will store output from execution of code.
      OPEN(15,FILE='ns100',STATUS='OLD')

*     Initialize the values in the matrix containing the number of energy bundles arriving at screen, which is
*     divided into H strips.

      DO 5 I=1,H
      NY(I)=0
5     CONTINUE

*     Assign seeds for random number generator.
      I=1
      J=3
      CALL RMARIN(I,J)

*     Begin Monte-Carlo solution.  This loop causes NUMRAYS rays to be fired in randomly and uniformly
*     from aperture.

      DO 10 J=1,NUMRAYS

*     Calculate point of entry of the current energy bundle.
       Y=YO + RANMAR()*W

*     Calculate distance from point of entry of the current energy bundle and the two aperture edges.
       DEL1=(W/2)+ABS(Y)
```

```
      DEL2=(W/2)-ABS(Y)

      IF (DEL1.LT.DEL2)THEN
      SD1=ATAN(1.0/(2.0*DEL1*K))
      ERFX1= 2.0*RANMAR()-1.0
      ARG1=XVAL(ERFX1)
      PHI1=ATAN(ARG1*SQRT(2.0)*SD1)
      PHIDIFF1=PHI1
      R1=TAN(PHIDIFF1)*Z
      DIST1=SQRT(Z**2+ R1**2)
      YSCREEN1=Y+R1
     IF (ABS(YSCREEN1).LE.YSCMAX) THEN
      IF (YSCREEN1.LT.0) THEN
      YDIV =INT(YSCREEN1/INCREM)-1
       YDIV=((H/2)+1)+YDIV
      ELSE IF (YSCREEN1.GE.0) THEN
      YDIV =INT(YSCREEN1/INCREM)+1
       YDIV=YDIV+(H/2)
      END IF
      NY(YDIV)=NY(YDIV)+1
      END IF


      ELSE
     SD2=ATAN(1.0/(2.0*DEL2*K))
      ERFX2= 2.0*RANMAR()-1.0
      ARG2=XVAL(ERFX2)
      PHI2=ATAN(ARG2*SQRT(2.0)*SD2)
       PHIDIFF2=PHI2
      R2=TAN(PHIDIFF2)*Z
      DIST2=SQRT(Z**2+ R2**2)
      YSCREEN2=Y+R2

     IF (ABS(YSCREEN2).LE.YSCMAX) THEN
      IF (YSCREEN2.LT.0) THEN
      YDIV =INT(YSCREEN2/INCREM)-1
       YDIV=((H/2)+1)+YDIV
      ELSE IF (YSCREEN2.GE.0) THEN
      YDIV =INT(YSCREEN2/INCREM)+1
       YDIV=YDIV+(H/2)
      END IF
      NY(YDIV)=NY(YDIV)+1
      END IF
      END IF

10    CONTINUE

*     After all rays have been fired, store the results in an output file.

     DO 40 I=1,H
     WRITE(15,*)NY(I),(YSCMIN+INCREM*(I-1)),'   -',(YSCMIN+INCREM*(I))
40    CONTINUE
     END
```

This portion of a program, written in FORTRAN, demonstrates the application of the statistical approach to the modeling of diffraction. It predicts the spatial distribution of diffracted energy as it passes through a single infinite slit, and arrives at an observation screen some distance away. The subroutines called can be found with the first code (for splitting rays approach) in Appendix A. This particular version is for the ***furthest edge effect*** *approach.*

```
PROGRAM DIFFRACT4
C$    NOEXTENSIONS NOWARNINGS


*     Initialize arrays, note that array NY is of size H, where H is the number of strips into which
*     observation screen is divided.
      DIMENSION NY(1000)

*     Initialize variables used in program

      DOUBLE PRECISION LAMDA, W,Z,Y,R1,YO,DEL1,DEL2,RANMAR,
     *PI,ERFX1,ERFX2,SD1,SD2,K,PHI1,PHI2,PHIDIFF1,YSCREEN1,ARG1,ARG2
     *YSCMIN,YSCMAX,INCREM,NY,DIST1,PHIDIFF2,R2,DIST2,YSCREEN2
      REAL XVAL
      INTEGER N,YDIV,H,NUMRAYS

      DATA LAMDA,W,Z,PI/100.0,60.0,60.25,3.141593/
      H=1000
      YO=-W/2
      YSCMAX=80.0
      YSCMIN=-80.0
      INCREM=(YSCMAX-YSCMIN)/H
      NUMRAYS=2000000
      K=(2*PI)/LAMDA

*     Open file that will store output from execution of code.
      OPEN(15,FILE='fs100',STATUS='OLD')

*     Initialize the values in the matrix containing the number of energy bundles arriving at screen, which is
*     divided into H strips.

      DO 5 I=1,H
      NY(I)=0
5     CONTINUE

*     Assign seeds for random number generator.
      I=1
      J=3
      CALL RMARIN(I,J)

*     Begin Monte-Carlo solution.  This loop causes NUMRAYS rays to be fired in randomly and uniformly
*     from aperture.

      DO 10 J=1,NUMRAYS

*     Calculate point of entry of the current energy bundle.
       Y=YO + RANMAR()*W

*     Calculate distance from point of entry of the current energy bundle and the two aperture edges.
       DEL1=(W/2)+ABS(Y)
```

```
      DEL2=(W/2)-ABS(Y)

      IF (DEL1.GT.DEL2)THEN
      SD1=ATAN(1.0/(2.0*DEL1*K))
      ERFX1= 2.0*RANMAR()-1.0
      ARG1=XVAL(ERFX1)
      PHI1=ATAN(ARG1*SQRT(2.0)*SD1)
      PHIDIFF1=PHI1
      R1=TAN(PHIDIFF1)*Z
      DIST1=SQRT(Z**2+ R1**2)
      YSCREEN1=Y+R1
     IF (ABS(YSCREEN1).LE.YSCMAX) THEN
      IF (YSCREEN1.LT.0) THEN
      YDIV =INT(YSCREEN1/INCREM)-1
       YDIV=((H/2)+1)+YDIV
      ELSE IF (YSCREEN1.GE.0) THEN
      YDIV =INT(YSCREEN1/INCREM)+1
       YDIV=YDIV+(H/2)
      END IF
      NY(YDIV)=NY(YDIV)+1
      END IF


      ELSE
     SD2=ATAN(1.0/(2.0*DEL2*K))
      ERFX2= 2.0*RANMAR()-1.0
      ARG2=XVAL(ERFX2)
      PHI2=ATAN(ARG2*SQRT(2.0)*SD2)
       PHIDIFF2=PHI2
      R2=TAN(PHIDIFF2)*Z
      DIST2=SQRT(Z**2+ R2**2)
      YSCREEN2=Y+R2

     IF (ABS(YSCREEN2).LE.YSCMAX) THEN
      IF (YSCREEN2.LT.0) THEN
      YDIV =INT(YSCREEN2/INCREM)-1
       YDIV=((H/2)+1)+YDIV
      ELSE IF (YSCREEN2.GE.0) THEN
      YDIV =INT(YSCREEN2/INCREM)+1
       YDIV=YDIV+(H/2)
      END IF
      NY(YDIV)=NY(YDIV)+1
      END IF
      END IF

10    CONTINUE

*     After all rays have been fired, store the results in an output file.

     DO 40 I=1,H
     WRITE(15,*)NY(I),(YSCMIN+INCREM*(I-1)),'   -',(YSCMIN+INCREM*(I))
40    CONTINUE
     END
```