M I E V   D O C U M E N T A T I O N

** NEWS ** Aug 1996: The 1979 NCAR Mie report, long out of print, has been converted to electronic form and considerably edited and brought up to date. Look for it in this directory in various forms: PostScript (.ps) and PDF (.pdf), mainly. The PDF version is very nice if you will take the time to get the free Adobe Acrobat Reader from the web site http://www.adobe.com/.

** NOTE ** The output variable SPIKE, having to do with the detection of resonances, is still under research and will undoubtedly change in the future. Presently, SPIKE only detects the broadest spikes, of width roughly 0.1 in size parameter; ultimately narrower spikes should also be detected, although the probability of hitting them is much smaller. SPIKE is mainly of use in avoiding spikes during numerical integration over a size distribution.

Author: Dr. Warren J. Wiscombe (wiscombe@climate.gsfc.nasa.gov)
NASA Goddard Space Flight Center
Code 913
Greenbelt, MD 20771

FTP availability: The entire package is available by anonymous ftp from Internet site climate.gsfc.nasa.gov in subdirectory pub/wiscombe. (ftp to 'climate', login as 'anonymous', give your e-mail address as password, then 'cd' to pub/wiscombe.)

The MIEV package contains the following files (besides the present one):

(1) MIEV0.f, the main subroutine which a user calls, plus ancillary subroutines

(1a) MIEV0noP.f: MIEV0.f with all the code relating to Legendre moments PMOM removed (smaller and requires less array storage); just created 12/89 and seems to be working fine but has not had the benefit of years of user testing like MIEVO.f; argument list same as MIEV0.f in order to allow swapping of this with MIEV0.f without changing calling program(s)

(2) ErrPack.f: a set of 4 error-handling routines needed by both MIEV0.f and MIEV0noP.f

(3) MVTstOld.f, the main program for running the 8 test cases at the end of Reference (1) below (the NCAR Mie report)

(4) MVTstOld.out, the output generated by MVTstOld.f

(5) MVTstNew.f, the main program for running an exhaustive set of 19 test cases.

(6) MVTstNew.out, the output generated by MVTstNew.f; usually Unix-compressed (.Z on end of file name)

(7) PMOMTest.f, a program to test the Legendre coefficients computed by MIEV0.f against those computed approximately by numerical quadrature of the phase matrix

Note that MIEV1, the Cray-customized version of MIEV0 described in Ref.(1) below, is omitted from this package. It is no longer supported, for reasons given in the new (Aug 96) version of Ref. (1) available in electronic form.

All subroutines and functions have some internal documentation in addition to that in this file. Also, all the declaration statements were standardized using the NAG Fortran Tools.

MIEV0 computes the following quantities involved in eletromagnetic scattering from a homogeneous sphere:

* scattering and extinction efficiencies;
* asymmetry factor;
* forward- and backscatter amplitude;
* scattering amplitudes vs. scattering angle for incident polarization parallel and perpendicular to the plane of scattering;
* coefficients in the Legendre polynomial expansions of either the unpolarized phase function or the polarized phase matrix;
* some quantities needed in polarized radiative transfer;
* information about whether or not a resonance has been hit.

NOTE -- MIEV0 differs from the original code published in Ref. (1) below in the following ways :

* computes Legendre moments, based on vast improvements to the formulas of Sekera (see Refs. 3-5) and correction of errors in the formulas of Ref. 3

* returns a measure of how nasty of a spike (resonance) you are sitting on (this is invaluable when integrating over size and you want to exclude unrepresentative points); this part of the program is a work-in-progress and is far from finished, but it may prove useful even in its present form

* allows real refractive indices less than unity

* adds a totally reflecting special case

* performs a self-test on the first call to the routine

* adds several new input and output variables, and makes all I/O through arguments of the subroutine

* allows complete freedom in specifying angles

* allows printing of all output variables at user option

* some variables names are more mnemonic

Also, major improvements have been made, based on modern ideas of documentation and program structure (e.g., Kernighan and Plauger, The Elements of Programming Style). Those interested in my thoughts in this area may find

PostScript documents in the anonymous ftp directory cited above, under pub/wiscombe/Writing_Programs.

REFERENCES

(1) Wiscombe, W., 1979: Mie Scattering Calculations--Advances in Technique And Fast, Vector-Speed Computer Codes, Ncar Tech Note TN-140+STR, National Center For Atmospheric Research, Boulder, Colorado (out of print but an updated electronic version available)

(2) Wiscombe, W., 1980: Improved Mie scattering algorithms, Appl. Opt. 19, 1505-1509

(3) Dave, J.V., 1970a: Coefficients of the Legendre and Fourier series for the scattering functions of spherical particles, Appl. Opt. 9, 1888-1896

(4) Dave, J.V., 1970b: Intensity and polarization of the radiation emerging from a plane-parallel atmosphere containing monodisperse aerosols, Appl. Opt. 9, 2673-84

(5) Van De Hulst, 1957, 1982: Light Scattering by Small Particles, Dover Press, New York.

(6) Bohren, C. and D. Huffman, 1983: Absorption and Scattering of Light by Small Particles, Wiley, New York. (has a Mie program in the back of the book)

I N P U T   V A R I A B L E S

(Even if an input variable is not needed for a particular application, make sure it has a legitimate value that can be written out and read in -- no indefinites, etc.)

XX	Mie size parameter (2 * pi * radius / wavelength)
CREFIN	Complex refractive index (imag part can be + or -, but internally a negative imaginary index is assumed). If imag part is - , scattering amplitudes as in Van de Hulst are returned; if imag part is + , complex conjugates of those scattering amplitudes are returned (the latter is the convention in physics). ** NOTE ** In the 'PERFECT' case, scattering amplitudes in the Van de Hulst (Ref. 6 above) convention will automatically be returned unless Im(CREFIN) is positive; otherwise, CREFIN plays no role.
PERFCT	TRUE, assume refractive index is infinite and use special case formulas for Mie coefficients 'a' and 'b' (see Kerker, M., The Scattering of Light and Other Electromagnetic Radiation, p. 90). This is sometimes called the 'totally reflecting', sometimes the 'perfectly conducting' case.

```
          ( see CREFIN for additional information )

MIMCUT    (positive) value below which imaginary refractive
          index is regarded as zero (computation proceeds
          faster for zero imaginary index)

ANYANG    TRUE, any angles whatsoever may be input through
          XMU.  FALSE, the angles are monotone increasing
          and mirror symmetric about 90 degrees (this option
          is advantageous because the scattering amplitudes
          S1,S2 for the angles between 90 and 180 degrees
          are evaluable from symmetry relations, and hence
          are obtained with little added computational cost.)

NUMANG    No. of angles at which scattering amplitudes
          S1,S2 are to be evaluated  ( set = 0 to skip
          calculation of S1,S2 ).  Make sure NUMANG does
          not exceed the parameter MAXANG in the program.

XMU(N)    Cosines of angles ( N = 1 TO NUMANG ) at which S1,S2
          are to be evaluated.  If ANYANG = FALSE, then

           (a) the angles must be monotone increasing and
               mirror symmetric about 90 degrees (if 90-A is
               an angle, then 90+A must be also)

           (b) if NUMANG is odd, 90 degrees must be among
               the angles

NMOM       Highest Legendre moment PMOM to calculate,
           numbering from zero ( NMOM = 0 prevents
           calculation of PMOM )

IPOLZN     POSITIVE, Compute Legendre moments PMOM for the
                     Mueller matrix elements determined by the
                     digits of IPOLZN, with 1 referring to M1,
                     2 to M2, 3 to S21, and 4 to D21 (Ref. 3).
                     E.g., if IPOLZN = 14 then only moments for
                     M1 and D21 will be returned.

           0,        Compute Legendre moments PMOM for the
                     unpolarized unnormalized phase function.

           NEGATIVE, Compute Legendre moments PMOM for the
                     Sekera phase quantities determined by the
                     digits of ABS(IPOLZN), with 1 referring to
                     R1, 2 to R2, 3 to R3, and 4 to R4 (REF. 4).
                     E.g., if IPOLZN = -14 then only moments for
                     R1 and R4 will be returned.

           ( NOT USED IF  NMOM = 0 )

MOMDIM     Determines first dimension of PMOM, which is dimensioned
           internally as PMOM( 0:MOMDIM, * ) (second dimension must
           be the larger of unity and the highest digit in
           IPOLZN; if not, serious errors will occur).
           Must be given a value, even if  NMOM = 0.  Minimum: 1.
```

```
  PRT(L)     Print flags (LOGICAL).  L = 1  prints  S1,S2, their
             squared absolute values, and degree of polarization,
             provided NUMANG is non-zero.   L = 2  prints all
             output variables other than  S1,S2.


O U T P U T   V A R I A B L E S
-------------------------------

  QEXT      (REAL) extinction efficiency factor  ( Ref. 2, Eq. 1A )

  QSCA      (REAL) scattering efficiency factor  ( Ref. 2, Eq. 1B )

  GQSC      (REAL) asymmetry factor times scattering efficiency
            ( Ref. 2, Eq. 1C )  ( allows calculation of radiation
            pressure efficiency factor  QPR = QEXT - GQSC )

  ======================================================================
  ==== NOTE --  S1, S2, SFORW, SBACK, TFORW, AND TBACK are calculated
  ====          internally for negative imaginary refractive index;
  ====          for positive imaginary index, their complex conjugates
  ====          are taken before they are returned, to correspond to
  ====          customary usage in some parts of physics ( in parti-
  ====          cular, in papers on CAM approximations to Mie theory ).
  ======================================================================

  S1(N),    (COMPLEX) Mie scattering amplitudes at angles specified
  S2(N)     by XMU(N) ( N=1 to NUMANG )  ( Ref. 2, Eqs. 1d-e ).

  SFORW     (COMPLEX) forward-scattering amplitude S1 at
            0 degrees.  ( S2(0 deg) = S1(0 deg) )

  SBACK     (COMPLEX) backscattering amplitude S1 at
            180 degrees.   ( S2(180 deg) = - S1(180 deg) )

  TFORW(I)  (COMPLEX) values of

                I=1:  T1 = ( S2 - (MU)*S1 ) / ( 1 - MU**2 )
                I=2:  T2 = ( S1 - (MU)*S2 ) / ( 1 - MU**2 )

            At angle theta = 0 ( MU = COS(theta) = 1 ), where the
            expressions on the right-hand side are indeterminate.
            ( these quantities are required for doing polarized
            radiative transfer (Ref. 4, Appendix). )

  TBACK(I)  (COMPLEX) values of  T1 (for I=1) or  T2 (for I=2) at
            angle  theta = 180 degrees ( MU = COS(theta) = - 1 ).

  SPIKE     (REAL) magnitude of the smallest denominator of
            either Mie coefficient (a-sub-n or b-sub-n),
            taken over all terms in the Mie series past
            N = size parameter XX.  Values of SPIKE below
            about 0.3 signify a ripple spike, since these
            spikes are produced by abnormally small denominators
            in the Mie coefficients (normal denominators are of
            order unity or higher).  Defaults to 1.0 when not
            on a spike.  Does not identify all resonances
            (we are still working on that).
```

```
PMOM(M,NP) (REAL) moments  M = 0 to NMOM  of unnormalized NP-th
           phase quantity PQ  ( moments with  M .GT. 2*NTRM  are
           zero, where  NTRM = no. terms in Mie series =
           XX + 4*XX**1/3 + 1 ) :

             PQ( MU, NP ) = sum( M=0 to infinity ) ( (2M+1)
                                 * PMOM( M,NP ) * P-sub-M( MU ) )

           WHERE  MU = COS( scattering angle )
                  P-sub-M = M-th Legendre polynomial

           and the definition of 'PQ' is as follows:

           IPOLZN.GT.0:  PQ(MU,1) = CABS( S1(MU) )**2
                         PQ(MU,2) = CABS( S2(MU) )**2
                         PQ(MU,3) = RE( S1(MU)*CONJG( S2(MU) ) )
                         PQ(MU,4) = - IM( S1(MU)*CONJG( S2(MU) ) )
                         ( called M1, M2, S21, D21 in literature )

           IPOLZN=0:  PQ(MU,1) = ( CABS(S1)**2 + CABS(S2)**2 ) / 2
                      ( the unnormalized phase function )

           IPOLZN.LT.0:  PQ(MU,1) = CABS( T1(MU) )**2
                         PQ(MU,2) = CABS( T2(MU) )**2
                         PQ(MU,3) = RE( T1(MU)*CONJG( T2(MU) ) )
                         PQ(MU,4) = - IM( T1(MU)*CONJG( T2(MU) ) )
                         ( called R1, R2, R3, R4 in literature )

           The sign of the 4th phase quantity is a source of
           confusion.  It flips if the complex conjugates of
           S1,S2  or  T1,T2  are used, as occurs when a
           refractive index with positive imaginary part is
           used (see discussion below).  The definition above
           is consistent with a negative imaginary part.

           See Ref. 5 for correct formulae for PMOM ( Eqs. 2-5
           of Ref. 3 contain typographical errors ).  Ref. 5 also
           contains numerous improvements to the Ref. 3 formulas.

           NOTE THAT OUR DEFINITION OF MOMENTS DIFFERS FROM REF. 3
           in that we divide out the factor (2M+1) and number the
           moments from zero instead of one.

           ** WARNING **  Make sure the second dimension of PMOM
           in the calling program is at least as large as the
           absolute value of IPOLZN.

           For small enough values of XX, or large enough values
           of M,  PMOM  will tend to underflow.  Thus, it is
           unwise to assume the values returned are non-zero and,
           for example, to divide some quantity by them.


 INTEGRATING OVER SIZES
 ----------------------

   The normalized phase function for a single size parameter is
```

```
     P(one size) = 4 / ( XX**2 * QSCA ) * ( i1 + i2 ) / 2

 where  i1 + i2 = CABS(S1)**2 + CABS(S2)**2.  But it is
 ( i1 + i2 ), not  P(one size), that must be integrated
 over sizes when a size distribution is involved.
 (Physically, this means that intensities are added,
 not probabilities. )  An a posteriori normalization
 then gives the correct size-averaged phase function.

Similarly, it is the CROSS-SECTIONS, proportional to
XX**2  times QEXT,QSCA,QPR, which should be integrated
over sizes, not QEXT,QSCA,QPR themselves.

Similar remarks apply to PMOM.   The normalized moments are
4 / ( XX**2 * QSCA ) * PMOM,  but it is PMOM itself, not
these normalized moments, which should be integrated over
a size distribution.

Unless avoided, ripple spikes can cause a systematic upward
bias in any integration over size parameter, because these
spikes tend to be smeared out by typical quadrature rules and
thus over-represented in the final result.  Checking the output
parameter SPIKE allows the user to filter out these cases.
```

NOTES ON PROGRAM USE

```
*** PMOM dimensioning:

    One user dimensioned PMOM(1,1) in his calling program and
    managed to clobber SFORW because he set MOMDIM=1 and
    internally PMOM is dimensioned PMOM( 0:MOMDIM, * ).
    Fortran seems to allow PMOM(0:0,*) so he could have
    saved himself by setting MOMDIM=0, but this is confusing
    and it is better to start your PMOM array at 0 just as
    the program does internally.

    Be sure to use the test problem drivers as templates
    when designing your calls to MIEV0 in order to avoid this
    kind of problem.

*** ON PORTABILITY :

    This package is written entirely in ANSI standard FORTRAN 77
    and should work on any computer.

*** ON PRECISION :

        "You should be aware that a complex program can produce
    different results on one computer than another because of
    differences in internal precision.  The difference can be
    minimized, but not necessarily eliminated, by using double-
    precision arithmetic and by using numerical methods that tend
    to retain maximum precision."
                                 (IBM Professional FORTRAN Manual)
```

This package was developed on computers offering 14-digit single-precision computation. On IBM-type machines with their 7-digit single precision, parts of the computation (like the upward recurrence for the Ricatti-Bessel functions) might need to be done in double precision, depending on how big the Mie size paramter XX is. See Ref. (1) for further discussion of this point.

The package has only been tested for XX up to 20,000 and for real and imaginary part of CREFIN up to 10 (this accomodates almost all imaginable applications). Slow deterioration of accuracy may be expected if the program is pushed beyond these limits. (Accuracy may degrade well before XX = 20000 with IBM-type 7-digit precision.)

Precision problems are most likely to OVERTLY afflict users

(a) in the self-test subroutine TESTMI, where it may be necessary to lower the required agreement with tabulated results by changing the variable ACCUR. For example, to run on the IBM PC in single precision using 'IBM Professional FORTRAN', a value ACCUR = 1.E-4 was necessary.

(b) in the testing routines MVTst..., where the user's precision may be unsatisfactory for numerically 'sensitive' quantities.

The quantities most sensitive to precision are those involving series of positive and negative terms with much cancellation. The smaller the end result of summing compared to the average term size, the worse the problem. The problem can occur in either of the scattering amplitudes (S1,S2) away from the forward scattering angle, esp. near a relative minimum. When the real and imaginary parts of S1 or S2 differ by orders of magnitude, the smaller part is likely to be less accurate than the larger.

The least accurate output quantities will be :

** TFORW and TBACK, because the numerical factors involved are on the order of XX**3

** PMOM(M, 4) for any M and larger XX

** PMOM(M, NP) for M approaching 2*XX

The most accurate will be QEXT, QSCA, GQSC, being sums of all positive terms.

Please do not call the author about precision problems. They are endemic and cannot be solved as long as different computers do arithmetic differently.

*** ON MEMORY REQUIREMENTS :

The parameter MaxTrm in MIEV0, LPCOEF must be set to 10,100 in order to do the test problems with size parameter = 10,000. Memory used by these routines can be significantly, often dramatically, reduced by lowering MaxTrm to a value no bigger than XMAX + 4*XMAX**1/3, where XMAX is the largest size parameter expected.

If PMOM is never needed, the version MIEV0noP.f should be used instead of MIEV0.f. This can substantially cut memory requirements.

*** The self-test on the first call to the program is a novel feature intended to catch bugs which users may introduce into the code. But it does not begin to test all the possible branches in the code. The test programs included with this package should be used for thorough checkout of all branches.

*** The arithmetic statement function F3 is built into MIEV0, but not used. It corresponds to the function f-sub-3 in Ref. 2, Eq. 8, and should be used instead of F2 when only intensity and degree of polarization are required. This can be implemented just by changing F2 to F3 in a single executable statement.

*** To avoid littering up the code with temporary variables, a reasonably optimizing compiler (one that recognizes invariant and repeated sub-expressions in DO-loops) has been assumed. This may make the code look wasteful to those accustomed to dumb FORTRAN compilers.

*** Equivalenced arrays have been used in one place (module LPCOEF). (EQUIVALENCE is a dangerous feature of FORTRAN and should generally be avoided.)

*** MIEV0 sacrifices some computational speed on vector computers in order to use the minimum possible amount of computer memory; however, it still allows loops over scattering angle to vectorize; and on vector computers which vectorize summing loops (like the Cray), the potentially time-consuming inner loops in the Legendre coefficient subroutine will also vectorize (these two kinds of loops account for the lion's share of computing time in a typical application).