Radiative properties of arbitrary material bodies

Yu-Lin Xu
University of Texas at El Paso - Jacobs JETS Contract, NASA Johnson Space Center, Mail Code XI4-9E, Houston, TX 77058, USA
(yu-lin.xu-1@nasa.gov, yxu5@utep.edu)

Abstract: This work introduces an effective way to study radiative responses of bulk materials, in which a material body is regarded as an array of microscopic particles. Practical examples include the comparison between theory and experiment. Related computer codes are openly available for test.

OCIS codes: (160.4760) Optical properties; (260.1960) Diffraction theory; (290.4020) Mie theory

The interaction of matter with light and other forms of electromagnetic radiation is an area of intensive research with a fascinating interdisciplinary complexion. Responses of bulk material to external sources of electromagnetic radiation are the cooperative reaction of all constituent subdivisions. This is particularly true when the size of an individual component is comparable to a molecule or an atom, in view of the fundamental physical structure of substance. "In the atomic theory, one regards matter as composed of interacting particles (atoms and molecules) embedded in the vacuum" [1]. An atom is the smallest, indivisible unit, which can exist alone and retains all of the chemical properties of an element. For practical purposes, atoms or atom groups are often modeled as solid spheres. In practical calculations, an adequate, finite periodic array (PA) of such infinitesimal "atomic spheres" can be an equivalent to a solid body with the same overall shape and spatial structure.

The present work refers to a feasible approach to the theoretical investigation of the radiative properties of an arbitrary material body through a characteristic, finite PA comprising a huge number of identical component units. An individual constituent unit, not limited to a single particle of simple shape, allows desired configurations of variously shaped particles. The development of this theoretical approach is based on the GMM-PA formulation [2], a special version of the Generalized Multi-particle Mie-solution (GMM) [3]. The standard GMM is an extension of Mie theory [4] for single spheres to clusters of a moderate number of component objects. Related GMM-PA computer codes, currently available upon request for the purpose of test, are implemented mainly for the cases that the standard GMM codes [5] are unable to handle in practical calculations.

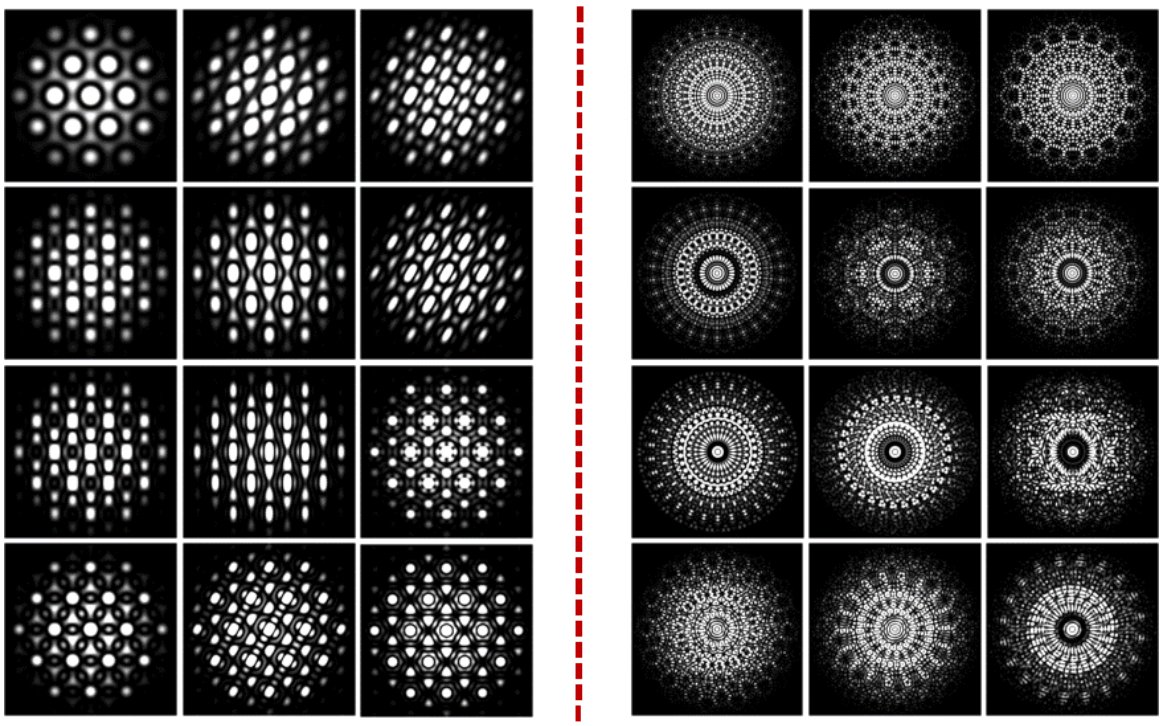

Fig. 1. GMM-PA predicted far-field diffraction patterns for the masks of Plate 4 (left) and Plate 14 (right) provided in the book of "Atlas of Optical Transform" by G. Harburn, C.A. Taylor, and T.R. Welberry [8], which are virtually identical to the experimental results shown in the book.

Diffraction is a shape effect and every shape and spatial structure has a unique diffraction pattern. The GMM-PA formulation, implying a general form of Babinet's theorem, is consistent with Fourier analysis and classical diffraction theories [6,7] in evaluating phase shifts and predicting diffraction patterns. Theoretically predicted diffraction images have been

scrutinized against a collection of experimental results found in literature [e.g., 6-8]. The characteristic, structural features of the theoretically and experimentally obtained diffraction patterns are in uniform agreement. Figure 1 presents some examples to illustrate such theoretical predictions from the GMM-PA. Figure 2 shows the geometrical cross section of a three-dimensional object with uniform thickness (left panel) and the predicted far-field diffraction pattern (right panel), when it is illuminated by a plane wave at 15° oblique incidence.

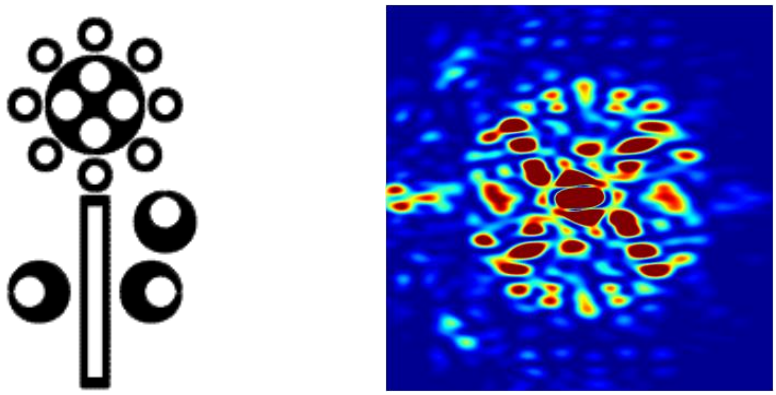

Fig. 2. The geometrical cross section of a 3D diffracting object of uniform thickness (left) and the predicted Fraunhofer diffraction pattern (right), when incident plane wave is unpolarized and at 15° oblique incidence.

Together with the solution to the spatial distribution of excited electromagnetic fields, external and internal to a material body, the GMM-PA approach provides predictions of cross sections, such as for extinction, absorption, and radiation pressure. For illustration, Table 1 lists the predicted dimensionless efficiencies of extinction, absorption and scattering, as well as the asymmetry parameter $\langle\cos\theta\rangle$ for a 0.5x0.5x0.25μm AU block. Incident wavelength is 0.5 μm, at which Au has the bulk refractive index of (0.9656,1.863). The AU block is substituted by a PA having, respectively, 64x64x32, 160x160x80, 320x320x160 component spheres, which are placed in a cubic lattice with separation distance equal to the sphere diameter. The numerical solutions are very close to those obtained from Drain and Flatau's DDSCAT codes [9], an implementation of the Discrete Dipole Approximation (DDA) that replaces a finite target by an array of polarizable points [10,11].

Table 1. Predicted efficiencies and asymmetry parameter of an Au block.

L*	Qext	Qabs	Qsca	$\langle\cos\theta\rangle$
131,072	3.623	1.456	2.166	0.413
2,048,000	3.630	1.459	2.171	0.425
16,384,000	3.632	1.460	2.172	0.429

* L is the total number of component spheres in the array that represents the Au block.

The theoretical approach described here, purely classical, is to put a sufficiently large number of mass points together to probe the radiative properties of material bodies. While taking into account the intricate interactions among the constituent particles, numerical solutions to the excited electromagnetic fields and the total cross sections of an object are sensitive to the behavior of an isolated component particle. The determination and description of the radiative properties of such an isolated point-like particle is critical in the preparation of input parameters, which involves Effective Medium Theories initiated by Maxwell-Garnett and Lorentz [12,13] and the transition between Maxwell's phenomenological and the atomistic theories of matter, especially in microscopic scale.

References

[1] M. Born and E. Wolf, *Principles of Optics*, 7th ed. (Cambridge University Press, 1999), p.89.
[2] Y.-L. Xu, J. Opt. Soc. Am. A **30**, 1053 (2013); **31**, 322 (2014); **32**, 12-21 (2015).
[3] Y.-L. Xu, Appl. Opt. **34**, 4573 (1995); **36**, 9496 (1997); Phys. Lett. A **249**, 30 (1998).
[4] G. Mie, Ann. Phys. **25**, 377 (1908).
[5] GMM codes can be found in either http://www.scattport.org or http://code.google.com/p/scatterlib.
[6] J.W. Goodman, *Introduction to Fourier Optics*, 2nd ed. (McGraw-Hill, 1996).
[7] A. Lipson, S.G. Lipson, and H. Lipson, *Optical Physics*, Fourth Edition, (Cambridge University Press, 2011).
[8] G. Harburn, C.A. Taylor, and T.R. Welberry, *Atlas of Optical Transforms*, (Cornell University Press, 1975).
[9] DDSCAT codes are in http://code.google.com/p/scatterlib.
[10] E.M. Purcell and C.R. Pennypacker, Astrophys. J. **186**, 705 (1973).
[11] B.T. Draine, Astrophys. J. **333**, 848 (1988); B.T. Draine and P. Flatau, J. Opt. Soc. Am. A **11**, 1491 (1994); **25**, 2693 (2008).
[12] J.C. Maxwell-Garnett, Phil. Trans. R. Soc. Lond. A 203, (1904).
[13] H.A. Lorentz, Wiedem. Ann. 9, 641 (1880); L. Lorentz, Wiedem. Ann. 11, 70 (1881).

Appendix. An example main FORTRAN program of GMM-PA.

Note that all the required subroutines are the same as those used in the standard GMM codes, which are available online (see ref. 6).

```
C
C       1       2       3       4       5       6       7
C 34567890123456789012345678901234567890123456789012345678901234567890123456789012
C ---------------------------------------------------------------------
C variable type limits:
C   integer*1 :          -128 to 127
C   integer*2 :       -32,768 to 32,767
C   integer*4 :  -2,147,483,648 to +2,147,483,647
C   real*4    : +/-1.17549e-38  to +/-3.40282e+38
C   real*8    : +/-2.22507e-308 to +/-1.79769e+308
C =====================================================================
C
C GMM01_PA_SB.f
C calculates electromagnetic scattering properties of an arbitrarly
C shaped and structured solid body, which is represented by an array of
C identical constituent spheres
C
C edited from standard GMM codes (such as gmm01s.f ...) to implement the
C GMM-PA scattering formulation, an extension of the standard GMM to the
C special case of finite periodic arrays (PAs) having an enormous number
C of identical component scattering units
C The overall shape of a PA may be either regular or irregular; it can
C also have additinal structures and/or internal openings
C When the physical size of an individual component unit is sufficiently
C small (say, comparable to the size of a molecule or an atom) and thus
C the total number of the component units in a PA is sufficiently large
C (say, >10^6 for a PA with overall dimensions comparable to incident
C wavelength), a densely packed PA of such identical units would be a
C satisfactory representation of a solid body with the same overall
C shape and structure
C
C With the GMM-PA scattering solution approach, an individual scattering
C unit (or a component particle) in a PA is not restricted to a single
C solid body; it may be of an arbitrary, complex multi-body structure
C itself; the only prerequisite is that its proper T-matrix, i.e., the
C scattering coefficients in its isolate scattering, are known or can be
C calculated with satisfactory accuracy
C
C *********************************************************************
C When an individual component scatteringunit in a PA is a homogeneous
C sphere, its proper T-matrix, i.e., the Mie coefficients in its isolate
C single-sphere scattering, will be calculated on the spot; otherwise,
C user must pre-calculate the proper T-matrix of an individual component
C scattering unit in the user-specified spatial orientation, prepared as
C an input data file in the required format
C *********************************************************************
C
C Strictly speaking, the GMM-PA solutions hold precise only for PAs with
C infinite dimensions that have no edge; In practice, however, numerical
```

```
C errors interoduced by the "edge effect" become insignificant when the
C size of an individual component scattering-unit is compatible roughly
C with a molecule or atom of the material of the scattering body and the
C total number of component units in a PA is sufficiently large (roughly
C >>10^5 for PAs with overall dimensions compatible with wavelength)
C
C The development of the GMM-PA approach is reported in the following
C reference papers:
C (1) Y.-L. Xu, "Scattering of electromagnetic waves by periodic
C     particle arrays," J. Opt. Soc. Am. A 30, pp.1053-1068 (2013).
C (2) Y.-L. Xu, "Scattering of electromagnetic radiation by three-
C     dimensional periodic arrays of identical particles," J. Opt. Soc.
C     Am. A 31, pp.322-331 (2014).
C (3) Y.-L. Xu, "Fraunhofer diffraction of electromagnetic radiation by
C     finite periodic structures with regular or irregular overall
C     shapes," J. Opt. Soc. Am. A 32, pp.12-21 (2015).
C
C ---------------------------------------------------------------------
C this GMM-PA code uses "IDshape", an identification number, to classify
C the shape of finite PAs for both overall exterior contour and internal
C opening structures
C   0 - detailed geometrical structure of the scatterer is input by user
C      (as an input data file that provides the precise position vectors
C       of all the component scattering-unit centers)
C   1 - 1D, line
C   2 - 2D, rectangular
C   3 - 2D, circular
C   4 - 2D, oval
C   5 - 2D, triangular
C   6 - 2D, regular polygon
C  20 - 3D, prism having the same 2D cross section (of the type 0 above)
C          all along its length
C  22 - 3D, rectangular prism (including cube), having the same 2D
C          rectangular cross section all along its length
C  23 - 3D, circular cylinder, having the same 2D circular cross section
C          all along its length
C  24 - 3D, prism shape, having exactly the same 2D cross section of an
C          oval along its length
C  25 - 3D, triangular prism, having the same 2D triangle cross-section
C          along its length
C  26 - 3D, octagonal prism, having the same 2D cross section of regular
C          polygon along its length
C  30 - 3D, a portion of sphere
C  31 - 3D, a portion of spheroid
C  32 - 3D, a portion of ellipsoid
C  33 - 3D, sphere
C  34 - 3D, spheroid
C  35 - 3D, ellipsoid
C  40 - 3D, pyramid, having the same irregular shape of cross section
C          with gradually decreasing (from base to a point-like) area
C  42 - 3D, pyramid with rectangular (including square) base
C  43 - 3D, cone (with a base of 2D circular shape)
C  44 - 3D, cone-like, having a 2D oval-shaped base
C  45 - 3D, triangular pyramid
C  46 - 3D, pyramid with a regular-polygon-shaped base
```

```
C
C (NOTE that any additional shapes can be added to the basis shape and
C likewise, geometrical structures can also be taken out to form desired
C internal openings)
C
C ---------------------------------------------------------------------
C The centers of the identical, constituent scattering-units of a finite
C PA must be distributed on the unit-cell corners (points) of a simple
C lattice (of finite dimensions) that just barely encloses the entire
C PA; the lattice is defined by user-input wxsu(1:3) and dx(1:3)
C wxsu(3) - possible minimum lengths along the "principal" dimensions
C        of the simple lattice enclosing just right the entire PA
C dx(3) - unit-cell dimensions of the simple lattice
C *** 1D PAs (i.e., linear chains) and 2D PAs (i.e., plane PAs) are only
C   special cases of 3D; for 1D PAs, wxsu(2)=wxsu(3)=0 and dx(2)=dx(3)=0;
C   similarly, for 2D PAs, wxsu(3)=0 and dx(3)=0 ***
C
C Same as in all standard gmm codes, an arbitrary fixed-orientation of
C a PA is defined by a Euler-angle triad (alpha,beta,gamma), given an
C initial orientation in an initial reference frame, chosen by user in
C an arbitrary but the possibly most convenient way; this is to say, to
C specify a PA's actual spatial orientation to calculate, input data
C must include: (1) an initial orientation of a PA in initial reference
C frame, and (2) a Euler-angle triad for rotating the initial reference
C frame to obtain the desired orientation
C
C As mentioned above, (1) in the initial reference frame, all component
C scattering-unit centers in a PA must be located on the points of a
C simple lattice along the unit-cell dimensions parallel to x, y, or z
C axes, and (2) the orientation of the PA. to be actually calculated,
C is obtained by rotating the initial refrence frame in terms of the
C user-specified Euler-angle triad
C Also, in the scattering calculations in this code, the incident vector
C of the exciting plane wave always points to the positive z direction
C
C ---------------------------------------------------------------------
C This code requires two basic input files:
C (1) "gmm01f.par"
C    to provide required parameters "nL" and "np" for code compiling;
C    (WARNING: NEVER CHANGE THE NAME AND FORMAT OF THIS INPUT FILE
C             AS WELL AS THE TWO PARAMETER NAMES OF "nL" and "np"
C             because it is also used in a number of subroutines)
C
C    an example for "gmm01f.par":
C    "       parameter (nL=30000000,np=10)"
C
C    nL -- the allowed maximum number of component particles, which can
C          conveniently set as the maximum allocation of the available
C          computer memory; However, actually used memory in the
C          scattering calculations depends on the type of the PA (for
C          instance, when IDshape=0, "nL" must be >= the actual total
C          number of the component particles in the PA tobe calculated)
C    np -- the maximum scattering order in the incident and scattered
C          field expansions; "np" must be >= the highest scattering
C          order required in the scattering calculations for the PA;
```

```
C (2) "GMM_PA.in"
C     to input (a) parameters and quantities to instruct what and how
C     to output, (b) detailed information on the physical & geometrical
C     properties of an individual, component scattering unit, and (c)
C     the PA's detailed geometrical structure
C     (WARNING: NEVER CHANGE THE NAME & FORMAT OF THIS INPUT FILE)
C     an example for the required format of "GMM_PA.in":
C     0
C     0.5
C     0
C     1.d-12 1.d-10 1.d-10
C     0.5 0.5
C     0 0 0 0 0 0
C     *** incident wavelength and structure of a scattering unit ***
C     31.416        ! incident wavelength
C     33  0         ! shape ID & switch for interactive scattering
C     0.5 1.6 0.1   ! sphere-radius, complex refractive index
C     1. 1. 1.      ! cubic volume just enclosing the single unit
C     *** geometrical structure of PA ***
C     0. 0. 0.      ! Euler-angle triad
C     2  102        ! # of dimensions & ID # for overall shape
C     1. 1.         ! lattice separations
C     1000. 1000.   ! array overall lengths
C     1000. 1000.   ! radius of a just PA-enclosing sphere or circle
C     *** additional shapes and/or internal openings ***
C     add_or_opens.dat ! data file to add and/or to take out shapes
C
C     (See the body of this code for the exact meaning of each entry)
C
C ---------------------------------------------------------------------
C
C Yu-Lin Xu
C
C January 2015: added the option for a PA to include desired internal
C   openings (i.e., to take out any geometrical structure from inside)
C
C For questions/comments/suggestions/bugs/problems please contact
C Yu-Lin Xu at yu-lin.xu-1@nasa.gov or yxu5@utep.edu
C
C =====================================================================
      PROGRAM GMM01_PA_SB
      implicit double precision (a-h,o-z)
      parameter (nLp=1)
      include 'gmm01f.par'
C ---------------------------------------------------------------------
C "gmm01f.par" inputs the two parameters of "nL" and "np"
C an example:
C     "       parameter (nL=20000000,np=6)"
C REMEMBER: never change this file name of "gmm01f.par" !!
C ---------------------------------------------------------------------
      parameter (nmp=np*(np+2),nmp0=(np+1)*(np+4)/2)
      parameter (NXMAX=15000,nangmax=1801,MOR=1801,ncmax=3600)
C ---------------------------------------------------------------------
C NXMAX - the maximum dimension of an auxiliary array in calculating
C         Mie scattering coefficients when the component particle is a
```

```
C       homogeneous sphere, which must not be smaller than
C             1.1*np*|m|+10
C       where |m| is the refractive index of the component spheres
C -------------------------------------------------------------------
      parameter (ni0=np*(np+1)*(2*np+1)/3+np*np)
      parameter (ng0=np*(2*np**3+10*np**2+19*np+5)/6)
      parameter (nrc=4*np*(np+1)*(np+2)/3+np)
      integer u,v,u0,v0,nmax(nLp),uvmax(nLp),ind(nLp),ind2(nLp)
      integer nx(3),nx1(3),nx0(3),nxc(3),nx1min(3),nx1max(3)
      integer id_uc(nL)
      double precision k,lnfacd,BESSJ1d,kp,
     + x(nLp),dang(nangmax),wx(5),wxsu(3),w1(np),w2(np),w3(np),
     + w4(np),r0(6,nLp),r00(3,nL),dx(3),rdist(nL),ctpha(3,3),
     + rsr0(NXMAX),rsi0(NXMAX),rsx0(NXMAX),px0(NXMAX),rsr(np,nLp),
     + rsi(np,nLp),rsx(np,nLp),px(np,nLp),betar(MOR),thetr(MOR),
     + phair(MOR),smue(4,4),
     + besj(0:2*np+1),besy(0:2*np+1),i11(nangmax),i21(nangmax),
     + i22(nangmax),i12(nangmax),inat(nangmax),pol(nangmax),
     + cscaxi(nLp),cscayi(nLp),cextxi(nLp),cextyi(nLp),
     + cabsxi(nLp),cabsyi(nLp),cexti(nLp),cabsi(nLp),
     + cscai(nLp),assymi(nLp),assymxi(nLp),assymyi(nLp),
     + cprxi(nLp),cpryi(nLp),cpri(nLp),drot(nrc),c0i(nLp),c1i(nLp)
      complex*16 A,B,cmz,Aj,Bj,A2,B2,Aj2,Bj2,A0,B0,ephi,ci,cin,Au,Bu,
     + phabak,tphac,
     + atr0(ni0),btr0(ni0),atr(2,np,nmp),at(nmp),bt(nmp),ek(np),
     + ref(nLp),ref0(nLp),p0(nLp,nmp),q0(nLp,nmp),an(np),bn(np),
     + aMie(nLp,np),bMie(nLp,np),as(nLp,nmp),bs(nLp,nmp),
     + as0(nLp,nmp),bs0(nLp,nmp),asc(nLp,nmp),bsc(nLp,nmp),
     + as1(nLp,nmp),bs1(nLp,nmp),ast(nLp,nmp),bst(nLp,nmp),
     + asp(nLp,nmp),bsp(nLp,nmp),asv(nLp,nmp),bsv(nLp,nmp),B2i(nLp),
     + s2x(ncmax,nangmax),s4x(ncmax,nangmax),s3y(ncmax,nangmax),
     + s1y(ncmax,nangmax),atj(nmp),btj(nmp),py0(NXMAX),py(NXMAX),
     + dpy(NXMAX),amnuv(nmp,nmp),bmnuv(nmp,nmp),scamu(-np:np,-np:np,2),
     + pmnuv(2,1:np,nmp),qmnuv(2,1:np,nmp),extmu(2,-np:np,2)
      CHARACTER FLNAME0*30,fileout*20,fileout2*22,filenm_sutmx*60,
     + tailn*3,cnr2*2,cnr3*3,cnr1*1,flout*22,fileSUamn*30,
     + FLNAME*30,filenm_sutmx_x*50,filenm_sutmx_y*50
      COMMON/MIESUB/twopi,pih
      common/rot/bcof(0:np+2),dc(-np:np,0:nmp)
      common/fnr/fnr(0:2*(np+2))
      common/pitau/pi(nmp0),tau(nmp0)
      common/tran/atr
      common/ig0/iga0(ni0)
      common/g0/ga0(ng0)
      common/cofmnv0/cof0(ni0)
      common/crot/cofsr(nmp)
      logical flag
c
      pih=dacos(0.d0)
      twopi=4.d0*pih
      pione=2.d0*pih
      ci=dcmplx(0.d0,1.d0)
      cin=dcmplx(0.d0,-1.d0)
      n_entry=0
      gcs=0.d0
```

```
      gcv=0.d0
      idpq=0
      facPAnL=0.5d0
      idMie=0
      MXINT=600
      idPAcase=0
      idsuTMX=0
      do j=1,3
       dx(j)=0.d0
       wx(j)=0.d0
       wxsu(j)=0.d0
       nx(j)=1
      enddo
      wx(4)=0
      wx(5)=0
C =====================================================================
C read in required input data
C ---------------------------------------------------------------------
      inquire(file='GMM_PA.in',exist=flag)
      if(.not.flag) then
       write(*,*) 'FATAL ERROR: needs input data file GMM_PA.in'
       stop
      endif
      open(1,FILE='GMM_PA.in',STATUS='OLD')
      read(1,*)
c the first line could be some comment or just leaves blank
      read(1,*,err=10) facPAnL
      write(*,101) 'fraction for the usage of nL: ',facPAnL
      if(facPAnL.lt.0.001d0.or.facPAnL.gt.0.9d0) then
       write(*,*) 'facPAnL is better not too small and <0.9: ',facPAnL
       write(*,*) 'it has been changed to 0.9'
       facPAnL=0.9d0
      endif
C ---------------------------------------------------------------------
C facPAnL -- a factor (fraction) for how much of "nL" (the parameter
C            provided in "gmm01f.par") will be used for the solution of
C            the interactive scattered field of an individual partice,
C            especially useful for 1- and 2-dimensional arrays when the
C            "nL" parameter is fixed to the maximum possible for the
C            computer used to run this code
C            (remember: when "nL", i.e., the input file "gmm01f.par"
C             changed, this code must be re-compiled)
C            (the maximum and default value is 1.0)
C ---------------------------------------------------------------------
      read(1,*,err=10) NADD
C ---------------------------------------------------------------------
C NADD is the number of terms to add to the scattering orders required
C by the Wiscombe criterion when calculating the Mie coefficients of a
C sphere, which can be negative or positive in the range of [-9,99]
C NOTE: NADD is usally used for the only case when the component unit in
C a PA is a homogeneous sphere
C default: NADD=0
C ---------------------------------------------------------------------
      write(*,102)
     + 'Scat. orders added to Wiscombe criterion:',NADD
```

```
      read(1,*,err=10) eps,small,eps_PA
      if(eps.gt.1.d-20) eps=1.d-20
C -------------------------------------------------------------------
C eps: error tolerance for determining single-sphere field-expansion
C      truncation when the component particle is sphere, default: 1.d-20
C      (the default value of 1.d-20 allows to use Wiscombi's criterion)
C small: error tolerance for the iterative solution process for solving
C        the interacting scattering coefficients of a component particle
C        (say, 1.d-6)
C eps_PA: error tolerance in solving the interactive scattering of an
C         individual component particle
C -------------------------------------------------------------------
      write(*,103) 'error tolerance for Mie-expansions:',eps
      write(*,103) 'Convergence criterion:',small
      write(*,103) 'Convergence criterion for transs:',eps_PA
      fint=0.d0
      read(1,*,err=10) sang,pang
C -------------------------------------------------------------------
C sang -- the scattering angle interval for output
C      example: when sang=1, the results in output will be written for
C               every degree of scattering angle from 0 to 180
C pang -- the azimuth angle interval for the two-dimensional Mueller
C         matrix output
C     (1) For each azimuth angle, the number of scattering angles that
C         will be calculated is the same as that determined by "sang",
C         i.e., the number of scattering angles that will calculated is
C         the same as calculated for the scattering plane of the azimuth
C         angle=0 (180/sang +1).
C     (2) When pang = 0., no additional calculations for the scattering
C         matrix map, i.e., calculating only the scattering plane of the
C         azimuth angle = 0.
C     (3) When pang>0, the number of azimuth angles in the range of
C         (0,360) degrees that will be calculated in addition to 0 (360)
C         degrees is npng-1 with npng=360/pang. For example, when
C         pang=180, npng=2, the azimuth angles 0 (360) and 180 degrees
C         will be calculated. In the output for the Mueller matrix at
C         each azimuth angle, there are nang2 (=2*nang-1) sets of the
C         16 elements.
C -------------------------------------------------------------------
      write(*,101) 'scat.-angle-interval in output: ',sang
      if(sang.le.0.d0) sang=1.d0
      nang=90.d0/sang
      nang=nang+1
      nang2=2*nang-1
      if(nang2.gt.nangmax) then
       write(*,*) 'sang too small'
       write(*,*) 'please increase sang in the input file GMM_PA.in'
       write(*,*) '  and try again, or'
       write(*,*) '  increase nangmax in the parameter line of the'
       write(*,*) '  main code, recompile, then try again'
       stop
      endif
      write(*,101)
     + 'azimuth-angle-interval in Mueller matrix output: ',pang
      if(pang.lt.0.0001d0) then
```

```
       npng=1
      else
       npng=360.0d0/pang
      endif
      if(npng.gt.ncmax) then
       write(*,*) 'pang too small'
       write(*,*)
     +  'please increase pang in the input file GMM_PA.in'
       write(*,*)
     +  'and try again, or increase ncmax in the parameter'
       write(*,*)
     +  ' line of the main code, recompile, then try again'
       stop
      endif
      read(1,*,err=10) idphoto,nphoto,dphi,istart,iend,istep
C ---------------------------------------------------------------------
C idphoto=0 or nphoto=0 for not calculating internal field distributions
C idphoto=1 for calculating internal field distributions in spherical
C          coordinates r/a [0,1] and theta [0,360] for each cross
C          section at angle phi
C idphoto=2 for calculating internal field distributions in Cartesian
C          coordinates in x'/a [-1,1] and z/a [-1,1] for each cross
C          section at angle phi
C nphoto+1: number of mesh points in (r/a,theta) or in (x'/a,z/a)
C          nphoto must be an even number
C dphi:    dphi=0 for calculating phi=0 only
C          otherwise, the calculated number of phi's would be 180/dphi
C          (dphi >= 0.1)
C The internal field distributions can be calculated for a selected
C component sphere or a selected range of spheres by specifying
C (istart,iend,istep). For example, only the internal field distribution
C of the 5th sphere will be calculated when (istart,iend,istep)=(5,5,1).
C When (istart,iend,istep)=(1,5,2), the internal field distributions of
C the 1st, 3rd, and 5th spheres will be calculated. When istart=1,
C iend=nL, and istep=1, the internal distributions will be calculated
C for all component spheres.
C Note that this code calculates the internal distributions for only a
C fixed orientation and does not average over orientations. Thus, it
C calculates the internal fields for only the first orientation.
C Note also that the output files for the internal field distributions
C have a large size, especially when nphoto is large.
C ---------------------------------------------------------------------
      if(idphoto.gt.0) then
       write(*,104) 'idphoto,nphoto,dphi: ',
     +            idphoto,nphoto,dphi,istart,iend,istep
      else
       write(*,102) 'idphoto: ',idphoto
      endif
C ***********************************************************************
C read in incident wavelength and the required information on the PA's
C individual constituent unit
C ***********************************************************************
      read(1,*)
      read(1,*,err=10) w,bulkmr,bulkmi
C ---------------------------------------------------------------------
```

```
C  w -- incident wavelength
C ---------------------------------------------------------------------
      read(1,*,err=10) idsu0,idsuTMX
C     idsuTMX=0
      write(*,*) idsu0,idsuTMX
C ---------------------------------------------------------------------
C idsu0 - ID for the geometry of a PA's component scattering unit, which
C         is taken from the list of IDshape
C    idsu0=33: a homogeneous sphere, its proper T-matrix, i.e, the Mie
C    coefficients in its single-sphere scattering will be calculated on
C    the spot via Mie theory; for all other cases, the proper T-matrix
C    of an individual component scattering unit, i.e., the scattering
C    coefficients in its isolate, single-body scattering, must be
C    pre-calculated and provided as input data
C idsuTMX - ID for calculating or directly inputing the interactive
C           scattering coefficients of an individual scattering unit
C    idsuTMX=0: to be calculated on the spot
C    idsuTMX=1: to be direct input from the existing data files named as
C              "scacof_x.dat" and "scacof_y.dat"
C ---------------------------------------------------------------------
      write(*,*) 'idsu0 (shape ID for an individual component particle:'
      write(*,*) idsu0
C
      if(idsu0.ne.33) then
       write(*,*) 'This code works only for the special case when the'
       write(*,*) 'scattering unit in a PA is a sphere; please change'
       write(*,*) 'idsu0 to 33, make corresponding changes in related'
       write(*,*) 'input files if necessary, and then try again'
       stop
      endif
C
      if(idsu0.eq.33) then
       read(1,*,err=10) (r0(j,1),j=4,6)
       write(*,*) 'an individual component particle is a homogeneous'
       write(*,*) 'sphere, its proper scattering coefficients are '
       write(*,*) 'calculated via Mie theory'
      else
       read(1,*,err=10) filenm_sutmx_x,filenm_sutmx_y
       write(*,*) 'component particle in the PA is not a sphere; its'
       write(*,*) 'proper scattering coefficients, i.e., the proper'
       write(*,*) 'T-matrices in its isolate, single-body scattering,'
       write(*,*) 'are pre-calculated and placed in the two existing'
       write(*,*) 'input data files for the two orthogonal linear'
       write(*,*) 'polarization states (with 0 or 90 deg linear'
       write(*,*) 'polarization angle) of the incident plane wave:'
       write(*,*) filenm_sutmx_x
       write(*,*) filenm_sutmx_y
       inquire(file=filenm_sutmx_x,exist=flag)
       if(.not.flag) then
        write(*,*) 'FATAL ERROR found:'
        write(*,*) 'The following input data file does not exist'
        write(*,*) filenm_sutmx_x
        stop
       endif
       inquire(file=filenm_sutmx_y,exist=flag)
```

```
       if(.not.flag) then
        write(*,*) 'FATAL ERROR found:'
        write(*,*) 'The following input data file does not exist'
        write(*,*) filenm_sutmx_y
        stop
       endif
       write(*,*) 'NOTE: the above two input data files must have'
       write(*,*) '      the exact format as required and for the'
       write(*,*) '      exact spatial orientation as specified by'
       write(*,*) '      the input Euler-angle triad'
      endif
      write(*,*) 'idsuTMX for the optional direct input of interactive'
      write(*,*) 'scattering coefficients: ',idsuTMX
      if(idsuTMX.gt.0) then
       write(*,*) 'Interactive scattering coefficients of a component'
       write(*,*) 'particle are provided in the two existing files'
       write(*,*) 'of scacof_x.dat and scacof_y.dat for the two'
       write(*,*) 'orthogonal incident linear polarization status of'
       write(*,*) '0 and 90 deg, respectively'
       inquire(file='scacof_x.dat',exist=flag)
       if(.not.flag) then
        write(*,*) 'ERROR: idsuTMX=1, needs data file scacof_x.dat'
        stop
       endif
       inquire(file='scacof_y.dat',exist=flag)
       if(.not.flag) then
        write(*,*) 'ERROR: idsuTMX=1, needs data file scacof_y.dat'
        stop
       endif
      endif
      read(1,*,err=10) (wxsu(j),j=1,3)
C -------------------------------------------------------------------
C wxsu(3) - the three minimum dimensions of a rectangular volume that
C           just encloses entirely the PA's component unit
C an example: when the component particle is a sphere,
C             wxsu(1) = wxsu(2) = wxsu(3) = sphere-diameter
C NOTE:
C (1) this is NOT the lattice unit-cell dimensions,
C (2) it is for the only purpose of checking whether the component
C     scattering units of a PA are overlapped or not by comparing "wxsu"
C     with lattice unit-cell separations "dx"
C     !! the scattering units of a PA must not overlap
C -------------------------------------------------------------------
      if(idsu0.eq.33) then
       temp=2.d0*r0(4,1)
       do j=1,3
        wxsu(j)=temp
       enddo
      endif
      write(*,105) (wxsu(j),j=1,3)
C ***********************************************************************
C read in the detailed information on the PA's geometrical structure
C such as the spatial orientation and number of dimensions of the PA,
C lattice unit-cell dimensions, and the distribution of the centers of
C the identical component units of the finite PA in the simple lattice
```

```
C in its initial orientation
C *********************************************************************
      read(1,*)
      read(1,*,err=10) alpha_PA,beta_PA,gamma_PA
C ---------------------------------------------------------------------
C alpha_PA,beta_PA,gamma_PA -- the triad of Euler angles to deterimine
C     the spatial orientation of the sphere array to be calculated
C     (0,0,0) for the default orientation
C ---------------------------------------------------------------------
      read(1,*,err=10) idimen,IDshape
C ---------------------------------------------------------------------
C idimen: 1, 2, or 3 to indicate the array is 1D, 2D or 3D
C IDshape: chosen from the definition list in terms of the overall shape
C          of the basis PA
C ---------------------------------------------------------------------
      read(1,*,err=10) (dx(j),j=1,3)
      read(1,*,err=10) (wx(j),j=1,5)
C ---------------------------------------------------------------------
C dx(3) - unit-cell dimensions of the simple lattice used to specify the
C spatial distribution of the centers of the component units of a PA
C wx(1:3) - the three minimum dimensions of a rectangular volume that
C     just enclose the entire PA; when IDshape = 2 or 22, "wx" is simply
C     the overall "principal" dimensions of a PA
C ---------------------------------------------------------------------
      call chkInp_getTotNum(idimen,IDshape,wx,dx,dnLPA)
      write(*,*) 'The three principal, overall dimensions of the PA:'
      write(*,105) (wx(j),j=1,3)
      if(IDshape.ge.30.and.IDshape.le.35)
     +  write(*,*) 'z-ends: ',wx(4),wx(5)
C ---------------------------------------------------------------------
C when IDshape=0, the position vector of every component-particle
C center must be directly input from an existing data file
C ---------------------------------------------------------------------
      read(1,*)
      if(IDshape.eq.0) then
       read(1,*,err=10) FLNAME0
       inquire(file=FLNAME0,exist=flag)
       if(.not.flag) then
        write(*,*) 'The input data file dies not exist: ',FLNAME0
        goto 10
       endif
       IDmoreShapes=0
       FLNAME=' '
       goto 11
      else
       FLNAME0=' '
       read(1,*)
      endif
      read(1,*)
      read(1,*) IDmoreShapes
      write(*,*) 'IDmoreShapes: ',IDmoreShapes
      if(IDmoreShapes.eq.0) then
       FLNAME=' '
       goto 11
      endif
```

```
      read(1,*,err=10) FLNAME
C ---------------------------------------------------------------------
C FLNAME: the name of an additional input data file as required
C when IDshape = 0, it provides the position vector of every component
C scattering-unit center in a simple lattice; otherwise (i.e., in all
C other cases), it specifies the detailed geometrical structure(s) of
C additional external shape(s) and/or internal opening(s)
C ---------------------------------------------------------------------
      inquire(file=FLNAME,exist=flag)
      if(.not.flag) then
       write(*,*) 'The input data file dies not exist: ',FLNAME
       goto 10
      endif
      goto 11
10    write(*,*) 'FATAL ERROR found in the input file GMM_PA.in'
      stop
11    close(1)
C ---------------------------------------------------------------------
C individual component scattering units must not overlap
C ---------------------------------------------------------------------
      do j=1,idimen
       temp=dx(j)-wxsu(j)
       if(temp.lt.0.d0) then
        write(*,*) 'ERROR: component scattering units may overlap'
        write(*,*) dx(j),wxsu(j)
        write(*,*) 'please check input data dx and wxsu,'
        write(*,*) 'make necessary corrections and then try again'
        stop
       endif
      enddo
C ---------------------------------------------------------------------
c read in the user-defined spatial distribution of component-particle
c centters of the PA when IDshape=0
C ---------------------------------------------------------------------
      if(IDshape.ne.0) goto 22
      open(3,file=FLNAME0,status='old')
      read(3,*,err=200) dnLPA
      if(dnLPA.gt.2.d9) then
       write(*,*) 'total number of component particles, nLPA, excedds'
       write(*,*) 'the integer variable limit when directly inputing'
       write(*,*) 'the detaild spatial distribution of the component'
       write(*,*) 'particle centers, which must be < ~2e9: ',dnLPA
       write(*,*) 'please check the number, make corrections in the'
       write(*,*) 'input configuration file, and then try again'
      endif
      nLPA=dnLPA
      if(nLPA.gt.nL) then
       write(*,*) 'total number of particles in the input config'
       write(*,*) 'file exceeds the allowed nL; please reduce'
       write(*,*) 'the particle number or change the value of nL'
       write(*,*) 'in the input parameter file gmm01f.par, then'
       write(*,*) 'run again'
       write(*,*) 'nL must >= nLPA: ',nL,nLPA
       stop
      endif
```

```
      do i=1,nLPA
       read(3,*,err=200) (r00(j,i),j=1,3)
      enddo
      close(3)
      goto 201
200   write(*,*) 'Somthing wrong in the input configuration file:'
      write(*,*) FLNAME0
      write(*,*) 'please make necessary corrections and try again'
      stop
201   open(10,file='z0sec.cfg',status='unknown')
      open(12,file='y0sec.cfg',status='unknown')
      open(15,file='x0sec.cfg',status='unknown')
      open(16,file='xyz3D.cfg',status='unknown')
      x1=0
      y1=0
      z1=0
      do j=1,nLPA
       x1=x1+r00(1,j)
       y1=y1+r00(2,j)
       z1=z1+r00(3,j)
      enddo
      temp=nLPA
      x1=x1/temp
      y1=y1/temp
      z1=z1/temp
      do j=1,nLPA
       r00(1,j)=r00(1,j)-x1
       r00(2,j)=r00(2,j)-y1
       r00(3,j)=r00(3,j)-z1
      enddo
      do i=1,nLPA
       id_uc(i)=i
       rdist(i)=0.d0
       do j=1,3
        rdist(i)=rdist(i)+r00(j,i)**2
       enddo
      enddo
      call isort2(nLPA,rdist,id_uc)
      x1=r00(1,id_uc(1))
      y1=r00(2,id_uc(1))
      z1=r00(3,id_uc(1))
      do j=1,nLPA
       r00(1,j)=r00(1,j)-x1
       r00(2,j)=r00(2,j)-y1
       r00(3,j)=r00(3,j)-z1
       x0=r00(1,j)
       y0=r00(2,j)
       z0=r00(3,j)
       write(16,105) x0,y0,z0
       if(z0.eq.0.d0) write(10,106) x0,y0
       if(y0.eq.0.d0) write(12,106) x0,z0
       if(x0.eq.0.d0) write(15,106) y0,z0
      enddo
      call rotatEuler3(alpha_PA,beta_PA,gamma_PA,nLPA,nL,r00)
      call ctotpha3(alpha_PA,beta_PA,gamma_PA,ctpha)
```

```
      write(*,*) 'IDshape = 0',idimen,nLPA
      write(*,*) id_uc(1)
      write(*,*) (r00(j,id_uc(1)),j=1,3)
      close(10)
      close(12)
      close(15)
      close(16)
      nLPA0=nLPA
C ======================================================================
C input-data reading complete
C begins working on both proper & interactive scattering coefficients
C for an individual scattering unit
C ----------------------------------------------------------------------
22    if(IDshape.eq.0) goto 16
c     if(idsuTMX.gt.0) goto 16
C ----------------------------------------------------------------------
C based on the input dx(1:3), constructing a simple PA to calculate the
C scattered field of an individual scattering unit; the total number
C of the component units in the PA must not be greater than  the input
C parameter "nL" in "gmm01f.par"
C NOTE: this step needs to be done only for the case of idsuTMX=0 when
C the interactive scattering coefficients need to be calculated; when
C idsuTMX>0, the interactive scattering coefficients are already known
C and this step is skipped
C ----------------------------------------------------------------------
      n0=(nL)*facPAnL
      if(n0.gt.nL) n0=nL
      temp0=n0
      goto(23,24,28) idimen
23    nLPA0=n0
      temp1=wx(1)/dx(1)
      if(temp1.lt.temp0) nLPA0=temp1
      nx0(1)=nLPA0
      nx0(2)=1
      nx0(3)=1
      write(*,*) 'Calculating a linear 1-D PA'
      write(*,*) 'nLPA0: ',nLPA0
      write(*,*) (nx0(j),j=1,3)
      write(*,*) (dx(j),j=1,3)
      goto 30
24    temp1=wx(1)/dx(1)
      temp2=wx(2)/dx(2)
      temp=temp1*temp2
      if(temp.le.temp0) then
       nx0(1)=temp1
       nx0(2)=temp2
       goto 25
      endif
      st=dsqrt(temp0)
      nx0(1)=st
      nx0(2)=st
      if(temp1.lt.st) then
       nx0(1)=temp1
       nx0(2)=temp0/temp1
       goto 25
```

```
      endif
      if(temp2.lt.st) then
       nx0(2)=temp2
       nx0(1)=temp0/temp2
      endif
25    nLPA0=nx0(1)*nx0(2)
      nx0(3)=1
      write(*,*) 'Calculating a 2-D PA'
      write(*,*) 'nLPA0: ',nLPA0
      write(*,*) (nx0(j),j=1,3)
      write(*,*) (dx(j),j=1,3)
      goto 30
28    temp1=wx(1)/dx(1)
      temp2=wx(2)/dx(2)
      temp3=wx(3)/dx(3)
      temp=temp1*temp2*temp3
      if(temp.le.temp0) then
       nx0(1)=temp1
       nx0(2)=temp2
       nx0(3)=temp3
       goto 29
      endif
      st=(temp0)**(1.d0/3.d0)
      nx0(1)=st
      nx0(2)=st
      nx0(3)=st
      if(temp1.lt.st) then
       nx0(1)=temp1
       ct=dsqrt(temp0/temp1)
       nx0(2)=ct
       nx0(3)=ct
       if(temp2.lt.ct) then
        nx0(2)=temp2
        nx0(3)=temp0/temp2/temp1
        goto 29
       endif
       if(temp3.lt.ct) then
        nx0(3)=temp3
        nx0(2)=temp0/temp3/temp1
        goto 29
       endif
       goto 29
      endif
      if(temp2.lt.st) then
       nx0(2)=temp2
       ct=dsqrt(temp0/temp2)
       nx0(1)=ct
       nx0(3)=ct
       if(temp1.lt.ct) then
        nx0(1)=temp1
        nx0(3)=temp0/temp1/temp2
        goto 29
       endif
       if(temp3.lt.ct) then
        nx0(3)=temp3
```

```
       nx0(1)=temp0/temp3/temp2
       goto 29
      endif
      goto 29
     endif
     if(temp3.lt.st) then
      nx0(3)=temp3
      ct=dsqrt(temp0/temp3)
      nx0(1)=ct
      nx0(2)=ct
      if(temp1.lt.ct) then
       nx0(1)=temp1
       nx0(2)=temp0/temp1/temp3
       goto 29
      endif
      if(temp2.lt.ct) then
       nx0(2)=temp2
       nx0(1)=temp0/temp2/temp3
      endif
     endif
29    nLPA0=nx0(1)*nx0(2)*nx0(3)
     write(*,*) 'Calculating a 3-D PA'
     write(*,*) idimen,nLPA0
     write(*,*) (nx0(j),j=1,3)
     write(*,*) (dx(j),j=1,3)
30    if(nLPA0.gt.nL) then
      write(*,*) 'nLPA0>nL, which should not happen'
      write(*,*) 'please check the source code'
      write(*,*) nLPA0,nL
      stop
     endif
C ---------------------------------------------------------------------
C preparing the PA to solve the partial scattered field from a component
C scattering unit: at first, complete the construction of the PA at its
C initial orientation, and then rotate the initial reference frame in
C terms of the Euler-angle triad specified in input
C (at the same time a 3x3 matrix "ctpha" is calculated for later use in
C the calculation of incident and scattered phase terms)
C ---------------------------------------------------------------------
     call getrgr3Dcfg(idimen,nLPA0,nx0,dx,r00)
     call rotatEuler3(alpha_PA,beta_PA,gamma_PA,nLPA0,nL,r00)
     call ctotpha3(alpha_PA,beta_PA,gamma_PA,ctpha)
C ---------------------------------------------------------------------
C sorting component particle centers by distance from origin
C ---------------------------------------------------------------------
     write(*,107)
     write(*,*) 'sorting particle centers by distance from origin'
     do i=1,nLPA0
      id_uc(i)=i
      rdist(i)=0.d0
      do j=1,3
       rdist(i)=rdist(i)+r00(j,i)**2
      enddo
     enddo
     call isort2(nLPA0,rdist,id_uc)
```

```
      write(*,*) 'Particle centers 1 and nLPA0 after sorting:'
      write(*,108) id_uc(1),(r00(j,id_uc(1)),j=1,3)
      write(*,108) id_uc(nLPA0),(r00(j,id_uc(nLPA0)),j=1,3)

C ---------------------------------------------------------------------
C calculating the proper Mie-scattering coefficients when an individual
C scattering unit is a homogeneous sphere; otherwise, reading in the
C proper scattering coefficients from the specified input data files
C ---------------------------------------------------------------------
16    k0=twopi/w
CC      if(r0(6,1).gt.0.d0) r0(6,1)=-r0(6,1)
c16    A=dcmplx(bulkmr,bulkmi)
c      A=A*A
c      emtmr=cdabs(A)
c      emtmr=dsqrt(emtmr)
      emtmr=bulkmr
cc      if(idimen.lt.3.or.bulkmr.lt.1.d0) emtmr=1.d0
c      emtmr=1.d0
c      wp=w/emtmr
      wp=w
CC 16    w=w/bulkmr
CC      r0(5,1)=r0(5,1)/bulkmr
CC      r0(6,1)=r0(6,1)/bulkmr
c      A=dcmplx(bulkmr,bulkmi)
c      B=dcmplx(r0(5,1),r0(6,1))
c      A=A/B
c      A=cdsqrt(A)
c      B=dcmplx(0.d0,-1.d0)
c      r0(5,1)=A
c      r0(6,1)=B*A
c      write(*,*) 'relative ref. index: ',r0(5,1),r0(6,1)
cxu      write(*,*) 'wavelength inside body: ',wp,emtmr
ccc16    write(*,*) 'particle refrac. index: ',r0(5,1),r0(6,1)
      write(*,*) 'particle refrac. index: ',r0(5,1),r0(6,1)
c      r0(5,1)=r0(5,1)/emtmr
c      r0(6,1)=r0(6,1)/emtmr
c      write(*,*) 'relative refrac. index: ',r0(5,1),r0(6,1)
      if(r0(5,1).lt.0.d0.or.r0(6,1).lt.0.d0) then
       write(*,*) 'refractive index must be >0, please check input data'
       stop
      endif
      if(r0(6,1).gt.0.d0) r0(6,1)=-r0(6,1)
      kp=twopi/wp
      k=kp
ccc      write(*,*) 'wavelength inside body: ',wp,emtmr
ccc      k=k0
ccc      kp=k
C ---------------------------------------------------------------------
C Note that the negative sign is for the subroutine of calculating Mie
C coefficients from R.T.Wang only because he uses himonic time
C dependence of exp(iwt) instead of exp(-iwt) as in GMM
C ---------------------------------------------------------------------
      x(1)=k*r0(4,1)
      if(idimen.eq.1) temp=4.d0*dx(1)*r0(4,1)**2
      if(idimen.eq.2) temp=2.d0*dx(1)*dx(2)*r0(4,1)
```

```
      if(idimen.eq.3) temp=dx(1)*dx(2)*dx(3)
      rimg=(0.75d0*temp/pione)**(1.d0/3.d0)
c       x(1)=k*rimg
      ref(1)=dcmplx(r0(5,1),r0(6,1))
      ref0(1)=dcmplx(r0(5,1),-r0(6,1))
      do j=1,np
       aMie(1,j)=0.d0
       bMie(1,j)=0.d0
      enddo
      nmax0=1
      write(*,107)
      write(*,109) 'single-sphere size-parameter: ',x(1)
c ----------------------------------------------------------------------
c  calculating Mie-scattering coefficients for each spheres
c  the ratio method of Wang and van der Hulst is used in calculating
c  Riccati-Bessel functions [see Wang and van der Hulst, Appl. Opt.
c  30, 106 (1991), Xu, Gustafson, Giovane, Blum, and Tehranian,
c  Phys. Rev. E 60, 2347 (1999)]
c ----------------------------------------------------------------------
      call abMiexud(x(1),ref(1),np,NXMAX,nmax(1),an,bn,NADD,
     +              rsr0,rsi0,rsx0,px0,w1,w2,w3,w4,eps)
      if(nmax(1).gt.np) then
       write(*,*) 'Parameter np too small, must be >',nmax(1)
       write(*,*) 'Please change np in gmm01f.par,'
       write(*,*) 'recompile, then try again'
       stop
      endif
      uvmax(1)=nmax(1)*(nmax(1)+2)
      write(*,110) 'Actual single-sphere expansion truncation:',nmax(1)
      i=1
      do j=1,nmax(i)
       rsr(j,i)=rsr0(j)
       rsi(j,i)=rsi0(j)
       rsx(j,i)=rsx0(j)
       px(j,i)=px0(j)
       temp1=an(j)
       temp2=bn(j)
       if(j.eq.1.or.j.eq.nmax(i)) write(*,111)
     +              j,temp1,dimag(an(j)),temp2,dimag(bn(j))
      enddo
      do j=1,nmax(i)
       aMie(i,j)=an(j)
       bMie(i,j)=bn(j)
       rsr(j,i)=rsr0(j)
       rsi(j,i)=rsi0(j)
       rsx(j,i)=rsx0(j)
       px(j,i)=px0(j)
      enddo
      if(nmax(i).gt.nmax0) nmax0=nmax(i)
c ----------------------------------------------------------------------
c begins scattering solution process with preparing, at first, necessary
c functions for evaluating vector translation coefficients
c ----------------------------------------------------------------------
      idcorrf=0
      cextx=0.d0
```

```
      cexty=0.d0
      cabsx=0.d0
      cabsy=0.d0
      cscax=0.d0
      cscay=0.d0
      cprx=0.d0
      cpry=0.d0
      cbakx=0.d0
      cbaky=0.d0
      cextx_ck=0.d0
      cexty_ck=0.d0
      cabsx_ck=0.d0
      cabsy_ck=0.d0
      cscax_ck=0.d0
      cscay_ck=0.d0
      cbakx_ck=0.d0
      cbaky_ck=0.d0
      i=1
      cextxi(i)=0.d0
      cextyi(i)=0.d0
      cabsxi(i)=0.d0
      cabsyi(i)=0.d0
      cscaxi(i)=0.d0
      cscayi(i)=0.d0
      cprxi(i)=0.d0
      cpryi(i)=0.d0
      do i=1,nang2
       i11(i)=0.d0
       i21(i)=0.d0
       i22(i)=0.d0
       i12(i)=0.d0
      enddo
C-----------------------------------------------
C  calculating constants and Gaunt coefficients
C-----------------------------------------------
      n0=nmax0+2
      fnr(0)=0.d0
      do n=1,2*n0
       fnr(n)=dsqrt(dble(n))
      enddo
      bcof(0)=1.d0
      do n=0,n0-1
       bcof(n+1)=fnr(n+n+2)*fnr(n+n+1)*bcof(n)/fnr(n+1)/fnr(n+1)
      enddo
C ---------------------------------------------------------------------
C  the formulation used here for the calculation of Gaunt coefficients
C  can be found in Bruning and Lo, IEEE Trans. Anten. Prop. Ap-19, 378
C  (1971) and Xu, J. Comput. Appl. Math. 85, 53 (1997), J. Comput. Phys.
C  139, 137 (1998)
C ---------------------------------------------------------------------
      call cofsrd(nmax0)
      call cofd0(nmax0)
      call cofnv0(nmax0)
      call gau0(nmax0)
c
```

```
      nram=1
c      call ctotpha3(alpha_PA,beta_PA,gamma_PA,ctpha)
c
      indpol=0
      write(*,107)
      write(*,*) 'Starting Bi-CGSTAB solution process'
      write(*,*) 'Solving for x-pol. inci. state'
3     k=kp
      nimprov=0
      fimprv0=100.d0
      tphac=0
      tpha2=0
      tphasy=0
      do imn=1,nmp
       p0(1,imn)=0.d0
       q0(1,imn)=0.d0
       as(1,imn)=0.d0
       bs(1,imn)=0.d0
      enddo
      if(idsuTMX.gt.0) goto 491
c ---------------------------------------------------------------------
c calculating incident wave expansion coefficients
c ---------------------------------------------------------------------
      i=1
      do 6 n=1,nmax(i)
       imn=n*n+n+1
       temp=0.5d0*fnr(2*n+1)
       p0(i,imn)=aMie(i,n)*temp
       q0(i,imn)=bMie(i,n)*temp
       p0(i,imn-2)=-p0(i,imn)
       q0(i,imn-2)=q0(i,imn)
       if(indpol.gt.1) then
        p0(i,imn)=p0(i,imn)*cin
        q0(i,imn)=q0(i,imn)*cin
        p0(i,imn-2)=p0(i,imn)
        q0(i,imn-2)=-q0(i,imn)
       endif
       as(i,imn)=p0(i,imn)
       bs(i,imn)=q0(i,imn)
       as(i,imn-2)=p0(i,imn-2)
       bs(i,imn-2)=q0(i,imn-2)
6     continue
c ---------------------------------------------------------------------
c begins BI-CGSTAB iteration process to solve the interaction equations
c for the partial interacting scatteing coefficients of an individual
c component particle
c [BI-CGSTAB, see H.A. van der Vorst, SIAM,  J. Sci. Stat. Comput. 13,
c  631 (1992)]
c ---------------------------------------------------------------------
      i=1
      ind(i)=0
      ind2(i)=0
      c0i(i)=0.d0
      do n=1,nmax(i)
       imn=n*n+n+1
```

```
      c0i(i)=c0i(i)+p0(i,imn)*dconjg(p0(i,imn))
      c0i(i)=c0i(i)+q0(i,imn)*dconjg(q0(i,imn))
      c0i(i)=c0i(i)+p0(i,imn-2)*dconjg(p0(i,imn-2))
      c0i(i)=c0i(i)+q0(i,imn-2)*dconjg(q0(i,imn-2))
     enddo
     icase=1
     niter=1
61    call transs_PA_v0(eps_PA,id_uc,k,nLPA0,r00,nmax0,nmax,uvmax,
    +              fint,ek,besj,besy,drot,as,bs,as1,bs1,ind,icase)
     i=1
     c1i(i)=0.d0
     do imn=1,uvmax(i)
      n=dsqrt(dble(imn))
      as1(i,imn)=-aMie(i,n)*as1(i,imn)
      bs1(i,imn)=-bMie(i,n)*bs1(i,imn)
      c1i(i)=c1i(i)+as1(i,imn)*dconjg(as1(i,imn))
      c1i(i)=c1i(i)+bs1(i,imn)*dconjg(bs1(i,imn))
     enddo
     temp=0.d0
     cext0=c1i(i)/c0i(i)
     if(cext0.gt.temp) temp=cext0
     B0=c1i(i)
     if(temp.lt.small) goto 490
     i=1
     do imn=1,uvmax(i)
      asp(i,imn)=as1(i,imn)
      bsp(i,imn)=bs1(i,imn)
      as0(i,imn)=as1(i,imn)
      bs0(i,imn)=bs1(i,imn)
     enddo
     call transs_PA_v0(eps_PA,id_uc,k,nLPA0,r00,nmax0,nmax,uvmax,
    +              fint,ek,besj,besy,drot,asp,bsp,ast,bst,ind,icase)
     A0=0.d0
     i=1
     do imn=1,uvmax(i)
      n=dsqrt(dble(imn))
      ast(i,imn)=aMie(i,n)*ast(i,imn)+asp(i,imn)
      bst(i,imn)=bMie(i,n)*bst(i,imn)+bsp(i,imn)
      A0=A0+dconjg(as0(i,imn))*ast(i,imn)
      A0=A0+dconjg(bs0(i,imn))*bst(i,imn)
     enddo
     if(cdabs(A0).lt.1.d-100) goto 490
     Aj=B0/A0
62    i=1
     do imn=1,uvmax(i)
      asv(i,imn)=asp(i,imn)-Aj*ast(i,imn)
      bsv(i,imn)=bsp(i,imn)-Aj*bst(i,imn)
     enddo
     call transs_PA_v0(eps_PA,id_uc,k,nLPA0,r00,nmax0,nmax,uvmax,
    +              fint,ek,besj,besy,drot,asv,bsv,asc,bsc,ind,icase)
     A2=0.d0
     B2=0.d0
     i=1
     do imn=1,uvmax(i)
      n=dsqrt(dble(imn))
```

```
 asc(i,imn)=aMie(i,n)*asc(i,imn)+asv(i,imn)
 bsc(i,imn)=bMie(i,n)*bsc(i,imn)+bsv(i,imn)
 A2=A2+dconjg(asc(i,imn))*asv(i,imn)
 A2=A2+dconjg(bsc(i,imn))*bsv(i,imn)
 B2=B2+dconjg(asc(i,imn))*asc(i,imn)
 B2=B2+dconjg(bsc(i,imn))*bsc(i,imn)
enddo
if(cdabs(B2).lt.1.d-100) goto 490
Bj=A2/B2
i=1
do imn=1,uvmax(i)
 asp(i,imn)=asv(i,imn)-Bj*asc(i,imn)
 bsp(i,imn)=bsv(i,imn)-Bj*bsc(i,imn)
enddo
i=1
c1i(i)=0.d0
do imn=1,uvmax(i)
 Aj2=Aj*as1(i,imn)+Bj*asv(i,imn)
 Bj2=Aj*bs1(i,imn)+Bj*bsv(i,imn)
 as(i,imn)=as(i,imn)+Aj2
 bs(i,imn)=bs(i,imn)+Bj2
 c1i(i)=c1i(i)+Aj2*dconjg(Aj2)
 c1i(i)=c1i(i)+Bj2*dconjg(Bj2)
enddo
cext0=c0i(1)
cext1=c1i(1)
temp=cext1/cext0
if(temp.lt.small) goto 490
if(niter.gt.MXINT) then
 write(*,*) 'Caution:'
 write(*,*) '*** Maximum iterations exceeded ***'
 write(*,*) '***  Results may be inaccurate  ***'
 goto 490
endif
write(*,112) 'iteration #',niter,temp
B2=0.d0
B2i(1)=0.d0
i=1
do imn=1,uvmax(i)
 B2i(i)=B2i(i)+dconjg(as0(i,imn))*asp(i,imn)
 B2i(i)=B2i(i)+dconjg(bs0(i,imn))*bsp(i,imn)
enddo
B2=B2i(1)
A0=B0*Bj
if(cdabs(A0).lt.1.d-100) goto 490
A0=-Aj*B2/A0
i=1
do imn=1,uvmax(i)
 as1(i,imn)=as1(i,imn)-Bj*ast(i,imn)
 bs1(i,imn)=bs1(i,imn)-Bj*bst(i,imn)
 as1(i,imn)=asp(i,imn)-A0*as1(i,imn)
 bs1(i,imn)=bsp(i,imn)-A0*bs1(i,imn)
enddo
B0=B2i(1)
call transs_PA_v0(eps_PA,id_uc,k,nLPA0,r00,nmax0,nmax,uvmax,
```

```
     +           fint,ek,besj,besy,drot,as1,bs1,ast,bst,ind,icase)
      A0=0.d0
      i=1
      do imn=1,uvmax(i)
       n=dsqrt(dble(imn))
       ast(i,imn)=aMie(i,n)*ast(i,imn)+as1(i,imn)
       bst(i,imn)=bMie(i,n)*bst(i,imn)+bs1(i,imn)
       A0=A0+dconjg(as0(i,imn))*ast(i,imn)
       A0=A0+dconjg(bs0(i,imn))*bst(i,imn)
      enddo
      if(cdabs(A0).lt.1.d-100) goto 490
      Aj=B0/A0
      niter=niter+1
      goto 62
c
490   write(*,*) 'Bi-CGSTAB iterations converged'
491   fileSUamn='scacof_x.dat'
      if(indpol.gt.0) fileSUamn='scacof_y.dat'
      if(idsuTMX.gt.0) then
       write(*,*) 'instead of being solved, interactive scattering'
       write(*,*) 'coefficients of a component particle are from'
       write(*,*) 'the existing file: ',fileSUamn
       goto 26
      endif
      open(17,file=fileSUamn,status='unknown')
      write(17,*) nmax(1)
      do 9 j=1,uvmax(1)
       n=dsqrt(dble(j))
       m=j-n*n-n
       write(17,113) m,n,dble(as(1,j)),dimag(as(1,j)),
     +             dble(bs(1,j)),dimag(bs(1,j))
9     continue
      close(17)
      goto 36
26    open(27,file=fileSUamn,status='old')
      read(27,*,err=27) nmax(1)
      uvmax(1)=nmax(1)*(nmax(1)+2)
      do j=1,uvmax(1)
       read(27,*,err=27) m,n,smue(1,1),smue(1,2),smue(1,3),smue(1,4)
       as(1,j)=dcmplx(smue(1,1),smue(1,2))
       bs(1,j)=dcmplx(smue(1,3),smue(1,4))
      enddo
      close(27)
      goto 36
27    write(*,*) 'Error found in the input scattering-coefficient file'
      stop
C =====================================================================
C completed the solution process of interactive scattering coefficients
C for an individual scattering unit
C
C begins the calculation of amplitude scattering and Mueller matrices,
C differential and total cross sections for extinction, absorption,
C scattering, radiation pressure, ...
C
C preparing first the detailed spatial distribution of all component
```

```
C paricle-centers when their initial positions are user inputs, i.e.,
C when IDshape = 0, this needs to be done only once when indpol=0
C ---------------------------------------------------------------------
36    if(indpol.gt.0) goto 39
C ---------------------------------------------------------------------
C when IDshape > 0, the calculation of total phase shifts does not
C require the information on position vectors of individual scattering
C units and needs only the overall dimenesions of a PA (as well as the
C dimensions and center positions of additinal external shapes and/or
C internal openings)
C
C *** ATTENTION: in current implementations, Cpr is calculated via
C                numerical integration techniques
C ---------------------------------------------------------------------
      write(*,*) 'dnLPA before internal openings taken out:'
      write(*,*) dnLPA
      if(IDmoreShapes.eq.0) goto 39
      open(2,file=FLNAME,status='old')
      read(2,*,err=21) n_entry
      do 20 j=1,n_entry
       read(2,*,err=21) idAorS,IDshapeAS,n_open
c ***************************************************************************
c currently working for only whwn IDshape = 2 or 3
c ---------------------------------------------------------------------
       i=1
       if(idimen.eq.2.and.IDshapeAS.gt.10) i=-1
       if(idimen.eq.3.and.IDshapeAS.lt.10) i=-1
       if(i.eq.-1) then
        write(*,*) 'wrong IDshape for the additional structure, which'
        write(*,*) 'is inconsistent with dimension # of the PA'
        write(*,*) 'dimension # and IDshape: ',idimen,IDshapeAS
        write(*,*) 'please check the data file: ',FLNAME
        stop
       endif
       temp=0
       temp1=dx(1)*dx(2)
       if(IDshapeAS.eq.2) then
        read(2,*,err=21) x0,y0
        temp=(n_open)*x0*y0/temp1
       endif
       if(IDshapeAS.eq.3) then
        read(2,*) x0
        temp=0.25d0*(n_open)*pione*x0**2/temp1
       endif
       if(IDshapeAS.eq.22) then
        read(2,*,err=21) x0,y0,z0
        temp=(n_open)*x0*y0*z0/(temp1*dx(3))
       endif
       if(IDshapeAS.eq.23) then
        read(2,*) x0,y0,z0
        temp=0.25d0*(n_open)*pione*x0**2*z0/(temp1*dx(3))
       endif
       i=temp
       if(i.eq.0) then
        write(*,*) 'IDshapeAS must be 2, 3, 22 or 23'
```

```
       goto 21
      endif
      dnLPA=dnLPA+(idAorS)*temp
      do n=1,n_open
       read(2,*,err=21)
      enddo
20    continue
     close(2)
     write(*,*) 'dnLPA after internal openings taken out:'
     write(*,*) dnLPA
     goto 39
21    write(*,*) 'ERROR: please check the format of the input file:'
     write(*,*) FLNAME
     stop
c -----------------------------------------------------------------------
c computing two-dimensional angular distribution of the total scattered
c field in a mesh of (npng+1)*nang2 specified by the input (sang,pang)
c Note that the definition used in GMM for amplitude scattering matrix
c is slightly different from that given by van de Hulst and by Bohren &
c Huffman. The concept of "scattering plane" is no longer used. 10/20/02
c
c At the same time, extinction, scattering and radiation-pressure cross
c sections are calculated via numerical integration
c -----------------------------------------------------------------------
c39    k=k0
39    k=kp
     scx=0.5d0*dx(1)*k
     scy=0.5d0*dx(2)*k
     scz=0.5d0*dx(3)*k
     totsca_ick0=0
     totasy_ick0=0
     totsca_ick=0
     totasy_ick=0
     totext_ick0=0
     totext_ick=0
     if(indpol.lt.1) then
      cextxi(1)=0.d0
      cabsxi(1)=0.d0
      cscaxi(1)=0.d0
      cprxi(1)=0.d0
     else
      cextyi(1)=0.d0
      cabsyi(1)=0.d0
      cscayi(1)=0.d0
      cpryi(1)=0.d0
     endif
c
c preparing quantities for the calculation of the cross sections for
c scattering and radiation pressure
c
     do m=1,2
      do n=1,np
       do j=1,nmp
        pmnuv(m,n,j)=0
        qmnuv(m,n,j)=0
```

```
        enddo
       enddo
      enddo
      do n=1,nmax(1)
       do iuv=1,uvmax(1)
c for m=1
        A2=fnr(2*n+1)*(as(1,iuv)+bs(1,iuv))
        pmnuv(1,n,iuv)=A2
        qmnuv(1,n,iuv)=A2
c for m=-1
            B2=-fnr(2*n+1)*(as(1,iuv)-bs(1,iuv))
        pmnuv(2,n,iuv)=B2
        qmnuv(2,n,iuv)=-B2
        if(indpol.gt.0) then
         pmnuv(1,n,iuv)=pmnuv(1,n,iuv)*ci
         qmnuv(1,n,iuv)=qmnuv(1,n,iuv)*ci
         pmnuv(2,n,iuv)=pmnuv(2,n,iuv)*cin
         qmnuv(2,n,iuv)=qmnuv(2,n,iuv)*cin
        endif
       enddo
      enddo
      do i=1,nmp
       do j=1,nmp
        amnuv(i,j)=0
        bmnuv(i,j)=0
       enddo
      enddo
      do imn=1,uvmax(1)
       do iuv=1,uvmax(1)
        A2=dconjg(as(1,imn))*as(1,iuv)+dconjg(bs(1,imn))*bs(1,iuv)
        amnuv(imn,iuv)=A2
        B2=dconjg(as(1,imn))*bs(1,iuv)+dconjg(bs(1,imn))*as(1,iuv)
        bmnuv(imn,iuv)=B2
       enddo
      enddo
c ---------------------------------------------------------------------
c calculate amplitude matrix for a mesh of theta and phi and obtain at
c the same time the cross sections for scattering and radiation pressure
c ---------------------------------------------------------------------
      do i=1,nang
       iang=2*nang-i
c iang for theta in [90,180] degs
       dang(i)=sang*dble(i-1)
       dang(iang)=180.0d0-dang(i)
       theta=dang(i)*pione/180.0d0
       xt=dcos(theta)
       st=dsin(theta)
       if(i.eq.1) then
        xt=1
        st=0
       endif
       if(i.eq.nang) then
        xt=0
        st=1
       endif
```

```
      call tipitaud(nmax0,xt)
c ---------------------------------------------------------------------
c calculations for extinction & scattering cross-section as well as
c asymmetry parameter
c ---------------------------------------------------------------------
      do u=-np,np
       do j=1,2
        do m=1,2
         extmu(m,u,j)=0
        enddo
        do m=-np,np
         scamu(m,u,j)=0
        enddo
       enddo
      enddo
      tempm=-1
      do m=0,nmax(1)
       tempm=-tempm
       n0=m
       if(m.eq.0) n0=1
       tempu=-1
       do u=0,nmax(1)
        tempu=-tempu
        v0=u
        if(u.eq.0) v0=1
        do n=n0,nmax(1)
         imn=n*n+n+m
         imn2=imn-2*m
         jmn=(n-1)*(n+2)/2+m+1
c imn - index for scattering coefficients
c jmn - index for function tau
         do 41 v=v0,nmax(1)
          temp0=(-1)**(m+n+u+v)
          iuv=v*v+v+u
          iuv2=iuv-2*u
          juv=(v-1)*(v+2)/2+u+1
          tau11=tau(jmn)*tau(juv)
          tau22=pi(jmn)*pi(juv)
          tau12=tau(jmn)*pi(juv)
          tau21=pi(jmn)*tau(juv)
          tau1122p=tau11+tau22
          tau1221p=tau12+tau21
          tau1122n=tau11-tau22
          tau1221n=tau12-tau21
c calculate the case of m,u for theta
          A2=tau1122p*amnuv(imn,iuv)
          B2=tau1221p*bmnuv(imn,iuv)
          scamu(m,u,1)=scamu(m,u,1)+A2+B2
          if(m.eq.1) then
           Au=tau1122p*pmnuv(1,n,iuv)
           Bu=tau1221p*qmnuv(1,n,iuv)
           extmu(1,u,1)=extmu(1,u,1)+Au+Bu
          endif
c calculate the case of m,u for pi-theta
c when i=iang, theta would be pi/2 and pi-pi/2, which are the same
```

```
          if(i.ne.iang) then
           scamu(m,u,2)=scamu(m,u,2)+temp0*(A2-B2)
               if(m.eq.1) then
            extmu(1,u,2)=extmu(1,u,2)+temp0*(Au-Bu)
           endif
          endif
          if(u.eq.0) goto 40
c calculate the cases of m,-u
          A2=tempu*tau1122n*amnuv(imn,iuv2)
          B2=tempu*tau1221n*bmnuv(imn,iuv2)
          scamu(m,-u,1)=scamu(m,-u,1)+A2-B2
          if(m.eq.1) then
           Au=tempu*tau1122n*pmnuv(1,n,iuv2)
           Bu=tempu*tau1221n*qmnuv(1,n,iuv2)
           extmu(1,-u,1)=extmu(1,-u,1)+Au-Bu
          endif
          if(i.ne.iang) then
           scamu(m,-u,2)=scamu(m,-u,2)+temp0*(A2+B2)
           if(m.eq.1) then
            extmu(1,-u,2)=extmu(1,-u,2)+temp0*(Au+Bu)
           endif
          endif
c calculate the case of -m,u
40         if(m.eq.0) goto 41
          A2=tempm*tau1122n*amnuv(imn2,iuv)
          B2=tempm*tau1221n*bmnuv(imn2,iuv)
          scamu(-m,u,1)=scamu(-m,u,1)+A2+B2
          if(m.eq.1) then
           Au=tempm*tau1122n*pmnuv(2,n,iuv)
           Bu=tempm*tau1221n*qmnuv(2,n,iuv)
           extmu(2,u,1)=extmu(2,u,1)+Au+Bu
          endif
          if(i.ne.iang) then
           scamu(-m,u,2)=scamu(-m,u,2)+temp0*(A2-B2)
               if(m.eq.1) then
            extmu(2,u,2)=extmu(2,u,2)+temp0*(Au-Bu)
           endif
          endif
          if(u.eq.0) goto 41
c calculate the case of -m,-u
          temp=tempm*tempu
          A2=temp*tau1122p*amnuv(imn2,iuv2)
          B2=temp*tau1221p*bmnuv(imn2,iuv2)
          scamu(-m,-u,1)=scamu(-m,-u,1)+A2-B2
          if(m.eq.1) then
           Au=temp*tau1122p*pmnuv(2,n,iuv2)
           Bu=temp*tau1221p*qmnuv(2,n,iuv2)
           extmu(2,-u,1)=extmu(2,-u,1)+Au-Bu
          endif
          if(i.ne.iang) then
           scamu(-m,-u,2)=scamu(-m,-u,2)+temp0*(A2+B2)
               if(m.eq.1) then
            extmu(2,-u,2)=extmu(2,-u,2)+temp0*(Au+Bu)
           endif
          endif
```

```
41        enddo
         enddo
        enddo
       enddo
c ---------------------------------------------------------------------
c calculating amplitude matrix elements
c ---------------------------------------------------------------------
       do jc=1,npng
        azphi=jc-1
        azphi=azphi*pang*pih/90.0d0
        sphi=dsin(azphi)
        cphi=dcos(azphi)
c ---------------------------------------------------------------------
c A and B are phase shifts:
c    A for theta in [0,90] degs and B for theta in [90,180]
c ---------------------------------------------------------------------
        A=0
        B=0
        if(IDshape.gt.0) goto 301
c ---------------------------------------------------------------------
c ALWAYS BEAR IN MIND that the set of input configuration data is for
c an PA at its initial orientation before the rotation of the refrence
c frame in terms of the Euler-angle triad
c However, when IDshape=0, the PA has already been rotated as specified
C (see above the subroutine call of c rotatEuler3) and all the component
C particle-center position vectors ("r00") used in the following refer
C to the actual PA orientation
c ---------------------------------------------------------------------
        do 31 j=1,nLPA
         x1=r00(1,j)
         y1=r00(2,j)
         z1=r00(3,j)
c         x1=r00(1,id_uc(j))
c         y1=r00(2,id_uc(j))
c         z1=r00(3,id_uc(j))
c         cz=dcos(k*z1)
c         sz=dsin(k*z1)
c         cmz=dcmplx(cz,sz)
c ---------------------------------------------------------------------
c the above cmz is the incident phase term
c ---------------------------------------------------------------------
c         sb=x1*cphi+y1*sphi
c         sb=sb*st
c         cb=z1*xt
c         cz=k*(sb+cb)
c         sz=k*(sb-cb)
c         A0=dcmplx(dcos(cz),-dsin(cz))
c         B0=dcmplx(dcos(sz),-dsin(sz))
c ---------------------------------------------------------------------
c the above A0 and B0 are the scattered phase terms for the direction of
c (theta,phi) and (pi-theta,phi), respectively
c ---------------------------------------------------------------------
c         A=A+A0*cmz
c         B=B+B0*cmz
         sb=(x1*cphi+y1*sphi)*st
```

```
        cb=z1*(1.d0-xt)
        cz=k*(-sb+cb)
        A0=dcmplx(dcos(cz),dsin(cz))
        A=A+A0
        cb=z1*(1.d0+xt)
        sz=k*(-sb+cb)
        B0=dcmplx(dcos(sz),dsin(sz))
        B=B+B0
31      continue
       goto 360
301     if(IDshape.eq.2.or.IDshape.eq.22) goto 302
       call phaseEllipsiod(xt,st,cphi,sphi,k,idimen,IDshape,wx,dx,
     +              ctpha,A,B,FLNAME,IDmoreShapes)
       goto 360
302     call totphase(xt,st,cphi,sphi,k,idimen,IDshape,wx,dx,ctpha,
     +              A,B,FLNAME,IDmoreShapes)
360     if(i.eq.1.and.jc.eq.1) then
        write(*,*) 'sca. ang: ',dang(i)
        write(*,*) '  A: ',A
            write(*,*) '  B: ',B
        write(*,*) 'n_entry: ',n_entry
        phabak=B
       endif
       if(i.eq.nang.and.jc.eq.1) then
        write(*,*) 'sca. ang: ',dang(i)
        write(*,*) '  A: ',A
        write(*,*) '  B: ',B
       endif
c ---------------------------------------------------------------------
c adding the differential contribution from the specified theta-phi bin
c to the integral kernel for the extinction cross sections (Note: these
c calculations do not include the constant 0.25/pi)
c ---------------------------------------------------------------------
       Au=0
       Bu=0
       do u=-nmax(1),nmax(1)
        p=u-1
            p=p*azphi
        t=dsin(p)
        p=dcos(p)
        Aj=dcmplx(p,t)
c Au for theta and Bu for pi-theta
        Au=Au+extmu(1,u,1)*Aj
        Bu=Bu+extmu(1,u,2)*Aj
        p=u+1
            p=p*azphi
        t=dsin(p)
        p=dcos(p)
        Aj=dcmplx(p,t)
c Au for theta and Bu for pi-theta
        Au=Au+extmu(2,u,1)*Aj
        Bu=Bu+extmu(2,u,2)*Aj
       enddo
       A0=A
       B0=B
```

```
c       if(i.eq.1) then
c        A0=0.5d0*A0
c        B0=0.5d0*B0
c       endif
        temp=A0*Au+B0*Bu
        totext_ick0=totext_ick0+temp*st
        tphac=tphac+(A0+B0)*st
c       temp=1.d0
c       if(i.eq.1) temp=0.5d0
c       temp=temp*(Au+Bu)
        temp=Au+Bu
        totext_ick=totext_ick+temp*st
c ---------------------------------------------------------------------
c adding the differential contribution from the specified theta-phi bin
c to the integral kernel for the scattering and radiation-pressure cross
c sections (Note: these calculations do not include constant 0.25/pi)
c ---------------------------------------------------------------------
        Au=0
        Bu=0
        do m=-nmax(1),nmax(1)
         do u=-nmax(1),nmax(1)
          p=u-m
              p=p*azphi
          t=dsin(p)
          p=dcos(p)
          Aj=dcmplx(p,t)
c Au for theta and Bu for pi-theta
          Au=Au+scamu(m,u,1)*Aj
          Bu=Bu+scamu(m,u,2)*Aj
         enddo
        enddo
        temp1=dconjg(A)*A
        temp2=dconjg(B)*B
c       if(i.eq.1) then
c        temp1=0.5d0*temp1
c        temp2=0.5d0*temp2
c       endif
        temp=temp1*Au+temp2*Bu
c
c       A2=dconjg(A)
c       B2=dconjg(B)
c       if(i.eq.1) then
c        A2=0.5d0*A2
c        B2=0.5d0*B2
c       endif
c       temp=A2*Au+B2*Bu
c
           totsca_ick0=totsca_ick0+temp*st
        tpha2=tpha2+(temp1+temp2)*st
c Remember the sign difference for cos(theta) and cos(pi-theta)
c       temp=A2*Au-B2*Bu
        temp=temp1*Au-temp2*Bu
        totasy_ick0=totasy_ick0+temp*st*xt
        tphasy=tphasy+(temp1-temp2)*st*xt
c ---------------------------------------------------------------------
```

```
c calculate total scattering coefficients of entire PA for the specific
c direction of (theta,phi), which are required in the calculation of the
c amplitude matrix elements
c ---------------------------------------------------------------------
      do imn=1,nmp
       at(imn)=0.d0
       bt(imn)=0.d0
       atj(imn)=0.d0
       btj(imn)=0.d0
      enddo
c ---------------------------------------------------------------------
c at & bt for theta in [0,90] degs; atj & btj for theta in [90,180];
c ---------------------------------------------------------------------
      do 361 imn=1,uvmax(1)
       n=dsqrt(dble(imn))
       if(n.gt.nmax(1)) goto 361
       at(imn)=at(imn)+A*as(1,imn)
       bt(imn)=bt(imn)+A*bs(1,imn)
       atj(imn)=atj(imn)+B*as(1,imn)
       btj(imn)=btj(imn)+B*bs(1,imn)
361     continue
c ---------------------------------------------------------------------
c calculates the amplitude matrix elements for the specified direction
c of (theta,phi)
c ---------------------------------------------------------------------
      if(indpol.lt.1) then
       s2x(jc,i)=0.d0
       s4x(jc,i)=0.d0
       s2x(jc,iang)=0.d0
       s4x(jc,iang)=0.d0
      else
       s3y(jc,i)=0.d0
       s1y(jc,i)=0.d0
       s3y(jc,iang)=0.d0
       s1y(jc,iang)=0.d0
      endif
      A=0.d0
      B=0.d0
      Aj=0.d0
      Bj=0.d0
c adding m=0 terms for theta and 180deg-theta separately
      do j=1,nmax0
       imn=(j-1)*(j+2)/2+1
       u=j*j+j
       A=A+at(u)*tau(imn)
       B=B+bt(u)*tau(imn)
       if(i.ne.iang) then
        t=(-1)**(j+1)
        Aj=Aj+atj(u)*tau(imn)*t
        Bj=Bj+btj(u)*tau(imn)*t
       endif
      enddo
      if(indpol.lt.1) then
       s2x(jc,i)=s2x(jc,i)+A
       s4x(jc,i)=s4x(jc,i)+B*dcmplx(0.d0,-1.d0)
```

```
        if(i.ne.iang) then
         s2x(jc,iang)=s2x(jc,iang)+Aj
         s4x(jc,iang)=s4x(jc,iang)+Bj*dcmplx(0.d0,-1.d0)
        endif
       else
        s3y(jc,i)=s3y(jc,i)-A
        s1y(jc,i)=s1y(jc,i)+B*dcmplx(0.d0,1.d0)
        if(i.ne.iang) then
         s3y(jc,iang)=s3y(jc,iang)-Aj
         s1y(jc,iang)=s1y(jc,iang)+Bj*dcmplx(0.d0,1.d0)
        endif
       endif
c adding nonzero m terms for both m and -m and both theta and pi-theta
       rm=1.d0
       do 362 m=1,nmax0
        A=0.d0
        B=0.d0
        A2=0.d0
        B2=0.d0
        Aj=0.d0
        Bj=0.d0
        Aj2=0.d0
        Bj2=0.d0
        rm=-rm
        do 363 j=m,nmax0
         imn=(j-1)*(j+2)/2+m+1
         u=j*j+j+m
         v=u-2*m
         A0=at(u)*tau(imn)+bt(u)*pi(imn)
         B0=rm*(at(v)*tau(imn)-bt(v)*pi(imn))
         A=A+A0+B0
         A2=A2+A0-B0
         A0=at(u)*pi(imn)+bt(u)*tau(imn)
         B0=rm*(at(v)*pi(imn)-bt(v)*tau(imn))
         B=B+A0-B0
         B2=B2+A0+B0
         if(i.ne.iang) then
          t=(-1)**(j+m+1)
          p=-t
          A0=atj(u)*tau(imn)*t+btj(u)*pi(imn)*p
          B0=rm*(atj(v)*tau(imn)*t-btj(v)*pi(imn)*p)
          Aj=Aj+A0+B0
          Aj2=Aj2+A0-B0
          A0=atj(u)*pi(imn)*p+btj(u)*tau(imn)*t
          B0=rm*(atj(v)*pi(imn)*p-btj(v)*tau(imn)*t)
          Bj=Bj+A0-B0
          Bj2=Bj2+A0+B0
         endif
363      continue
        temp=dble(m)*azphi
        sb=dsin(temp)
        cb=dcos(temp)
        if(indpol.lt.1) then
         s2x(jc,i)=s2x(jc,i)+A2*dcmplx(0.d0,1.d0)*sb+A*cb
         s4x(jc,i)=s4x(jc,i)+B*dcmplx(0.d0,-1.d0)*cb+B2*sb
```

```
        if(i.ne.iang) then
         s2x(jc,iang)=s2x(jc,iang)+Aj*cb
     +               +Aj2*dcmplx(0.d0,1.d0)*sb
         s4x(jc,iang)=s4x(jc,iang)+Bj2*sb
     +               +Bj*dcmplx(0.d0,-1.d0)*cb
        endif
       else
        s3y(jc,i)=s3y(jc,i)+A2*dcmplx(0.d0,-1.d0)*sb-A*cb
        s1y(jc,i)=s1y(jc,i)-B2*sb+B*dcmplx(0.d0,1.d0)*cb
        if(i.ne.iang) then
         s3y(jc,iang)=s3y(jc,iang)-Aj*cb
     +               +Aj2*dcmplx(0.d0,-1.d0)*sb
         s1y(jc,iang)=s1y(jc,iang)-Bj2*sb
     +               +Bj*dcmplx(0.d0,1.d0)*cb
        endif
       endif
362     continue
      if(indpol.lt.1) then
       temp=cdabs(s2x(jc,i))**2
       temp=temp+cdabs(s4x(jc,i))**2
       temp=temp*st
       totsca_ick=totsca_ick+temp
           totasy_ick=totasy_ick+temp*xt
       if(i.ne.iang) then
        temp=cdabs(s2x(jc,iang))**2
        temp=temp+cdabs(s4x(jc,iang))**2
        temp=temp*st
        totsca_ick=totsca_ick+temp
            totasy_ick=totasy_ick-temp*xt
       endif
      else
       temp=cdabs(s3y(jc,i))**2
       temp=temp+cdabs(s1y(jc,i))**2
       temp=temp*st
       totsca_ick=totsca_ick+temp
           totasy_ick=totasy_ick+temp*xt
       if(i.ne.iang) then
        temp=cdabs(s3y(jc,iang))**2
        temp=temp+cdabs(s1y(jc,iang))**2
        temp=temp*st
        totsca_ick=totsca_ick+temp
            totasy_ick=totasy_ick-temp*xt
       endif
      endif
     enddo    ! end of jc
    enddo     ! end of i
    totext_ick=totext_ick*dnLPA

    p=sang*pih/90.d0
    t=pang*pih/90.d0
    temp=1.d0/k**2*p*t
    temp1=0.5d0*temp
    write(*,*) 'solution accuracy (test 1):'
    write(*,*) 'Cext: ',totext_ick0*temp1,totext_ick*temp1
    write(*,*) 'Csca: ',totsca_ick0*temp,totsca_ick*temp
```

```
      write(*,*) 'asym: ',totasy_ick0/totsca_ick0,totasy_ick/totsca_ick
c -------------------------------------------------------------------
c calculate total scattering and radiation-pressure cross sections using
c the quantities obtained above as the results of numerical integrations
c -------------------------------------------------------------------
      temp1=4.d0*pione/k**2
      if(indpol.lt.1) then
       tempext=s2x(1,1)*temp1
      else
       tempext=s1y(1,1)*temp1
      endif
      write(*,*) 'Cext: ',tempext
c
c      write(*,*) 'int_pha_ext: ',tphac,totext_ick0
c      write(*,*) 'int_pha_sca: ',tpha2,totsca_ick0
c      write(*,*) 'int_pha_asy: ',tphasy,totasy_ick0
c      totsca_ick=totsca_ick0
c      totasy_ick=totasy_ick0
c
c      p=sang*pih/90.d0
c      t=pang*pih/90.d0
c
c      temp=dnLPA/k**2*p*t
c      temp=1.d0/k**2*p*t
c -------------------------------------------------------------------
c NOTE the constant factor 1/k**2 used here, which results from the two
c constants: one is 1/(4*pi) applied to the numerical integation and the
c other is 4*pi/k**2 for summing over differential cross sections
c p,t - the bin sizes for theta and phi
c -------------------------------------------------------------------
      if(indpol.lt.1) then
       assymx_ick=totasy_ick/totsca_ick
       cscax_ick=totsca_ick*temp
       cextx_ick0=0.5d0*totext_ick0*temp
c       cextx_ick=0.5d0*totext_ick*temp*dnLPA
       cextx_ick=0.5d0*totext_ick*temp
       cabsx_ick=cextx_ick-cscax_ick
      else
       assymy_ick=totasy_ick/totsca_ick
       cscay_ick=totsca_ick*temp
       cexty_ick0=0.5d0*totext_ick0*temp
c       cexty_ick=0.5d0*totext_ick*temp*dnLPA
       cexty_ick=0.5d0*totext_ick*temp
       cabsy_ick=cexty_ick-cscay_ick
      endif
c -------------------------------------------------------------------
c computing extinction and absorption cross-sections of an individual
c component particle [see Xu, Appl. Opt. 36, 9496 (1997) for the
c formulations used here]
c -------------------------------------------------------------------
C     if(IDshape.ge.1.and.IDshape.le.6) then
C       dnLPA=nx(1)
C       temp=nx1(1)
C       do i=2,3
C          dnLPA=dnLPA*(nx(i))
```

```
C           temp=temp*(nx1(i))
C         enddo
C         dnLPA=dnLPA-temp
C       else
C         dnLPA=nLPA
C       endif
        A=0.d0
        B=0.d0
        do n=1,nmax(1)
         rn=fnr(2*n+1)
         m0=n*n+n+1
         u0=n*n+n-1
         A=A+rn*(as(1,m0)+bs(1,m0))
         B=B-rn*(as(1,u0)-bs(1,u0))
        enddo
        if(indpol.lt.1) then
         cextxi(1)=dble(A+B)
        else
         cextyi(1)=-dimag(A-B)
        endif
        if(indpol.lt.1) then
         cextx=twopi*cextxi(1)*dnLPA/k**2
        else
         cexty=twopi*cextyi(1)*dnLPA/k**2
        endif
c
        j=1
        do n=1,nmax(j)
         A=ref(j)*dcmplx(rsr(n,j),-rsi(n,j))
         temp1=-dimag(A)
         A=px(n,j)*(ref(j)*rsx(n,j)-dcmplx(rsr(n,j),rsi(n,j)))
         temp=cdabs(A)*cdabs(A)
         if(temp.eq.0.d0) then
          dn=0.d0
         else
          dn=temp1/temp
         endif
         A=dcmplx(r0(5,j),-r0(6,j))*dcmplx(rsr(n,j),-rsi(n,j))
         temp1=-dimag(A)
         A=px(n,j)*(rsx(n,j)-ref(j)*dcmplx(rsr(n,j),rsi(n,j)))
         temp=cdabs(A)*cdabs(A)
         if(temp.eq.0.d0) then
          cn=0.d0
         else
          cn=temp1/temp
         endif
         do m=-n,n
          i=n*n+n+m
          temp1=dn*cdabs(as(j,i))*cdabs(as(j,i))
     +          +cn*cdabs(bs(j,i))*cdabs(bs(j,i))
          if(indpol.lt.1) then
           cabsxi(j)=cabsxi(j)+temp1
          else
           cabsyi(j)=cabsyi(j)+temp1
          endif
```

```
       enddo
      enddo
      if(indpol.lt.1) then
       cabsx=cabsxi(1)*dnLPA
       cabsx=2.0d0*twopi*cabsx/k**2
       cextx_ck=cscax_ick+cabsx
       write(*,*) ' cextx_ick0,cextx_ick: ',cextx_ick0,cextx_ick
       write(*,*) ' cextx_ck,cextx:       ',cextx_ck,cextx
       write(*,*) ' cabsx_ick,cabsx: ',cabsx_ick,cabsx
       cscax=cextx-cabsx
       write(*,*) ' cscax_ick,cscax: ',cscax_ick,cscax
      else
       cabsy=cabsyi(1)*dnLPA
       cabsy=2.0d0*twopi*cabsy/k**2
       cexty_ck=cscay_ick+cabsy
       write(*,*) ' cexty_ick0,cexty_ick: ',cexty_ick0,cexty_ick
       write(*,*) ' cexty_ck,cexty:       ',cexty_ck,cexty
       write(*,*) ' cabsy_ick,cabsy: ',cabsy_ick,cabsy
       cscay=cexty-cabsy
       write(*,*) ' cscay_ick,cscay: ',cscay_ick,cscay
      endif
c ---------------------------------------------------------------------
c computing back-scattering cross-sections in the exact backward
c direction [see, for example, Xu, Appl. Opt. 36, 9496 (1997)]
c ---------------------------------------------------------------------
      if(IDshape.ne.0) goto 55
      cmz=0.d0
      do j=1,nLPA
        x1=r00(1,j)
        y1=r00(2,j)
        z1=r00(3,j)
c         x1=x1-x0
c         y1=y1-y0
c         z1=z1-z0
c         if(j.gt.1) then
c           if(x1.eq.0.d0.and.y1.eq.0.d0.and.z1.eq.0.d0) then
c             write(*,*) 'ERROR: input particle centers overlapped'
c              write(*,*) j,id_uc(j),id_uc(1)
c              stop
c           endif
c         endif
        z1=2.d0*z1
        cz=dcos(k*z1)
        sz=dsin(k*z1)
        cmz=cmz+dcmplx(cz,sz)
      enddo
      A=0.d0
      B=0.d0
      do n=1,nmax(1)
        temp=(-1)**n
        rn=temp*fnr(2*n+1)
c         if(n/2*2.eq.n) rn=-rn
        m0=n*n+n+1
        u0=n*n+n-1
c         A=A-rn*(as(1,m0)-bs(1,m0))
```

```
        A=A+rn*(as(1,m0)-bs(1,m0))
        B=B-rn*(as(1,u0)+bs(1,u0))
      enddo
      write(*,*) 'cmz for bak: ',cmz
      write(*,*) 'A for bak: ',A
      write(*,*) 'B for bak: ',B
      rn=pione/k**2
      if(indpol.lt.1) then
        Aj=(A+B)*cmz
        cbakx0=rn*cdabs(Aj)**2
        write(*,*) ' Cbakx0: ',cbakx0
      else
        Aj=(-A+B)*cmz*ci
        cbaky0=rn*cdabs(Aj)**2
        write(*,*) ' Cbaky0: ',cbaky0
      endif
c -----------------------------------------------------------------------
c  computing heat-source functions (i.e., the three-dimensional internal
c  electric field distributions) for each of the component spheres [see
c  Xu et al. Physical Review E 60 (1999)]
c  in spherical coordinates when idphoto=1 or in Cartesian coordinates
c  when idphoto=2
c  idphoto=0 for not calculating
c -----------------------------------------------------------------------
55    A=0.d0
      B=0.d0
      do n=1,nmax(1)
        rn=-fnr(2*n+1)
        if(n/2*2.eq.n) rn=-rn
        m0=n*n+n+1
        u0=n*n+n-1
c         A=A-rn*(at(m0)-bt(m0))
c         B=B+rn*(at(u0)+bt(u0))
c         A=A-rn*(as(1,m0)-bs(1,m0))
        A=A+rn*(as(1,m0)-bs(1,m0))
        B=B-rn*(as(1,u0)+bs(1,u0))
      enddo
      write(*,*) 'phabak: ',phabak
      write(*,*) 'A for bak: ',A
      write(*,*) 'B for bak: ',B
c      A=A*phabak
c      B=B*phabak
      rn=pione/k**2
      if(indpol.lt.1) then
       Aj=(A+B)*phabak
       cbakx_ck=rn*cdabs(Aj)**2
       temp1=cextx/cextx_ck
       temp2=cscax_ick/cscax
       if(temp2.gt.1.05d0) then
          idcorrf=1
          corrf=temp2**2
          temp2=1.d0/corrf
       endif
       cextx=cextx*temp1
       cscax=cscax*temp1
```

```
      temp=cabsx*temp1
      cabsx=cextx-cscax
      assymx=assymx_ick*temp2
c
c       temp=cextx_ick0/cextx_ick
c       tempsca=cextx_ick0-cscax_ick
c       if(tempsca.lt.-0.01d0.and.temp2.gt.1.05d0) then
c         if(temp.lt.0.95d0) then
c            assymx=assymx_ick*temp/temp2
c         endif
c       endif
c
c       cbakx=cbakx_ck/temp2
      cbakx=cbakx_ck*temp1
      cbakx=cbakx_ck
c       if(cabsx_ick.gt.cabsx) then
c        cbakx=cbakx_ck*cscax/cscax_ick
c        assymx=assymx_ick*cscax/cscax_ick
c       else
c        cbakx=cbakx_ck*cscax_ick/cscax
c        assymx=assymx_ick*cscax_ick/cscax
c       endif
c       if(cscax_ick.gt.cscax) assymx=assymx_ick*cscax/cscax_ick
      write(*,*) ' corr. fac.:: ',temp1,temp2
      write(*,*) ' cbakx_ck,cbakx: ',cbakx_ck,cbakx
      if(nimprov.eq.0)
     +    write(*,*) ' assymx_ick,assymx: ',assymx_ick,assymx
      write(*,*) ' cabsx,cabsx_ck: ',cabsx,temp
      write(*,*) ' cextx: ',cextx
      write(*,*) ' cscax: ',cscax
      write(*,*) ' '
     else
      Aj=(-A+B)*phabak*ci
      cbaky_ck=rn*cdabs(Aj)**2
      temp1=cexty/cexty_ck
      temp2=cscay_ick/cscay
      if(temp2.gt.1.05d0) then
        idcorrf=1
        corrf=temp2**2
        temp2=1.d0/corrf
      endif
      cexty=cexty*temp1
      cscay=cscay*temp1
      temp=cabsy*temp1
      cabsy=cexty-cscay
      assymy=assymy_ick*temp2
c
c       temp=cexty_ick0/cexty_ick
c       tempsca=cexty_ick0-cscay_ick
c       if(tempsca.lt.-0.01d0.and.temp2.gt.1.05d0) then
c         if(temp.lt.0.95d0) then
c            assymy=assymy_ick*temp/temp2
c         endif
c       endif
c
```

```
c       cbaky=cbaky_ck/temp2
        cbaky=cbaky_ck*temp1
        cbaky=cbaky_ck
c       if(cabsy_ick.gt.cabsy) then
c        cbaky=cbaky_ck*cscay/cscay_ick
c        assymy=assymy_ick*cscay/cscay_ick
c       else
c        cbaky=cbaky_ck*cscay_ick/cscay
c        assymy=assymy_ick*cscay_ick/cscay
c       endif
c       if(cscay_ick.gt.cscay) assymy=assymy_ick*cscay/cscay_ick
        write(*,*) ' corr. fac.:: ',temp1,temp2
        write(*,*) ' cbaky_ck,cbaky: ',cbaky_ck,cbaky
        if(nimprov.eq.0)
     +    write(*,*) ' assymy_ick,assymy: ',assymy_ick,assymy
        write(*,*) ' cabsy,cabsy_ck: ',cabsy,temp
        write(*,*) ' cexty: ',cexty
        write(*,*) ' cscay: ',cscay
        write(*,*) ' '
       endif
       if(nimprov.eq.0) then
        do imn=1,uvmax(1)
         asv(1,imn)=as(1,imn)
         bsv(1,imn)=bs(1,imn)
        enddo
        if(indpol.lt.1) then
         assymx0=assymx
         cbakx00=cbakx
c        cabsx_ick0=cabsx_ick
c        cabsx0=cabsx
        else
         assymy0=assymy
         cbaky00=cbaky
c        cabsy_ick0=cabsy_ick
c        cabsy0=cabsy
        endif
       endif

       nimprov=nimprov+1
       tempsca=temp1
       if(tempsca.lt.0.d0) then
        write(*,*) 'the scattering solutions are invalid'
        write(*,*) 'please ckeck input data'
        stop
       endif
cxu       tempsca=dsqrt(tempsca)

c      tempext=dsqrt(tempext)
       temp0=dabs(tempsca-1.d0)
c      temp1=tempext/tempsca
cx       if(nimprov.eq.1) then
cx        write(*,*) '# of iteration for improve.: ',nimprov
cx        write(*,*) ' corr. factor for: ',tempsca
cx        do n=1,nmax(1)
cx         do m=-n,n
```

```
cx          imn=n*n+n+m
cxcxu          if(m.ne.1.and.m.ne.-1) then
cxc           temp=as(1,imn)
cxc           temp=temp/tempext**2
cxc           temp2=dcmplx(0.d0,-1.d0)*as(1,imn)
cxc           as(1,imn)=dcmplx(temp,temp2)
cxc           temp=bs(1,imn)
cxc           temp=temp/tempext**2
cxc           temp2=dcmplx(0.d0,-1.d0)*bs(1,imn)
cxc           bs(1,imn)=dcmplx(temp,temp2)
cxc           as(1,imn)=as(1,imn)*tempext
cxc           bs(1,imn)=bs(1,imn)*tempext
cxc          else
cx           as(1,imn)=as(1,imn)*tempsca
cx           bs(1,imn)=bs(1,imn)*tempsca
cxcxu          else
cxcxu           as(1,imn)=as(1,imn)*tempext
cxcxu           bs(1,imn)=bs(1,imn)*tempext
cxcxu          endif
cxcxu          if(m.eq.1.or.m.eq.-1) then
cxcxu           as(1,imn)=as(1,imn)*tempext
cxcxu           bs(1,imn)=bs(1,imn)*tempext
cxcxu          endif
cx         enddo
cx        enddo
cx        fimprv0=temp0
cx        nimprov=nimprov+1
cx        goto 39
cxcxu       endif
cx      endif
c
cx      if(nimprov.gt.0) then
cx        if(indpol.lt.1) then
cx           assymx=assymx0
cx           if(idcorrf.gt.0) then
cx              cextx=cextx*corrf
cx              cscax=cscax*corrf
cx              cabsx=cextx-cscax
cx              idcorrf=0
cx           endif
cx        else
cx           assymy=assymy0
cx           if(idcorrf.gt.0) then
cx              cexty=cexty*corrf
cx              cscay=cscay*corrf
cx              cabsy=cexty-cscay
cx              idcorrf=0
cx           endif
cx        endif
cx      endif
c
      if(idphoto.eq.0) goto 56
      call internd_PA(1,idphoto,nphoto,dphi,istart,iend,istep,indpol,
     +     idMie,x,nmax,ref0,px,rsx,rsr,rsi,NXMAX,py0,py,dpy,as,bs)
56    write(*,*) 'idimen,IDshape: '
```

```
      write(*,*) idimen,IDshape
      indpol=indpol+2
      if(indpol.lt.3) then
       write(*,107)
       write(*,*) 'Solving for y-pol. inci. state'
       goto 3
      endif
c
      if(idimen.eq.1) totv=4.d0*dnLPA*dx(1)*r0(4,1)*r0(4,1)
      if(idimen.eq.2) totv=2.d0*dnLPA*dx(1)*dx(2)*r0(4,1)
      if(idimen.eq.3) totv=dnLPA*dx(1)*dx(2)*dx(3)
      temp=4.d0*pione/3.d0
      totvpar=dnLPA*temp*r0(4,1)**3
      write(*,107)
      write(*,*) 'dnLPA: ',dnLPA
      gcv=totv/temp
      gcvr=gcv**(1.d0/3.d0)
      gcvpar=totvpar/temp
      gcvrpar=gcvpar**(1.d0/3.d0)
      write(*,*) 'Effective radius: ',gcvrpar,gcvr
      xv=k*gcvr
      xvpar=k*gcvrpar
      write(*,*) 'volume-equiv. xv: ',xvpar,xv
      write(*,*) ' '
c ---------------------------------------------------------------------
c output amplitude scattering matrix
c ---------------------------------------------------------------------
      write(*,*)
     + 'amplitude scattering matrix is in the output file amp.out'
      open(23,file='amp.out',status='unknown')
      write(23,*) 'amp.out   (amplitude scattering matrix)'
      write(23,*) 'wavelength,totv: ',w,totv
      write(23,*) 'sphere#,x,y,z,r,complex refractive index:'
      do i=1,2
       write(23,117) i,r0(1,i),r0(2,i),
     +              r0(3,i),r0(4,i),r0(5,i),r0(6,i)
      enddo
      write(23,*) 'scattering angle, s2x(complex), s3y(complex)'
      write(23,*) '                  s4x(complex), s1y(complex)'
      do jc=1,npng
       temp=pang*dble(jc-1)
       write(23,101) 'phi (in degrees): ',temp
       do i=1,nang2
        write(23,118) dang(i),dble(s2x(jc,i)),dimag(s2x(jc,i)),
     +                dble(s3y(jc,i)),dimag(s3y(jc,i))
        write(23,119)        dble(s4x(jc,i)),dimag(s4x(jc,i)),
     +                dble(s1y(jc,i)),dimag(s1y(jc,i))
       enddo
      enddo
      close(23)
c ---------------------------------------------------------------------
c output Mueller matrix
c ---------------------------------------------------------------------
      open(11,file='mueller.out',status='unknown')
      write(11,134) 'mueller.out','(Mueller matrix)'
```

```
      write(11,135) 'totv: ',totv
      write(11,136) 'nbeta,nthet,nphai: ',nbeta,nthet,nphai
      write(11,137) 'Ranges of Euler angles: ',
     +              betami,betamx,thetmi,thetmx,phaimi,phaimx
      write(11,138) '# of orientations averaged: ',nram
      write(11,139) 'interval of scattering angle: ',sang
      write(11,139) 'interval of azimuth angle:    ',pang
      do jc=1,npng
       t=pang*dble(jc-1)
       write(11,140) 'phi (in degrees): ',t
       do i=1,nang2
        call mueller(s1y(jc,i),s2x(jc,i),s3y(jc,i),s4x(jc,i),smue)
c        do n=1,4
c         do m=1,4
c          smue(n,m)=smue(n,m)/totv**2
c         enddo
c        enddo
         write(11,141) dang(i),(smue(1,m),m=1,4)
         write(11,142)         (smue(2,m),m=1,4)
         write(11,142)         (smue(3,m),m=1,4)
         write(11,142)         (smue(4,m),m=1,4)
       enddo
      enddo
      write(11,*) 'phi (in degrees): 360'
      jc=1
      do i=1,nang2
       call mueller(s1y(jc,i),s2x(jc,i),s3y(jc,i),s4x(jc,i),smue)
c        do n=1,4
c         do m=1,4
c          smue(n,m)=smue(n,m)/totv**2
c         enddo
c        enddo
       write(11,141) dang(i),(smue(1,m),m=1,4)
       write(11,142)         (smue(2,m),m=1,4)
       write(11,142)         (smue(3,m),m=1,4)
       write(11,142)         (smue(4,m),m=1,4)
       if(i.eq.nang2) then
        open(15,file='rightcirc.out',status='unknown')
        cz=k*k
        temp0=4.d0*pione/cz
        temp=smue(1,1)
            temp2=smue(1,2)
            temp=temp*temp0
            temp2=temp2*temp0
            temp1=0.5d0*(temp+temp2)
            temp2=0.5d0*(temp-temp2)
            write(*,*) ' '
            write(*,*) 'linear cross sections: x,y,total'
            write(*,*) temp1,temp2,temp
            write(15,*) 'linear cross sections: x,y,total'
            write(15,*) temp1,temp2,temp
        temp=smue(1,1)+smue(4,1)
            temp2=smue(1,4)+smue(4,4)
            temp=temp*temp0
            temp2=temp2*temp0
```

```
            temp1=0.5d0*(temp+temp2)
            temp2=0.5d0*(temp-temp2)
            write(*,*) 'right-circular cross sections: pp,op,total'
            write(*,*) temp2,temp1,temp
            write(*,*) ' '
            write(15,*) 'right-circular cross sections: pp,op,total'
            write(15,*) temp2,temp1,temp
            close(15)
         endif
        enddo
        close(11)
c -------------------------------------------------------------------
c output polarization and cross sections
c -------------------------------------------------------------------
        do i=1,nang2
         i22(i)=i22(i)+cdabs(s2x(1,i))*cdabs(s2x(1,i))
         i21(i)=i21(i)+cdabs(s4x(1,i))*cdabs(s4x(1,i))
         i11(i)=i11(i)+cdabs(s1y(1,i))*cdabs(s1y(1,i))
         i12(i)=i12(i)+cdabs(s3y(1,i))*cdabs(s3y(1,i))
         inat(i)=i11(i)+i22(i)+i12(i)+i21(i)
         pol(i)=(i11(i)+i12(i)-i22(i)-i21(i))/inat(i)
        enddo
        cbakx_ck2=cdabs(s2x(1,nang2))*cdabs(s2x(1,nang2))
        cbaky_ck2=cdabs(s1y(1,nang2))*cdabs(s1y(1,nang2))
        cz=k*k
        rn=4.d0*pione/cz
        cbakx_ck2=cbakx_ck2*rn
        cbaky_ck2=cbaky_ck2*rn
        write(*,*) ' Cbakx,Cbakx_ck2: ',cbakx,cbakx_ck2
        write(*,*) ' Cbaky,Cbaky_ck2: ',cbaky,cbaky_ck2
        cextx2=s2x(1,1)
        cexty2=s1y(1,1)
c       cextx_ck1=twopi*cextx_ck/cz
c       cexty_ck1=twopi*cexty_ck/cz
c       cext_ck1=0.5d0*(cextx_ck1+cexty_ck1)
        cextx2=2.d0*twopi*cextx2/cz
        cexty2=2.d0*twopi*cexty2/cz
        cext2=0.5d0*(cextx2+cexty2)
c
c       write(*,*) '  cextx(y)_ck: ',cextx_ck,cexty_ck
c       write(*,*) '  cextx(y)   : ',cextx,cexty
cc       cextx_ck=cextx
cc       cexty_ck=cexty
        cext_ck=0.5d0*(cextx_ck+cexty_ck)
cc       cextx=cscax+cabsx
cc       cexty=cscay+cabsy
        cext=0.5d0*(cextx+cexty)

c       write(*,*) '  cextx(y)_ck: ',cextx_ck,cexty_ck
c       write(*,*) '  cextx(y)   : ',cextx,cexty

        cext_ick=0.5d0*(cextx_ick+cexty_ick)
        write(*,*) 'checking Cext:'
        write(*,*) 'cext_ick,cext: ',cext_ick,cext
        write(*,*) 'cextx_ick,cextx: ',cextx_ick,cextx
```

```
      write(*,*) 'cexty_ick,cexty: ',cexty_ick,cexty
      write(*,*) 'checking Cext(2):'
      write(*,*) 'Cext_ck,cext2:  ',cext_ck,cext2
      write(*,*) 'Cextx_ck,cextx2: ',cextx_ck,cextx2
      write(*,*) 'Cexty_ck,cexty2: ',cexty_ck,cexty2
c
      csca=0.5d0*(cscax+cscay)
      cabs=0.5d0*(cabsx+cabsy)
      cprx=cextx-cscax*assymx
      cpry=cexty-cscay*assymy
      assym_ck=0.5d0*(assymx+assymy)
      assymx_ck=assymx
      assymy_ck=assymy
      cpr=cext-csca*assym
c      assym0_ck=0.5d0*(cprx/cscax+cpry/cscay)
c
cc      cscax_ck=cextx_ck-cabsx
cc      cscay_ck=cexty_ck-cabsy
      cscax_ck=cextx-cabsx
      cscay_ck=cexty-cabsy
      csca_ck=0.5d0*(cscax_ck+cscay_ck)
      cabs_ck=0.5d0*(cabsx_ck+cabsy_ck)
      write(*,*) 'checking Csca:'
      write(*,*) 'Csca,Csca_ck:   ',csca,csca_ck
      write(*,*) 'Cscax,Cscax_ck: ',cscax,cscax_ck
      write(*,*) 'Cscay,Cscay_ck: ',cscay,cscay_ck
      write(*,*) 'checking Cabs:'
      write(*,*) 'Cabs,Cabs_ck:   ',cabs,cabs_ck
      write(*,*) 'Cabsx,Cabsx_ck: ',cabsx,cabsx_ck
      write(*,*) 'Cabsy,Cabsy_ck: ',cabsy,cabsy_ck
c
cc      temp1=cscax_ck/cscax
cc      assymx=assymx/temp1
cc      temp=cextx_ck/cextx
cc      cscax=cscax*temp
cc      temp1=cscax_ck/cscax
cc      temp2=temp1*temp
cc      cabsx=cabsx*temp2
cc      cextx=cabsx+cscax
cc      temp1=cscay_ck/cscay
cc      assymy=assymy/temp1
cc      temp=cexty_ck/cexty
cc      cscay=cscay*temp
cc      temp1=cscay_ck/cscay
cc      temp2=temp1*temp
cc      cabsy=cabsy*temp2
cc      cexty=cabsy+cscay
c
ccc      temp=cextx_ck/cextx
c      temp=cextx/cextx_ck
c      cscax=cscax*temp
ccc      temp1=cextx_ck/cextx_ick
c      temp1=cextx/cextx_ick
c      assymx=assymx/(temp*temp1)
c      temp2=cscax_ck/cscax
```

```
c     if(temp1.gt.temp2) temp1=temp2
c     cabsx0=cabsx
c     cabsx=cabsx*temp1*temp
c     cextx=cabsx+cscax
ccc     temp=cexty_ck/cexty
c     temp=cexty/cexty_ck
c     cscay=cscay*temp
ccc     temp1=cexty_ck/cexty_ick
c     temp1=cexty/cexty_ick
c     assymy=assymy/(temp*temp1)
c     temp2=cscay_ck/cscay
c     if(temp1.gt.temp2) temp1=temp2
c     cabsy0=cabsy
c     cabsy=cabsy*temp1*temp
c     cexty=cabsy+cscay


ccc     temp=cextx_ck/cextx
ccc     cscax=0.5d0*(cscax_ick+cscax_ck2)
ccc     cabsx=0.5d0*(cabsx_ick+cabsx)
ccc     cextx=cabsx+cscax
ccc     temp=cexty_ck/cexty
ccc     cscay=0.5d0*(cscay_ick+cscay_ck2)
ccc     cabsy=0.5d0*(cabsy_ick+cabsy)
ccc     cexty=cabsy+cscay

cxu2      temp=cextx_ick/cextx
cxu      cextx=0.5d0*(cextx+cextx_ick)
cxu      cscax=cscax_ick
cxu      cabsx=cabsx_ick
cxu      cbakx=cbakx*temp
cxu      temp=cbakx/cbakx00
cxu2      assymx=temp*assymx
cxu2      cbakx=cbakx*temp
cxu      cbakx=cbakx00
cxu2      temp=cexty_ick/cexty
cxu      cexty=0.5d0*(cexty+cexty_ick)
cxu      cscay=cscay_ick
cxu      cabsy=cabsy_ick
cxu      cbaky=cbaky*temp
cxu      temp=cbaky/cbaky00
cxu2      assymy=temp*assymy
cxu2      cbaky=cbaky*temp
cxu      cbaky=cbaky00

c
ccc     temp1=cextx_ck/cextx
ccc     temp2=cextx_ick/cextx
ccc     temp=cscax_ick/cscax*cabsx0/cabsx
ccc     temp=temp*temp1*temp2
ccc     write(*,*) ' adj.fac. for Cbakx: ',temp
c     cbakx=cbakx*temp
ccc     temp1=cexty_ck/cexty
ccc     temp2=cexty_ick/cexty
ccc     temp=cscay_ick/cscay*cabsy0/cabsy
```

```
ccc      temp=temp*temp1*temp2
ccc      write(*,*) ' adj.fac. for cbaky: ',temp
c      cbaky=cbaky*temp
c
      cext=0.5d0*(cextx+cexty)
      csca=0.5d0*(cscax+cscay)
      cabs=0.5d0*(cabsx+cabsy)
      cprx=cextx-cscax*assymx
      cpry=cexty-cscay*assymy
      assym=0.5d0*(assymx+assymy)
      cpr=cext-csca*assym
      cbak=0.5d0*(cbakx+cbaky)
c      assym0=0.5d0*(cprx/cscax+cpry/cscay)
c
c      cscax_ck=cextx-cabsx
c      cscay_ck=cexty-cabsy
c      csca_ck=0.5d0*(cscax_ck+cscay_ck)
c      cabs_ck=0.5d0*(cabsx_ck+cabsy_ck)
      write(*,*)
      write(*,*) 'assym. par.:   ',assym,assym_ck
      write(*,*) 'assym. par.(x):',assymx,assymx_ck
      write(*,*) 'assym. par.(y):',assymy,assymy_ck
      write(*,*) 'Cext:'
      write(*,*) 'Cext,Cext_ck:   ',cext,cext_ck
      write(*,*) 'Cextx,Cextx_ck: ',cextx,cextx_ck
      write(*,*) 'Cexty,Cexty_ck: ',cexty,cexty_ck
      write(*,*) 'Csca:'
      write(*,*) 'Csca,Csca_ck:   ',csca,csca_ck
      write(*,*) 'Cscax,Cscax_ck: ',cscax,cscax_ck
      write(*,*) 'Cscay,Cscay_ck: ',cscay,cscay_ck
      write(*,*) 'Cabs:'
      write(*,*) 'Cabs,Cabs_ck:   ',cabs,cabs_ck
      write(*,*) 'Cabsx,Cabsx_ck: ',cabsx,cabsx_ck
      write(*,*) 'Cabsy,Cabsy_ck: ',cabsy,cabsy_ck
c
c      cbakx=2.0d0*twopi*cbakx/cz
c      cbaky=2.0d0*twopi*cbaky/cz
c      cbak=0.5d0*(cbakx+cbaky)
cPA      cbak_ck=0.5d0*(cbakx_ck+cbaky_ck)
cPA      write(*,*) 'checking Cbak:'
cPA      write(*,*) 'Cbak,Cbak_ck: ',cbak,cbak_ck
cPA      write(*,*) 'Cbakx,Cbakx_ck: ',cbakx,cbakx_ck
cPA      write(*,*) 'Cbaky,Cbaky_ck: ',cbaky,cbaky_ck
cPA      write(*,'(5x,2e15.6)') assym,assym0
      write(*,107)
      write(*,120)
     + 'Cext','Cabs','Csca','Cbak','Cpr','<cos(theta)>'
c00     write(*,'(2x,6e13.5)') cext,cabs,csca,cbak,cext-cpr,assym
      write(*,121) cext,cabs,csca,cbak,cpr,assym
      write(*,127) 'x',cextx,cabsx,cscax,cbakx,cprx,assymx
      write(*,127) 'y',cexty,cabsy,cscay,cbaky,cpry,assymy
c00
cPA      i=1
cPA      cabsxi(i)=4.0d0*pione*cabsxi(i)
cPA      cabsyi(i)=4.0d0*pione*cabsyi(i)
```

```
cPA      cextxi(i)=2.0d0*pione*cextxi(i)
cPA      cextyi(i)=2.0d0*pione*cextyi(i)
cPA      cscaxi(i)=4.0d0*pione*cscaxi(i)
cPA      cscayi(i)=4.0d0*pione*cscayi(i)
cPA      cprxi(i)=4.0d0*pione*cprxi(i)
cPA      cpryi(i)=4.0d0*pione*cpryi(i)
cPA      cscai(i)=0.5d0*(cscaxi(i)+cscayi(i))
cPA      cexti(i)=0.5d0*(cextxi(i)+cextyi(i))
cPA      cabsi(i)=0.5d0*(cabsxi(i)+cabsyi(i))
cPA      cpri(i)=0.5d0*(cprxi(i)+cpryi(i))
c00      cpri(i)=cscai(i)+cabsi(i)-cpri(i)
cPA      assymxi(i)=cprxi(i)/cscaxi(i)
cPA      assymyi(i)=cpryi(i)/cscayi(i)
cPA      assymi(i)=0.5d0*(cprxi(i)+cpryi(i))/cscai(i)
c00      cprxi(i)=cscaxi(i)+cabsxi(i)-cprxi(i)
c00      cpryi(i)=cscayi(i)+cabsyi(i)-cpryi(i)
cPA      write(*,'(i5,5e15.6)') i,cexti(i),cabsi(i),
cPA     +                cscai(i),cpri(i),assymi(i)
cPA      write(*,'(/,a)') 'efficiencies for radiation pressure'
cPA      write(*,'(5x,6e13.5)') assym,assym0
cPA      write(*,'(5x,6e13.5)') assym,cext-cpr,assymx,cextx-cprx,
cPA     +                                assymy,cexty-cpry
c00      write(*,'(5x,6e13.5)') assym,cpr,assymx,cprx,assymy,cpry
cspc ----------------------------------------------------------------
cPA      i=1
cPA      write(*,'(i5,6e13.5)') i,assymi(i),cpri(i),
cPA     +       assymxi(i),cprxi(i),assymyi(i),cpryi(i)
c      betami=betami*90.d0/pih
c      betamx=betamx*90.d0/pih
c      thetmi=thetmi*90.d0/pih
c      thetmx=thetmx*90.d0/pih
c      phaimi=phaimi*90.d0/pih
c      phaimx=phaimx*90.d0/pih
       flout='crGMM_PA.out'
       open(33,file=flout,status='unknown')
       write(33,122)
      +   flout,'(Total and individual-particle cross sections)'
       write(33,123)
      +   'input sphere-aggregate filename:',FLNAME
cPA      write(33,'(a19,3i5)') 'nbeta,nthet,nphai: ',
cPA     +                    nbeta,nthet,nphai
cPA       write(33,'(a24,6f7.2)') 'Ranges of Euler angles: ',
cPA     +          betami,betamx,thetmi,thetmx,phaimi,phaimx
cPA       write(33,'(a28,i5)') '# of orientations averaged: ',nram
       write(33,124)
      +    'Cext','Cabs','Csca','Cpr','<cos(theta)>'
cPA      write(33,'(a5,5e15.6)') 'total',cext,cabs,csca,cext-cpr,
cPA     +     assym
       write(33,125) 'total',cext,cabs,csca,cpr,assym
       write(33,127) 'x',cextx,cabsx,cscax,cbakx,cprx,assymx
       write(33,127) 'y',cexty,cabsy,cscay,cbaky,cpry,assymy
       i=1
cPA      write(33,'(i5,5e15.6)') i,cexti(i),cabsi(i),cscai(i),
cPA     +          cpri(i),assymi(i)
       close(33)
```

```
      cz=pione*gcvr*gcvr
c       assym=cpr/csca
c       assymx=cprx/cscax
c       assymy=cpry/cscay
      cabsxv=cabsx/cz
      cabsyv=cabsy/cz
      cextxv=cextx/cz
      cextyv=cexty/cz
      cscaxv=cscax/cz
      cscayv=cscay/cz
      cprxv=cprx/cz
cPA       cprxv=cextxv-cprxv
      cpryv=cpry/cz
cPA       cpryv=cextyv-cpryv
      cscav=0.5d0*(cscaxv+cscayv)
      cextv=0.5d0*(cextxv+cextyv)
      cabsv=0.5d0*(cabsxv+cabsyv)
      cprv=0.5d0*(cprxv+cpryv)
      cbakxv=cbakx/cz
      cbakyv=cbaky/cz
      cbakv=0.5d0*(cbakxv+cbakyv)
c       temp=gcvr*gcvr/gcs
c       cabsxs=cabsxv*temp
c       cabsys=cabsyv*temp
c       cextxs=cextxv*temp
c       cextys=cextyv*temp
c       cscaxs=cscaxv*temp
c       cscays=cscayv*temp
c       cprxs=cprxv*temp
c       cprys=cpryv*temp
c       cscas=cscav*temp
c       cexts=cextv*temp
c       cabss=cabsv*temp
c       cprs=cprv*temp
c       cbakxs=cbakxv*temp
c       cbakys=cbakyv*temp
c       cbaks=cbakv*temp
      write(*,126)
     +   'Qextv','Qabsv','Qscav','Qbakv','Qprv','<cos(theta)>'
      write(*,127) 't',cextv,cabsv,cscav,cbakv,cprv,assym
      write(*,127) 'x',cextxv,cabsxv,cscaxv,cbakxv,cprxv,assymx
      write(*,127) 'y',cextyv,cabsyv,cscayv,cbakyv,cpryv,assymy
c       write(*,126)
c      +   'Qexts','Qabss','Qscas','Qbaks','Qprs','<cos(theta)>'
c       write(*,127) 't',cexts,cabss,cscas,cbaks,cprs,assym
c       write(*,127) 'x',cextxs,cabsxs,cscaxs,cbakxs,cprxs,assymx
c       write(*,127) 'y',cextys,cabsys,cscays,cbakys,cprys,assymy
      fileout='GMM_PA.out'
      open(22,file=fileout,status='unknown')
      write(22,129) fileout,
     +   '--- input file: ',FLNAME,' xv:',xv,'  xs:',xs
      write(22,130) 'Ranges of Euler angles: ',
     +          betami,betamx,thetmi,thetmx,phaimi,phaimx
      write(22,131) 'nbeta,nthet,nphai: ',nbeta,nthet,nphai,
     +             '# of orientations averaged: ',nram
```

```
      write(22,126)
     +   'Cext','Cabs','Csca','Cbak','Cpr','<cos(theta)>'
      write(22,127) 't',cext,cabs,csca,cbak,cpr,assym
      write(22,127) 'x',cextx,cabsx,cscax,cbakx,cprx,assymx
      write(22,127) 'y',cexty,cabsy,cscay,cbaky,cpry,assymy
      write(22,126)
     +   'Qextv','Qabsv','Qscav','Qbakv','Qprv','<cos(theta)>'
      write(22,127) 't',cextv,cabsv,cscav,cbakv,cprv,assym
      write(22,127) 'x',cextxv,cabsxv,cscaxv,cbakxv,cprxv,assymx
      write(22,127) 'y',cextyv,cabsyv,cscayv,cbakyv,cpryv,assymy
c     write(22,126)
c     +   'Qexts','Qabss','Qscas','Qbaks','Qprs','<cos(theta)>'
c     write(22,127) 't',cexts,cabss,cscas,cbaks,cprs,assym
c     write(22,127) 'x',cextxs,cabsxs,cscaxs,cbakxs,cprxs,assymx
c     write(22,127) 'y',cextys,cabsys,cscays,cbakys,cprys,assymy
      write(22,107)
      write(22,*)
     +'For the scattering plane of phi=0 (i.e., the x-z plane) only:'
      write(22,132)
     +    's.a.','itotal','pol.','S1*S1','S4*S4','S3*S3','S2*S2'
      do i=1,nang2
        write(22,133)
     +      dang(i),inat(i),pol(i),i11(i),i21(i),i12(i),i22(i)
      enddo
      close(22)
  101   format(a,f9.3)
  102   format(a,i3)
  103   format(a,e10.2)
  104   format(a,i3,i6,f5.1,3i6)
  105   format(e15.6,2(1x,e15.6))
  106   format(e15.6,1x,e15.6)
  107   format(/)
  108   format(i12,3e14.5)
  109   format(a,f10.4)
  110   format(a,1x,i4)
  111   format(i10,4e15.7)
  112   format(a,i4,2x,e15.7)
  113   format(i6,1x,i5,4(1x,e20.12))
  115   format(a,1x,e15.6)
  116   format(a,f7.4,3x,a,a20)
  117   format(i5,6f11.4)
  118   format(f8.2,4e15.6)
  119   format(8x,4e15.6)
  120   format(6x,a4,9x,a4,9x,a4,9x,a4,9x,a3,8x,a12)
  121   format(2x,6e13.5)
  122   format(a20,a47)
  123   format(a32,2x,a22)
  124   format(12x,a4,11x,a4,11x,a4,11x,a3,8x,a12)
  125   format(a5,5e15.6)
  126   format(6x,a5,8x,a5,8x,a5,8x,a5,8x,a4,5x,a12)
  127   format(1x,a1,6e13.5)
  128   format(/,a,e14.5)
  129   format(a20,a16,a18,a4,f8.3,a5,f8.3)
  130   format(a24,6f7.2)
  131   format(a19,3i5,6x,a28,i5)
```

```
132   format(2x,a4,5x,a6,6x,a4,6x,a5,8x,a5,8x,a5,8x,a5)
133   format(f7.1,e13.5,f8.4,4e13.5)
134   format(a11,6x,a16)
135   format(a6,1x,e16.8)
136   format(a19,3i5)
137   format(a24,6f7.2)
138   format(a28,i5)
139   format(a30,f9.3)
140   format(a18,f8.3)
141   format(f7.1,4e16.7)
142   format(7x,4e16.7)
      stop
      end
      include 'abMiexud.f'
      include 'besseljd.f'
      include 'besselyd.f'
      include 'carsphd.f'
      include 'cofd0.f'
      include 'cofnv0.f'
      include 'cofsrd.f'
      include 'cofxuds0.f'
      include 'gau0.f'
      include 'gid0.f'
      include 'gxurcd0.f'
      include 'internd_PA.f'
      include 'lnfacd.f'
      include 'mueller.f'
      include 'plgndrd.f'
      include 'psiy.f'
      include 'rotcoef.f'
      include 'rtr.f'
      include 'tipitaud.f'
      include 'trv.f'
      include 'transs_PA_v0.f'
      include 'getrgr3Dcfg.f'
      include 'isort2.f'
      include 'rotatEuler3.f'
      include 'ctotpha3.f'
      include 'totphase.f'
      include 'BESSJ1d.f'
      include 'phaseEllipsiod.f'
      include 'chkInp_getTotNum.f'
```