Electronic Supplementary Material (ESI) for Analytical Methods.

Supporting Information for: Measuring the Relative Concentration of Particle Populations using Differential Centrifugal Sedimentation

Alexander G. Shard[1], Katia Sparnacci,[2] Aneta Sikora,[1] Louise Wright,[1] Dorota Bartczak,[3] Heidi Goenaga-Infante[3] and Caterina Minelli[1*]

[1] National Physical Laboratory, Hampton Road, Teddington, Middlesex, TW11 0LW, UK

[2] Università del Piemonte Orientale ‘‘A. Avogadro’’, Viale T. Michel 11, I-15121 Alessandria, Italy

[3]LGC Limited, Queens Road, Teddington, Middlesex, TW11 0LY, UK

*Corresponding author: Tel: +44 (0)20 8943 6193; Fax: +44 (0)20 8943 6453; e-mail: caterina.minelli @npl.co.uk

S.1: mass concentration of monodisperse, spherical particles in DCS

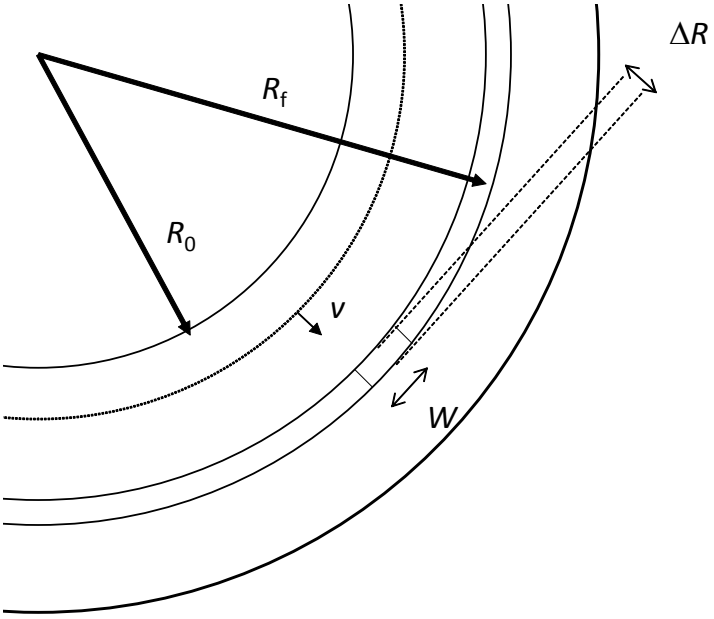

Figure S.1 Schematic of a DCS instrument. The detection area is defined by W and ΔR.

Particles sediment at a velocity, v_R, which for spherical particles is given by Equation S.1:

$$v_R = \frac{dR}{dt} = \frac{R\omega^2(\rho_P - \rho_R)D^2}{18\eta_R} \quad \text{S.1}$$

Where R is the distance from the centre of rotation, with fluid density and viscosity at that radius given by ρ_R and η_R respectively. ω is the disk angular velocity, D is the spherical particle diameter and ρ_P is the particle density.

Integration provides the normal sedimentation equation S.2, where the subscript 'a' represents an average value across the gradient, which is generally assumed to be the mean of the values at R_0 and R_f.

$$t = \frac{18\eta_a \ln\left(R_f/R_0\right)}{\omega^2(\rho_P - \rho_a)D^2} \qquad \text{S.2}$$

The detector covers an area approximated as a rectangle with width W and length ΔR. The path length of light through the medium is L and the detection volume is the product of the three parameters. Measured light intensity, I, is recorded as a function of time and converted into absorbance, $A = \ln(I_0/I)$ where I_0 is a reference intensity recorded when no particles are in the detection area.

Ignoring finer details such as: dispersity in the particle population, wall effects and particle diffusion, the time taken for particles in the population to cross the detection area, Δt, is given to a good approximation by equation S.3. Here, the subscript 'f' indicates the values at the average detector position.

$$\Delta t = \frac{\Delta R}{v_f} = \frac{\Delta R 18\eta_f}{R_f\omega^2(\rho_P - \rho_f)D^2} \qquad \text{S.3}$$

With the same assumptions, the absorbance, A, will be constant during this time and is related to the individual particle extinction cross section, σ, the number concentration of particles in the analysis volume, $N/(LW\Delta R)$, and the path length L by: $A = \sigma N/W\Delta R$. Thus the integrated absorbance with time is given by equation S.4.

$$A\Delta t = \frac{18\sigma N\eta_f}{WR_f\omega^2(\rho_P - \rho_f)D^2} \qquad \text{S.4}$$

Conversion of the time scale to a diameter scale changes Δt to ΔD and, through differentiation of S.2 to obtain $dt/dD \approx \Delta t/\Delta D$, the integrated absorbance with diameter is given by equation S.5

$$A\Delta D = \frac{\sigma N D}{2WR_f \ln\left(R_f/R_0\right)}\left[\frac{\eta_f(\rho_P - \rho_a)}{\eta_a(\rho_P - \rho_f)}\right] \qquad \text{S.5}$$

The final ratio in square brackets relates to the difference between the 'average' fluid parameters and those at the detection position. We combine these into a correction parameter, δ, which is a number larger than 1 for sedimentation and depends largely upon viscosity differences, but also upon particle density. With the same assumption used for S.3, Equation S.6 relates the total mass of particles injected, M, to the number of particles in the detection zone during Δt. Here it is assumed that the physical diameter of the particle is the same as the sedimentation diameter.

$$N = \frac{6M \quad W}{\pi D^3 \rho_P 2\pi R_f} \qquad \text{S.6}$$

Combining S.6 and S.5 and rearranging to obtain the mass of particles injected divided by the diameter range in which they are detected, Equation S.7 is obtained.

$$\frac{M}{\Delta D} = \frac{1}{\delta}\left[\frac{8\pi R_f^2 \ln\left(R_f/R_0\right)}{3}\right]\left\{\frac{\pi D^2}{4\sigma}\right\}\rho_P A = \frac{\gamma}{\delta}\rho_P \frac{A}{Q} \qquad \text{S.7}$$

The square brackets enclose terms that relate to the instrument, γ. For homogeneous, rigid, spherical particles the terms in curly brackets can be replaced with the reciprocal of the extinction efficiency, Q^{-1} and Q is obtained directly from Mie theory. Assuming, once again, that the physical diameter of the particle is the same as the sedimentation diameter. Note that error in R_f and R_0 contribute significantly to uncertainty in the detected mass.

Here, we have neglected consideration of the finite area of the detector which will collect a fraction of light scattered at low angles. The manufacturer's software accounts for this and uses an adjusted value for Q, Q_{net}, which will be smaller than Q calculated directly from Mie theory [CPS instruments application note 'conversion of CPS data into a weight distribution']. The value of Q_{net} can be extracted from the instrument and compared to Q_{Mie} calculated directly from Mie theory without accounting for forward scattered light hitting the detector. The ratio of these values for gold, silica and polystyrene are shown in Figure S.2. For particles less than 100 nm in diameter the two numbers are nearly identical and remain within 5% of each other until the particles are significantly larger than 200 nm diameter.

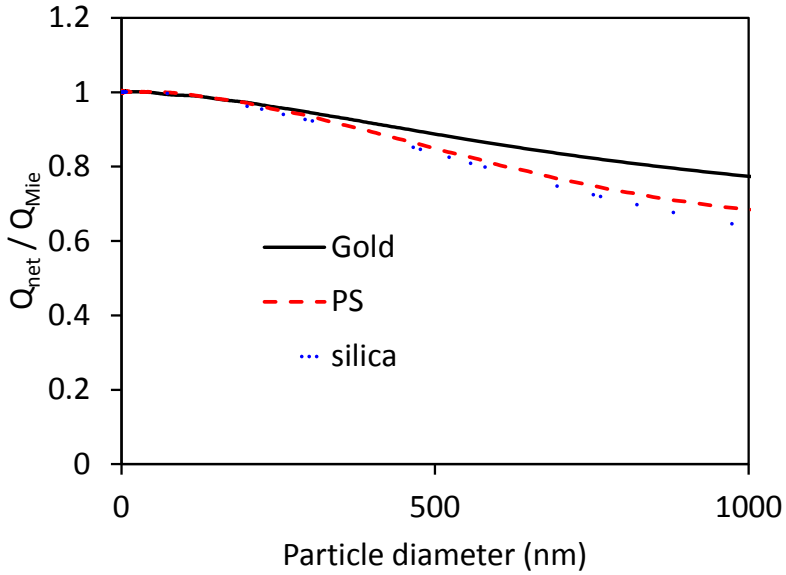

Figure S.2 Ratio of Q_{net} calculated by the instrument software to Q_{Mie} calculated using the same physical parameters. The difference between these values relates to the amount of forward scattered light hitting the detector, which is explicitly evaluated for Q_{net}.

The conversion can be verified by calculating the ratio of absorbance to mass per diameter step directly from the data and showing that this is proportional to Q. See Figure S.3. The value of γ calculated using typical values: $R_f = 45$ mm, $R_0 = 35$ mm is approximately 4×10^9 μm^2. Silica has a density of approximately $\rho_P = 2\times10^{-6}$ μg/μm^3. The slope on the graph should be approximately equal to $1/(\gamma\rho_P)$, which is ~1.2×10^{-4} and within a factor of two of the slope found here. The reason for the difference between these values is not clear.

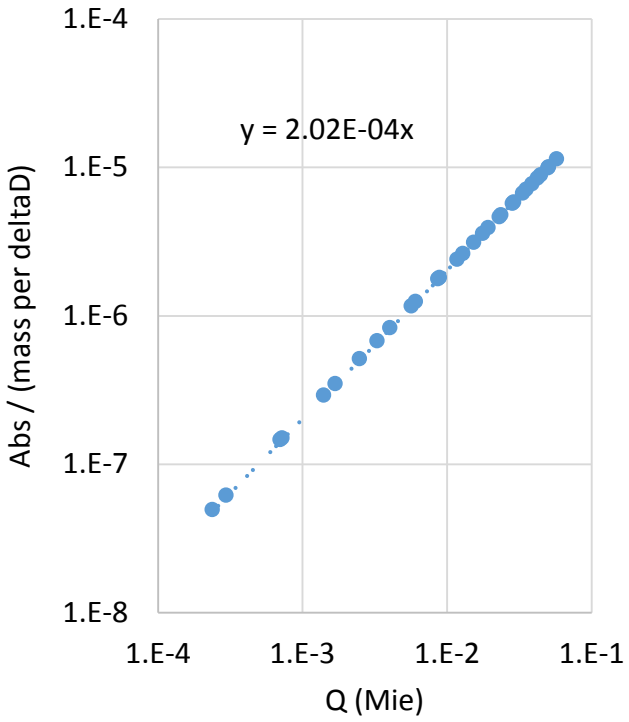

Figure S.3 Conversion factor for a DCS instrument. Absorbance data for silica particles in the range 40 nm to 200 nm were background corrected using a simple linear function and divided by the software output of mass per diameter step (µg/µm). These are plotted against Mie extinction efficiencies at selected diameters. The dashed line is a linear fit.

For a defined particle population with a narrow size distribution, $M/\Delta D$ is integrated over D to obtain the total mass in that population injected into the centrifuge. Integrating S.7, assuming either that σ is constant for that population or that a suitable average value can be found, provides the relationship shown in S.8.

$$M \propto \frac{D^3 \rho_P}{\sigma} \qquad \text{S.8}$$

Whilst this result seems obvious, the analysis is necessary to identify the sources of error. The terms in S.8 are M, the total mass of particles in the population *measured* by integration of the converted absorbance *versus* time data. The diameter, D, is the physical diameter of the spherical particles and the density, ρ_P, is the physical density. Note that D may not be identical to the measured diameter, D_m, which is often in error due to shape effects (see next section) or incorrect densities. The input density of the particle, ρ_X, is a source of uncertainty but, through the relationship to sedimentation time in S.2, is anticorrelated with D_m.

The absorption cross section, σ, is the 'true' cross section of the particles and not necessarily the same as the cross section calculated by Mie theory, σ_X. As well as errors that arise from possibly incorrect refractive indices, the cross section is calculated from the measured diameter, D_m, which depends upon ρ_X as noted above. For small ($D < 50$ nm) particles σ scales as D^6 for purely scattering materials and D^3 for absorbing particles. Thus, for small absorbing particles any errors in diameter cancel in S.8, leaving only the density term.

Thus, equation S.9 relates the mass of particles in the population to the measured mass, M_m.

$$M = \frac{D^3 \rho_P \ \ \sigma_X}{\sigma \ \ D_m^3 \rho_X} M_m \qquad \text{S.9}$$

S.2: mass concentration of monodisperse, non-spherical particles in DCS

For non-spherical particles, a correction to the equations in section S.1 are required, starting with S.1:

$$v_R = \frac{dR}{dt} = \frac{R\omega^2(\rho_P - \rho_R)D^2}{18\chi\eta_R} \qquad \text{S.10}$$

All terms have the same meaning as S.1, except that D is now interpreted as the diameter of a sphere with an equal volume to that of the particle and a dynamic shape factor, χ, is introduced to express the ratio of the frictional force on the particle to that of a sphere of diameter D. This factor is propagated through equations S.2 to S.4 and in S.5 it (like the angular velocity term) cancels and the form of that equation is unchanged except for the definition of D. Because of the definition of D, S.6 is unchanged and so S.7 is also valid except for the conversion of the diameter and the cross section into Q because, except in limited circumstances, Mie theory is not valid for non-spherical particles. Therefore, the form of S.9 is unchanged and the D^3 terms are better replaced with particle volumes, V, to avoid confusion concerning the definition of D.

$$M = \frac{V\rho_P \ \ \sigma_X}{\sigma \ \ V_m \rho_X} M_m \qquad \text{S.11}$$

From S.10, the following relationships are also useful: $D = \chi^{0.5} D_\mathrm{m}$ and therefore $V = \chi^{1.5} V_\mathrm{m}$.

S3: Repeatability and precision

To provide the context for the analysis of error in this paper, it is important to point out that the DCS technique provides excellent precision and repeatability. Figure S.7 overlays two repeat measurements of the same mixed particle system. There is some variability in the peak height and width, which are of the order of 5 to 10%, however the important parameters are the modal peak diameter and the area of each peak in mass. The modal diameter has a variability of <0.5% and, although the area varies between runs by ~6%, the ratio of peak areas, w_m, varies by <0.5%. The latter is expected if the change in peak areas are correlated due to e.g. uncertainty in the injection volume. Therefore the measurement of particle size and relative concentration in DCS is not limited by the precision and repeatability of the method.

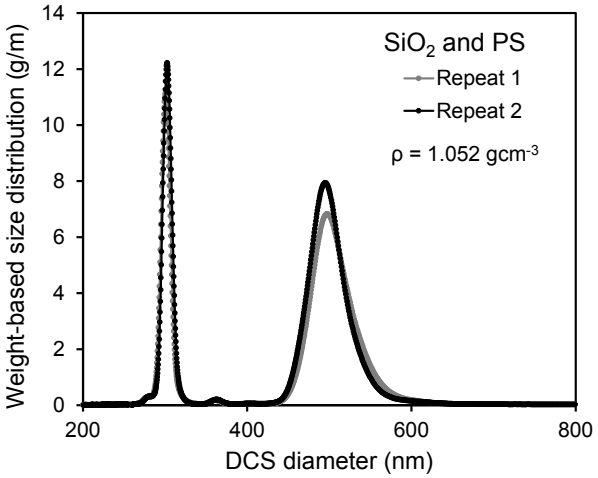

Figure S.4. Repeat measurements of a mixed particle system acquired in two separate experiments.

S4: Thiol functionalised gold particles

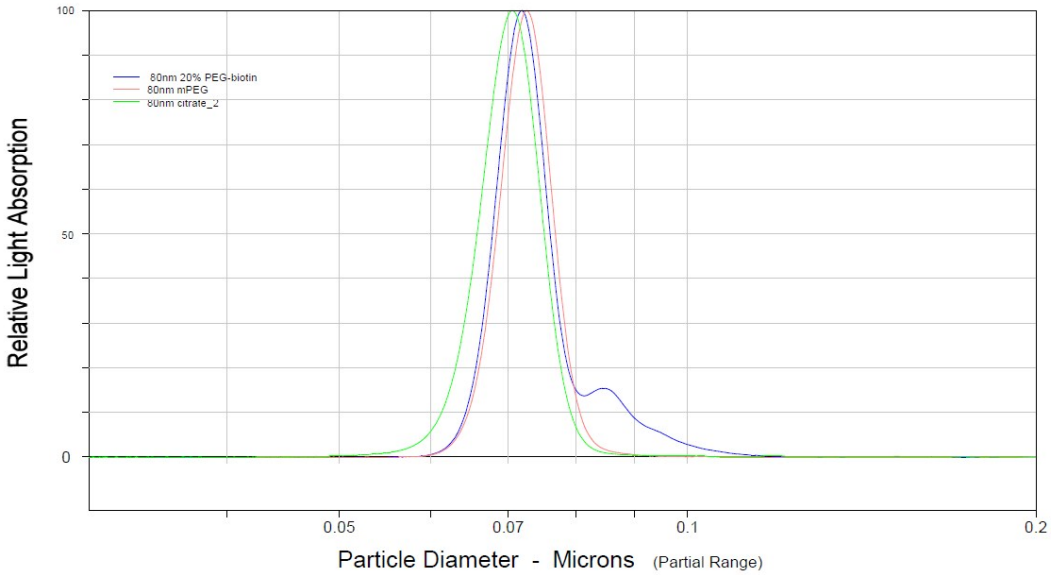

Figure S.5. Relative light absorption-based size distributions as measured by DCS of 80 nm gold nanoparticles coated with a mixture of mPEG and PEG-biotin molecules (blue), mPEG molecules (red) and citrate (green). The distributions are plotted assuming a density of the particles of 19.3 g/cm^3.

S5. Size analysis of agglomerated samples

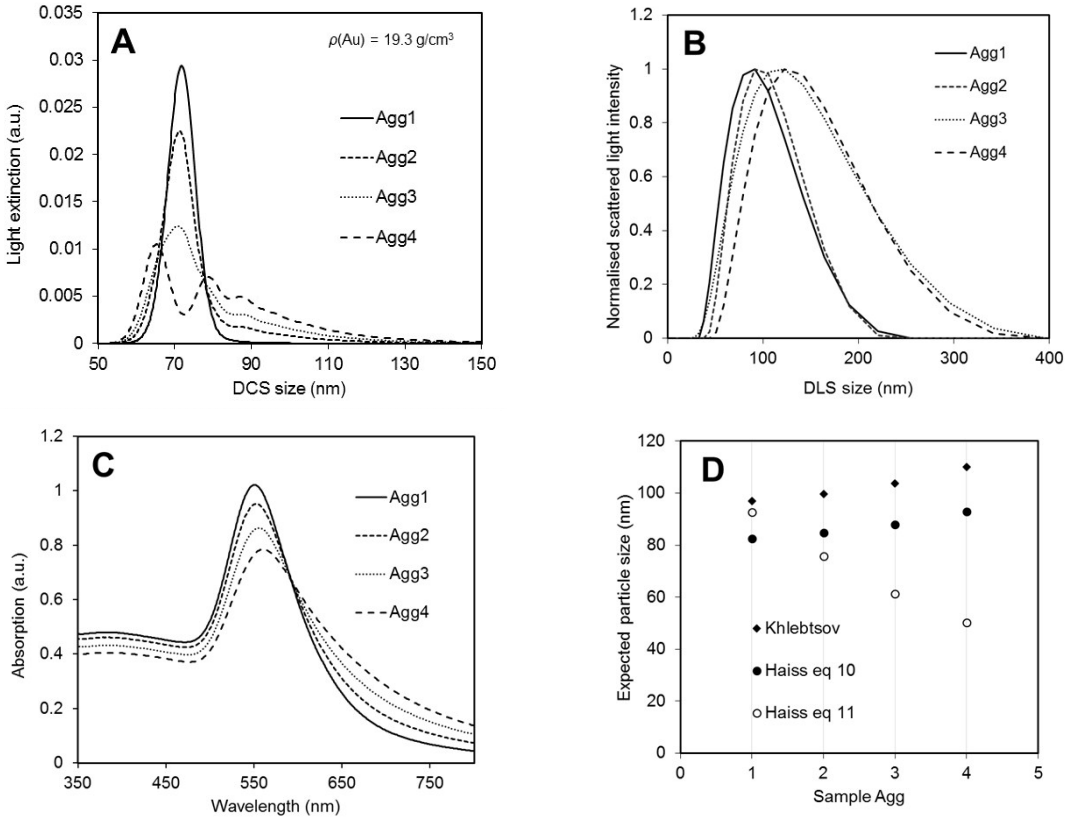

Figure S.6. (A) Light-extinction based size distributions as measured by DCS (for particle density of 19.3 g/cm^3), (B) normalised scattered-light intensity based size distributions as measured by DLS and (C) UV-Vis adsorption spectra and of the samples described in Table 1. (D) Apparent size of the nanoparticles based on the measured UV-Vis spectra in Figure (C) and theories from Khlebtsov and Haiss et al.

Samples Agg1 to Agg4 are shown in Figure S.5A. The high resolution of the technique allows the identification of a prominent peak in all the size distributions which that relates to the non agglomerated particles. Samples Agg2, Agg3 and Agg4 also exhibit tails in their distributions which describe the agglomerated particles.

It is interesting to observe that when a technique with lower size resolution is used, for example DLS in Figure S4B, it is impossible to discriminate whether the distributions describe samples with increasing agglomeration or mean size. With previous knowledge of the nature of the sample, we evince from Figure S.5B that the samples Agg1 to Agg4 are increasingly agglomerated because both the mean and the width of the distributions increase, but no detail on the level of agglomeration can be extracted. However, some more quantitative information can be obtained when the DLS data is supported by the DCS distributions in Figure S.5A. For example, we know from Figure S.5A that Agg1 exhibits a

narrow size distribution and no level of agglomeration. In these conditions, and by modelling the particles as perfect rigid spheres, the Z-ave value of Agg1 measured by DLS is a precise measurement of the hydrodynamic diameter of the particles, which resulted in a value of (81.19 ± 0.02) nm.

Similarly to DLS, the information provided by the UV-Vis spectra of the samples shown in Figure S.5C can not be easily interpreted in terms of sample agglomeration. If the spectra are interpreted as those of monodisperse gold nanoparticles, the corresponding diameter of these particles can be computed according to the empirical calibration curve obtained by Khlebtsov, as shown by dark diamonds in Figure S.5D. The diameter of the particles can also be computed according to Haiss et al as shown by the black and open circles in the same figure. Equation 11 in Haiss et al (open circles) results in opposite trend with respect to the other two methods. This difference is due to the first two methods utilising the wavelegth of the LSPR for computation of diameter, while Haiss' Equation 11 estimate is based on the ratio of the absorptions measured at the LSPR over that at 450 nm. Figure S.5C clearly shows that the LSPR wavelengths increase going from Agg1 to Agg4, i.e. increasing sample agglomeration. The LSPR peak broadens and decreases in intensity accordingly, while the adsoption at 450 nm varies to a lower extent. The disparity between these methods may therefore be used to identify agglomeration or other non-sphericity in gold particle populations.

The DCS traces may be fitted with excellent precision using normal distribution functions to represent different particle populations, as shown in Figure S.6. Here, seven populations are used but only the primary, doublet, triplet and tetramer populations can be identified with any certainty. The fit enables us to confirm that, in the mixed samples Agg2 and Agg3, the data can be described as a linear combination of the Agg1 and Agg4 results with relative weightings very close to the volumetric mixing ratios. Therefore no further interactions occur between particle populations upon mixing. Furthermore, the relative mass concentrations of dimer to monomer can be extracted from the fits.

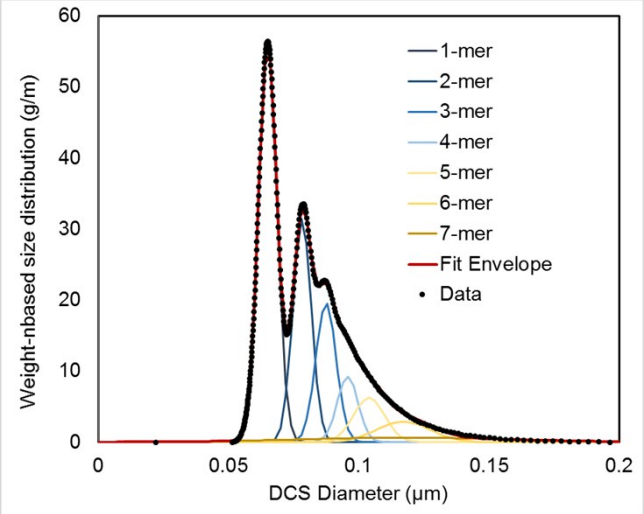

Figure S.7. The distribution of sample Agg 4 fitted by a sum of 7 normal distributions as shown in the figure, which are nominally assigned to different aggregation states.

S6: MATLAB script

Fortran code available from a NASA website was translated into MATLAB [http://www.giss.nasa.gov/staff/mmischenko/t_matrix.html]. For purposes of reproducibility the code is transcribed here. Each subroutine begins on a new page. Key input variables are in red text.

```
%C   CALCULATION OF LIGHT SCATTERING BY RANDOMLY ORIENTED TWO-SPHERE
%C   CLUSTERS WITH TOUCHING OR SEPARATED COMPONENTS
%C
%C   Last update 04/15/2000.
%C
%C   The code has been developed by Michael Mishchenko at the NASA
%C   Goddard Institute for Space Studies and Daniel Mackowski
%C   at Auburn University.
%
% The code was "translated" directly into Matlab at NPL by L. Wright. No
% attempt has been made to vectorize or otherwise improve the efficiency
% of the code. In addition, no attempts have been made to replace
% analytical formulae or iterative routines with direct Matlab functions.
%C
%C   The code can be used without any limitations in any not-for-
%C   profit scientific research.  The only request is that in any
%C   publication using the code the source of the code be acknowledged
%C   and relevant references be made.
%C
%C   The computational method is based on the superposition T-matrix
%C   approach:
%
%C   INPUTS:
%C
%C    LAM: wavelength of light
%C    R, N, K: radius and real and imaginary parts of the
%C          refractive index for each sphere.  Note that
%C          the component spheres may have different sizes and/or
%C          refractive indices.
%C    R12: distance between sphere centers.
%C         Touching spheres have R12=R(1)+R(2).
%C    NODR1:  a convergence parameter
%C       The number of multipole orders of each sphere are calculated
%C       using the formula of Wiscombe
%C          NODR(I)=NINT(XI(I)+4.*XI(I)**(1./3.)+NODR1)
%C       where XI(I) is the size parameter of sphere I.
%C       Usually NODR1=2 provides reliable results.  However,
%C       checking convergence over this parameter on a case-by-case
%C       basis is a good idea.
%C    NODRT1:  a convergence parameter
%C       The number of multipole orders for T^0 matrix expansion
%C       is calculated using the above formula for an equivalent
%C       sphere that completely encloses the two spheres.
```

```
%C       usually NODRT1=2 gives good results, but occasional
%C       checks of convergence over this parameter are recommended.
%C    NPNA: the number of equidistant scattering angles (from 0
%C       to 180 deg) for which the scattering matrix is
%C       calculated.
%C
%C
%C   OUTPUT:
%C
%C    CEXT: extinction cross section
%C    CSCA: scattering cross section
%C    W: single-scattering albedo
%C    <COS>: asymmetry parameter of the phase function
%C    ALPHA1,...,BETA2 - coefficients appearing in the expansions
%C         of the elements of the scattering matrix in
%C         generalized spherical functions
%C         [Eqs. (2.25)-(2.30) of Ref. 1].
%C    F11,...,F44: elements of the scattering matrix [as defined
%C          by equations (2.16)-(2.18) and (2.24) of Ref. 1]
%C          versus scattering angle
%
% Ref 1: 1. M. I. Mishchenko, Light scattering by randomly oriented axially
% symmetric particles.  J. Opt. Soc. Am. A, vol. 8, 871-882 (1991).

LAM=400e-9;% Wavelength of light in vacuum
N(1)=1.6178;  % real part of refractive index of first sphere
K(1)=1.9493; % imaginary part of refractive index of first sphere
N(2)=N(1);  % real part of refractive index of second sphere
K(2)=K(1); % imaginary part of refractive index of second sphere
R(1)=5.0e-9; % Radius of first sphere
R(2)=5.0e-9; % Radius of second sphere
R12=20.0e-9; % Separation of spheres
NODR1=2; % Convergence parameter
NODRT1=2; % Convergence parameter
NPNA=37; % Number of equispaced angles between 0 and 180 deg for which scattering matrix is
calculated.
%
% Adjust such that LAM is wavelength in the medium
% and N is ratio of refractive indices (so that n*XI will come out right)
%
NM=1.34; % Refractive index of the medium
N=N/NM;
K=K/NM;
LAM = LAM/NM; % Wavelength of light in the medium
%C
%C CALCULATE THE EXPANSION COEFFICIENTS
%C
PI=acos(-1.0);
XI(1)=2.0*PI*R(1)/LAM; % Dimensionless parameter for sphere 1
XI(2)=2.0*PI*R(2)/LAM; % Dimensionless parameter for sphere 2
X12=2.0*PI*R12/LAM;
% Calculate cross-sections and expansion coefficients
[QEXT2,QSCA2,QABS2,CEXT2,CSCA2,WALB,ASYMM,ALPH1,ALPH2,ALPH3,ALPH4,BET1,
BET2,L1MAX] = Bisphere(XI,X12,N,K,...
   NODR1,NODRT1,LAM);
```

```
% Carries out a test that may indicate a problem but does not guarantee
% correctness: expect KONTR2 = 1 if passed.
KONTR2=Hovenr(L1MAX,ALPH1,ALPH2,ALPH3,ALPH4,BET1,BET2);
%C
%C CALCULATE THE SCATTERING MATRIX
%C
LMAX=L1MAX-1;
% Calculates matrix.
[F11, F22, F33, F44, F12,
F34]=MatrBi(ALPH1,ALPH2,ALPH3,ALPH4,BET1,BET2,LMAX,NPNA);
%end
```

```
function XI = BesselFunc(N, DS)
%C
%      SUBROUTINE BESSEL(N,DS,XI)
%      IMPLICIT REAL*8(A-H,O-Z)
%      COMPLEX*16 XI(0:*)
%C
%C COMPUTES RICCATTI-BESSEL FUNCTION XI(0:N)=PSI(N)+I*CHI(N)
%C FOR REAL ARGUMENT DS
%
% Returns x*h_(n)(x) i.e. spherical Hankel*argument.
%
%C
%C CHECK IF RECURRENCE IS FORWARD OR BACKWARDS
%C
if (floor(DS)<N)
%C
%C BACKWARDS RECURRENCE FOR LOG. DERIVATIVE DN1(N)=PSI'(N)/PSI(N)
%C
     NS=floor(DS+4.0*(DS^0.3333)+17.0);
     DNS=0.D0;
     for I=NS-1:-1:N
       SN=(I+1)/DS;
       DNS=SN-1.0/(DNS+SN);
     end
     XI(N+1)=DNS;
     XI(N)=N/DS-1.0/(DNS+N/DS);
     for I=N-2:-1:0
       SN=(I+1)/DS;
       XI(I+1)=SN-1.D0/(XI(I+2)+SN);
     end
%C
%C FORWARD RECURRENCE: PSI(N) COMPUTED FROM DN1(N)
%C
     CHI0=-cos(DS);
     PSI=sin(DS);
     CHI1=CHI0/DS-PSI;
     XI(1)=PSI+sqrt(-1)*CHI0;
     for I=1:N
%      DO 30 I=1,N
       CHI2=(I+I+1)/DS*CHI1-CHI0;
       PSI=PSI/((I)/DS+XI(I+1));
       XI(I+1)=PSI+sqrt(-1)*CHI1;
       CHI0=CHI1;
       CHI1=CHI2;
     end
else
%C
%C FORWARD RECURRENCE: APPLICABLE ONLY IF DS > N
%C
     CHI0=-cos(DS);
     PSI0=sin(DS);
     CHI1=CHI0/DS-PSI0;
     PSI1=PSI0/DS+CHI0;
     XI(1)=PSI0+sqrt(-1)*CHI0;
     for I=1:N
```

```
        SN=(I+I+1)/DS;
        CHI2=SN*CHI1-CHI0;
        PSI2=SN*PSI1-PSI0;
        XI(I+1)=PSI1+sqrt(-1)*CHI1;
        CHI0=CHI1;
        CHI1=CHI2;
        PSI0=PSI1;
        PSI1=PSI2;
      end
end
```

```
function [QEXT,QSCA,QABS,CEXT,CSCA,WALB,ASYMM,
AL1,AL2,AL3,AL4,BE1,BE2,L1MAX] = Bisphere(XI,X12,SN,SK,...
   NODR1,NODRT1,LAM)
% This routine carries out the main calculations for the scattering
% coefficients and the cross-sections.
% See reference 1 for more details.
% NM is the refractive index of the medium.

% Set up parameters
CI=sqrt(-1.0);
PI=4.0*atan(1.0);
FAC=zeros(165,1);
FAC(1)=1.;
for N=1:165
   FAC(N+1)=(N)*FAC(N); % Factorials
end
NOPT=0;
NODR=zeros(1,2);
QE1=NODR; QS1=NODR; AN1=zeros(120,2); CN1=AN1;
for I=1:2
   NODR(I)=round(XI(I)+4.*XI(I)^(1./3.)+NODR1);
   %C
   %C  MIE1 CALCULATES THE MIE MULTIPOLE COEFFICIENTS FOR THE SPHERES
   %C
   [QEE1,QSS1,ANN1,CNN1] = Mie1(XI(I),SN(I),SK(I),NODR(I), 0);
   QE1(I)=QEE1;
   QS1(I)=QSS1;
   AN1(1:2*NODR(I),I)=ANN1;
   CN1(1:2*NODR(I),I)=CNN1;
end
R12 = X12;
RT=R12*0.50;
REFF=R12;
NODRT=round(REFF+4.*REFF^(1./3.)+NODRT1);
XV=(XI(1)^3+XI(2)^3)^(1./3.);
%C
%C HERE'S THE BEGINNING OF THE LOOP OVER DEGREE M
%C
QE=0.;
QS=0.;
CSCA=0.0;
CEXT=0.0;
for M=0:NODR(1)
   M1=max(1,M);
   MO=M-1;
   %C
   %C NBLK'S ARE THE DIMENSIONS OF THE MATRICES
   %C
   NBLK1=2*(NODR(1)-M1+1);
   NBLK2=2*(NODR(2)-M1+1);
   NBLKT=2*(NODRT-M1+1);
   NEQNS=NBLK1+NBLK2;
   for N=1:NEQNS
      for L=1:NEQNS
         AM(N,L)=0.;
```

```
      end
   end
   for N=M1:NODRT
      EN(N)=N*(N+1)/(N+N+1)*FAC(N+M+1)/FAC(N-M+1); % +1 added to FAC because indexing
is from 1 not 0 here
   end
   %C
   %C CHECK IF M HAS EXCEEDED NODR(2) -- IF SO, THE ELEMENTS OF T WILL
   %C BE GIVEN DIRECTLY BY THE TRANSLATED MIE COEFFICIENTS.
   %C
   if (M<=NODR(2))
      %C
      %C  COMPUTE SCALAR ADDITION COEFFICIENTS FOR MA
      %C
      if (MO==-1)||(abs(M)<MO)
      [CC, MO,CMR12,CNR12,CN1R12, NMIN] = VWHAC2_00(3,
R12,NODR(2),NODR(1),M,MO);
      else
         [CC, MO,CMR12,CNR12,CN1R12, NMIN] = VWHAC2_0(3,
R12,NODR(2),NODR(1),M,MO,CC,CMR12,CNR12,CN1R12);
      end
      %C
      %C  COMPUTE A MATRIX
      %C
      for N=M1:NODR(1)
         N1=N+N-1;
         N2=N1+1;
         I1=2*(N-M1)+1;
         I2=I1+1;
         AM(I1,I1)=1.;
         AM(I2,I2)=1.;
         for L=M1:NODR(2)
            J1=2*(L-M1)+1+NBLK1;
            J2=J1+1;
            INL=(-1)^(N+L);
            [A1, B1] = VWHAC2_1(R12,M,L,N,CC, NMIN, FAC);

            AM(I1,J1)=INL*AN1(N1,1)*A1;
            AM(I1,J2)=-INL*AN1(N1,1)*B1;
            AM(I2,J1)=-INL*AN1(N2,1)*B1;
            AM(I2,J2)=INL*AN1(N2,1)*A1;
         end
      end
      for N=M1:NODR(2)
         N1=N+N-1;
         N2=N1+1;
         I1=2*(N-M1)+1+NBLK1;
         I2=I1+1;
         AM(I1,I1)=1.;
         AM(I2,I2)=1.;
         for L=M1:NODR(1)
            J1=2*(L-M1)+1;
            J2=J1+1;
            [A1, B1] = VWHAC2_1(R12,M,L,N,CC, NMIN, FAC);
```

```
            AM(I1,J1)=AN1(N1,2)*A1;
            AM(I1,J2)=AN1(N1,2)*B1;
            AM(I2,J1)=AN1(N2,2)*B1;
            AM(I2,J2)=AN1(N2,2)*A1;
         end
      end
      %C
      %C  INVERT
      %C
      % Can probably be replaced by intrinsic routine.
      % May also be where the problem is.
      %
      % Mymat = inv(AM);
      AM = GJ2(AM, NBLK1, NEQNS) ;
      %C
      %C  MULT AM BY AN1
      %C
      for J=1:NBLK1
         L=J+2*(M1-1);
         for I=1:NEQNS
            AM(I,J)=AM(I,J)*AN1(L,1);
         end
      end
      for J=1:NBLK2
         L=J+2*(M1-1);
         for I=1:NEQNS
            AM(I,J+NBLK1)=AM(I,J+NBLK1)*AN1(L,2);
         end
      end
   end
   %C
   %C  CALCULATE J ADDITION COEFFICIENTS - TRANSLATION OVER
   %C  DISTANCE OF RT
   %C
   MO=M-1;
      if (MO==-1)||(abs(M)<MO)
         [CC, MO,CMRT,CNRT,CN1RT, NMIN] = VWHAC2_00(3, RT,NODR(1),NODRT,M,MO);
      else
         [CC, MO,CMRT,CNRT,CN1RT, NMIN] = VWHAC2_0(3,
RT,NODR(1),NODRT,M,MO,CC,CMRT,CNRT,CN1RT);
      end
   for N=M1:NODR(1)
      for L=M1:NODRT
         [A1, B1] = VWHAC2_1(RT,M,L,N,CC, NMIN, FAC);
         JM(N,L,1)=real(A1);
         JM(N,L,2)=-real(CI*B1);
      end
   end
   %C
   %C  TRANSLATE THE ELEMENTS OF THE T^{IJ} MATRIX TO THE CLUSTER
   %C  ORIGIN TO OBTAIN T^0
   %C
   %C
   %C  THIS BLOCK DOES TRANSLATION TO THE MIE COEFFICIENTS OF SPHERE 1
   %C  ONLY -- OCCURS FOR M > NODR(2)
```

```
%C
if(M>NODR(2))
   for I=1:NBLKT
         if mod(I,2)==1
            N=(I-1)/2+M1;
         else
            N=(I-2)/2+M1;
         end
      for J=1:NBLKT
         if mod(J,2)==1
            L=(J-1)/2+M1;
         else
            L=(J-2)/2+M1;
         end
         AT(I,J)=0.;
         IS=(-1)^(N+L);
         for K=1:NBLK1
         if mod(K,2)==1
            NP=(K-1)/2+M1;
         else
            NP=(K-2)/2+M1;
         end
            if(mod(K+I,2)==0)
               A1=JM(NP,N,1)*EN(NP)/EN(N);
            else
               A1=CI*JM(NP,N,2)*EN(NP)/EN(N);
            end
            if(mod(K+J,2)==0)
               A1=A1*JM(NP,L,1);
            else
               A1=-A1*CI*JM(NP,L,2);
            end
            AT(I,J)=AT(I,J)+AN1(K+2*(M1-1),1)*A1*IS;
         end
      end
   end
else
   %C
   %C THIS BLOCK TRANSLATES THE FULL T-MATRIX
   %C
   for I=1:NEQNS
      for J=1:NBLK1
         AMT(J,1)=AM(I,J);
         AMT(J,2)=0.;
      end
      for J=1:NBLK2
         AMT(J,2)=AM(I,J+NBLK1);
      end
      for J=1:NBLKT
         AM(I,J)=0.;
         if mod(J,2)==1
         L=(J-1)/2+M1;
         else
            L=(J-2)/2+M1;
         end
```

```
          for K=1:NBLK1
          if mod(K,2)==1
          LP=(K-1)/2+M1;
          else
             LP=(K-2)/2+M1;
          end
             IS=(-1)^(L+LP);
             if(mod(K+J,2)==0)
                A=JM(LP,L,1);
             else
                A=CI*JM(LP,L,2);
                IS=-IS;
             end
             AM(I,J)=AM(I,J)+(AMT(K,1)*IS+AMT(K,2))*A;
          end
       end
    end
    for J=1:NBLKT
       for I=1:NBLK1
          AMT(I,1)=AM(I,J);
          AMT(I,2)=0.;
       end
       for I=1:NBLK2
          AMT(I,2)=AM(I+NBLK1,J);
       end
       for I=1:NBLKT
          AT(I,J)=0.;
          if mod(I,2)==1
          N=(I-1)/2+M1;
          else
             N=(I-2)/2+M1;
          end
          for K=1:NBLK1
          if mod(K,2)==1
          NP=(K-1)/2+M1;
          else
             NP=(K-2)/2+M1;
          end
             IS=(-1)^(N+NP);
             if(mod(K+I,2)==0)
                A=JM(NP,N,1)*EN(NP)/EN(N);
             else
                A=-CI*JM(NP,N,2)*EN(NP)/EN(N);
                IS=-IS;
             end
             AT(I,J)=AT(I,J)+(AMT(K,1)*IS+AMT(K,2))*A;
          end
       end
    end
end
%C
%C  COMPUTE RANDOM-ORIENTATION EXTINCTION, SCATTERING
%C
QEM=0.;
QSM=0.;
```

```
   for N=M1:NODRT
      I1=2*(N-M1)+1;
      I2=I1+1;
      N1=N+N-1;
      N2=N1+1;
      QEM=QEM+AT(I1,I1)+AT(I2,I2);
      for L=M1:NODRT
         J1=2*(L-M1)+1;
         J2=J1+1;
         QSM=QSM+EN(N)*(AT(I1,J1)*(AT(I1,J1)')+AT(I1,J2)*(AT(I1,J2)'))/EN(L);
         QSM=QSM+EN(N)*(AT(I2,J1)*(AT(I2,J1)') +AT(I2,J2)*(AT(I2,J2)'))/EN(L);
      end
   end
   if(M==0)
      QE=QE+QEM;
      QS=QS+QSM;
   else
      QE=QE+2.*QEM;
      QS=QS+2.*QSM;
   end
   NM=NODRT-M1+1;
   MIND=M+1;
   for N2=1:NM
      NN2=N2+M1-1;
      N22=(N2-1)*2+1;
      for N1=1:NM
         NN1=N1+M1-1;
         N11=(N1-1)*2+1;
         EPSC=-sqrt(EN(NN1)/EN(NN2));
         ZZ1=real(AT(N11+1,N22+1)*EPSC);
         RT11(MIND,NN1,NN2)=ZZ1;
         ZZ2=real(AT(N11+1,N22+1)*(-CI)*EPSC);
         IT11(MIND,NN1,NN2)=ZZ2;
         ZZ3=real(AT(N11+1,N22)*EPSC);
         RT12(MIND,NN1,NN2)=ZZ3;
         ZZ4=real(AT(N11+1,N22)*(-CI)*EPSC);
         IT12(MIND,NN1,NN2)=ZZ4;
         ZZ5=real(AT(N11,N22+1)*EPSC);
         RT21(MIND,NN1,NN2)=ZZ5;
         ZZ6=real(AT(N11,N22+1)*(-CI)*EPSC);
         IT21(MIND,NN1,NN2)=ZZ6;
         ZZ7=real(AT(N11,N22)*EPSC);
         RT22(MIND,NN1,NN2)=ZZ7;
         ZZ8=real(AT(N11,N22)*(-CI)*EPSC);
         IT22(MIND,NN1,NN2)=ZZ8;
         if (M==0)

CSCA=CSCA+(ZZ1*ZZ1+ZZ2*ZZ2+ZZ3*ZZ3+ZZ4*ZZ4+ZZ5*ZZ5+ZZ6*ZZ6+ZZ7*ZZ7+ZZ8*
ZZ8);
         else

CSCA=CSCA+(ZZ1*ZZ1+ZZ2*ZZ2+ZZ3*ZZ3+ZZ4*ZZ4+ZZ5*ZZ5+ZZ6*ZZ6+ZZ7*ZZ7+ZZ8*
ZZ8)*2.0;
         end
      end
```

```
   end
   for N=M1:NODRT
      if (M==0)
         CEXT=CEXT+RT11(MIND,N,N)+RT22(MIND,N,N);
      else
         CEXT=CEXT+2.0*(RT11(MIND,N,N)+RT22(MIND,N,N));
      end
   end
   %C
   %C LOOP ON TO THE NEXT DEGREE M
   %C
end
%       C
%       C CALCULATE THE EXPANSION COEFFICIENTS
%       C

DLAM=2.0*PI;
COEFF1=DLAM*DLAM*0.50/PI;
CSCA=CSCA*COEFF1;
CEXT=-CEXT*COEFF1;

% Get the scattering coefficients from the T matrices
[AL1,AL2,AL3,AL4,BE1,BE2,LMAX] =
GSP(NODRT,NODR(1),CSCA,DLAM,RT11,RT12,RT21,RT22,IT11,IT12,IT21,IT22);
L1MAX=LMAX+1;
COEFF1=LAM*LAM*0.250/(PI*PI);
CSCA=CSCA*COEFF1;
CEXT=CEXT*COEFF1;
WALB=CSCA/CEXT;
ASYMM=AL1(2)/3.0;
XI1=XI(1);
XI2=XI(2);
REV=(XI1*XI1*XI1+XI2*XI2*XI2)^(1.0/3.0);
COEFF1=1.0/(PI*REV*REV);
% Factors calculated based on a volume-averaged radius.
QSCA=CSCA*COEFF1;
QEXT=CEXT*COEFF1;
QABS=QEXT-QSCA;
end
```

```
%C*****************************************************************
%C
%C  CALCULATION OF CLEBSCH-GORDAN COEFFICIENTS
%C  (N,M:N1,M1/NN,MM)
%C  FOR GIVEN N AND N1. M1=MM-M, INDEX MM IS FOUND FROM M AS
%C  MM=M*K1+K2
%C
%C  INPUT PARAMETERS :  N,N1,NMAX,K1,K2
%C                      N.LE.NMAX
%C                      N.GE.1
%C                      N1.GE.0
%C                      N1.LE.N+NMAX
%C  OUTPUT PARAMETERS : GG(M+NPN6,NN+1) - ARRAY OF THE CORRESPONDING
%C                           COEFFICIENTS
%C                      /M/.LE.N
%C                      /M1/=/M*(K1-1)+K2/.LE.N1
%C                      NN.LE.MIN(N+N1,NMAX)
%C                      NN.GE.MAX(/MM/,/N-N1/)
%C  IF K1=1 AND K2=0, THEN 0.LE.M.LE.N
%     REAL*8 GG(NPL1,NPN6),CD(0:NPN5),CU(0:NPN5) Needs care.

function [GG, CU, CD]=CCG(N,N1,NMAX,K1,K2,F, SSIGN, GG, CU, CD)
NOTD=85;
NPN4=NOTD;
NPN5=2*NPN4;
NPN6=NPN4+1;
NPL1=NPN5+1;
if (NMAX<=NPN4)&&(0<=N1)&&(N1<=NMAX+N)&&(N>=1)&&(N<=NMAX)
   NNF=min(N+N1,NMAX);
   MIN=NPN6-N;
   MF=NPN6+N;
   if(K1==1)&&(K2==0); MIN=NPN6; end
   for MIND=MIN:MF
      M=MIND-NPN6;
      MM=M*K1+K2;
      M1=MM-M;
      if (abs(M1)<=N1)
         NNL=max(abs(MM),abs(N-N1));
         if (NNL<=NNF)
            NNU=N+N1;
            if mod(NNU+NNL,2) ==0
               NNM=(NNU+NNL)/2;
            else
               NNM = (NNU+NNL-1)/2;
            end
            if (NNU==NNL); NNM=NNL; end
            [MM, M, N, N1, C] = CCGIN(MM, M, N, N1, F, SSIGN);
            CU(NNL+1)=C; % +1 added to shift indexing to start from 1 not 0.
            if (NNL~=NNF)
               C2=0.0;
               C1=C;
               for NN=NNL+1:min(NNM,NNF)
                  A=((NN+MM)*(NN-MM)*(N1-N+NN));
                  A=A*((N-N1+NN)*(N+N1-NN+1)*(N+N1+NN+1));
                  A=(4*NN*NN)/A;
```

```
                A=A*((2*NN+1)*(2*NN-1));
                A=sqrt(A);
                B=0.50*(M-M1);
                D=0.0;

                if (NN~=1)
                   B=(2*NN*(NN-1));
                   B=((2*M-MM)*NN*(NN-1)-MM*N*(N+1)+MM*N1*(N1+1))/B;
                   D=(4*(NN-1)*(NN-1));
                   D=D*((2*NN-3)*(2*NN-1));
                   D=((NN-MM-1)*(NN+MM-1)*(N1-N+NN-1))/D;
                   D=D*((N-N1+NN-1)*(N+N1-NN+2)*(N+N1+NN));
                   D=sqrt(D);
                end
                C=A*(B*C1-D*C2);
                C2=C1;
                C1=C;
                CU(NN+1)=C; % +1 added to shift indexing to start from 1 not 0.
             end
             if (NNF>NNM)
                C= Direct(N,M,N1,M1,NNU,MM,F);
                CD(NNU+1)=C; % +1 added to shift indexing to start from 1 not 0.
                if (NNU~=NNM+1)
                   C2=0.0;
                   C1=C;
                   for NN=NNU-1:-1:NNM+1
                      A=((NN-MM+1)*(NN+MM+1)*(N1-N+NN+1));
                      A=A*((N-N1+NN+1)*(N+N1-NN)*(N+N1+NN+2));
                      A=(4*(NN+1)*(NN+1))/A;
                      A=A*((2*NN+1)*(2*NN+3));
                      A=sqrt(A);
                      B=(2*(NN+2)*(NN+1));
                      B=((2*M-MM)*(NN+2)*(NN+1)-MM*N*(N+1)+MM*N1*(N1+1))/B;
                      D=(4*(NN+2)*(NN+2));
                      D=D*((2*NN+5)*(2*NN+3));
                      D=((NN+MM+2)*(NN-MM+2)*(N1-N+NN+2))/D;
                      D=D*((N-N1+NN+2)*(N+N1-NN-1)*(N+N1+NN+3));
                      D=sqrt(D);
                      C=A*(B*C1-D*C2);
                      C2=C1;
                      C1=C;
                      CD(NN+1)=C; % +1 added to shift indexing to start from 1 not 0.
                   end
                end
             end
          end
          for NN=NNL:NNF
             if (NN<=NNM); GG(MIND,NN+1)=CU(NN+1); end
             if (NN>NNM); GG(MIND,NN+1)=CD(NN+1); end
          end
       end
    end
 end
    function CC = Direct(N,M,N1,M1,NN,MM, F)
```

```
        CC=F(2*N+1)+F(2*N1+1)+F(N+N1+M+M1+1)+F(N+N1-M-M1+1);
        CC=CC-F(2*(N+N1)+1)-F(N+M+1)-F(N-M+1)-F(N1+M1+1)-F(N1-M1+1);
        CC=exp(CC);
    end
end
```

```
%     SUBROUTINE CCGIN(N,N1,M,MM,G)
%     IMPLICIT REAL*8 (A-H,O-Z)
%     REAL*8 F(500),SSIGN(600)
%     COMMON /SS/ SSIGN
%     COMMON /FAC/ F
function [MM, M, N, N1, G] = CCGIN(MM, M, N, N1, F, SSIGN)
M1=MM-M;
if (((N>=abs(M))&&(N1>=abs(M1)))&&(abs(MM)<=N+N1))
   if (abs(MM)<=abs(N-N1))
      L1=N;
      L2=N1;
      L3=M;
      if (N1>N)
         K=N;
         N=N1;
         N1=K;
         K=M;
         M=M1;
         M1=K;
      end
      N2=N*2;
      M2=M*2;
      N12=N1*2;
      M12=M1*2;
      G=SSIGN(N1+M1+1)*exp(F(N+M+1)+F(N-M+1)+F(N12+1)+F(N2-N12+2)-F(N2+2) ...
           -F(N1+M1+1)-F(N1-M1+1)-F(N-N1+MM+1)-F(N-N1-MM+1));
      N=L1;
      N1=L2;
      M=L3;
   else
      A=1.0;
      L1=M;
      L2=MM;
      if (MM<0)
         MM=-MM;
         M=-M;
         M1=-M1;
         A=SSIGN(MM+N+N1+1);
      end
      G=A*SSIGN(N+M+1)*exp(F(2*MM+2)+F(N+N1-MM+1)+F(N+M+1)+F(N1+M1+1)-...
         F(N+N1+MM+2)-F(N-N1+MM+1)-F(-N+N1+MM+1)-F(N-M+1)-F(N1-M1+1));
      M=L1;
      MM=L2;
   end
end
```

```
%PARAMETER(NOD=60,NMAX=2*NOD)
%COMPLEX*16 A(NP,NP),AT(NMAX)
%C
%C GAUSS ELIMINATION OF A 2-SPHERE INTERACTION MATRIX, IN THE FORM
%C
%C       I  A^12
%C  A =
%C      A^21 I
%C
%C  CALC T^11 = [I-A^12 A^21]^-1
%C
function A = GJ2(A, N1, N)

for I=1:N1
   for J=1:N1
      for K=N1+1:N
         A(I,J)=A(I,J)-A(I,K)*A(K,J);
      end
   end
end
A = GaussJ(A, N1);
for I=N1+1:N
   for J=1:N1
      AT(J) = A(I,J);
      A(I,J) = 0.0;
   end
   for J=1:N1
      for K=1:N1
         A(I,J)=A(I,J)-AT(K)*A(K,J);
      end
   end
end
for I=N1+1:N
   for J=N1+1:N
      for K=1:N1
         A(I,J) = A(I,J)-A(I,K)*A(K,J);
      end
   end
end
for J=N1+1:N
   for I=1:N1
      AT(I) = A(I,J);
      A(I,J) = 0.0;
   end
   for I=1:N1
      for K=1:N1
         A(I,J) = A(I,J) - A(I,K)*AT(K);
      end
   end
end

%SUBROUTINE GAUSSJ(A,N,NP)
%IMPLICIT REAL*8(A-H,O-Z)
%PARAMETER (NOD=60,NMAX=2*NOD)
%COMPLEX*16 A(NP,NP),DUM,PIVINV
```

```
%DIMENSION IPIV(NMAX),INDXR(NMAX),INDXC(NMAX)
   function A=GaussJ(A, N)
      IPIV=zeros(N,1);
      for II=1:N
         BIG=0.;
         for JJ=1:N
            if (IPIV(JJ)~=1)
               for KK=1:N
                  if (IPIV(KK)==0)
                  if (abs(A(JJ,KK))>=BIG)
                     BIG=abs(A(JJ,KK));
                     IROW=JJ;
                     ICOL=KK;
                  end
                  end
               end
            end
         end
         IPIV(ICOL)=IPIV(ICOL)+1;
         if (IROW~=ICOL)
            for L=1:N
               DUM=A(IROW,L);
               A(IROW,L)=A(ICOL,L);
               A(ICOL,L)=DUM;
            end
         end
         INDXR(II)=IROW;
         INDXC(II)=ICOL;
         PIVINV=1./A(ICOL,ICOL);
         A(ICOL,ICOL)=1.;
         for L=1:N
            A(ICOL,L)=A(ICOL,L)*PIVINV;
         end
         for LL=1:N
            if (LL~=ICOL)
               DUM=A(LL,ICOL);
               A(LL,ICOL)=0.;
               for L=1:N
                  A(LL,L)=A(LL,L)-A(ICOL,L)*DUM;
               end
            end
         end
      end
      for L=N:-1:1
         if INDXR(L)~=INDXC(L)
            for KK=1:N
               DUM=A(KK,INDXR(L));
               A(KK,INDXR(L))=A(KK,INDXC(L));
               A(KK,INDXC(L))=DUM;
            end
         end
      end
   end
end
```

```
%C*****************************************************************
%C                                                                *
%C  CALCULATION OF THE EXPANSION COEFFICIENTS FOR (I,Q,U,V) -     *
%C  REPRESENTATION.                                              *
%C                                                                *
%C  INPUT PARAMETERS:                                            *
%C                                                                *
%C    LAM - WAVELENGTH OF LIGHT                                   *
%C    CSCA - SCATTERING CROSS SECTION                             *
%C    TR AND TI - ELEMENTS OF THE T-MATRIX.                       *
%C    NMAX - DIMENSION OF T(M)-MATRICES                           *
%C    MORD - MAXIMUM M-NUMBER                                     *
%C                                                                *
%C  OUTPUT INFORTMATION:                                          *
%C                                                                *
%C    ALF1,...,ALF4,BET1,BET2 - EXPANSION COEFFICIENTS            *
%C    LMAX - NUMBER OF COEFFICIENTS MINUS 1                       *
%C                                                                *
%C*****************************************************************

function [ALF1,ALF2,ALF3,ALF4,BET1,BET2,LMAX] =
GSP(NMAX,MORD,CSCA,LAM,TR11,TR12,TR21,TR22,TI11,TI12,TI21,TI22)

NOTD=85;
NPN4=NOTD;
NPN5=2*NPN4;
NPN6=NPN4+1;
NPL1=NPN5+1;
G1 = zeros(NPL1, NPN6); G2 = G1;
CU = zeros(NPN5+1,1); CD = CU;
F = FactNum;
SSIGN = Signum;
LMAX=2*NMAX;
L1MAX=LMAX+1;
CI=sqrt(-1);
CIM(1)=CI;
for I=2:NMAX
   CIM(I)=CIM(I-1)*CI;
end
SSI(1)=1.0;
for I=1:LMAX
   I1=I+1;
   SI=(2*I+1);
   SSI(I1)=SI;
   if(I<=NMAX); SSJ(I)=sqrt(SI); end
end
CI=-CI;
for I=1:NMAX
   SI=SSJ(I);
   CCI=CIM(I);
   for J=1:NMAX
      SJ=1.0/SSJ(J);
      CCJ=CIM(J)*SJ/CCI;
      FR(J,I)=real(CCJ);
      FI(J,I)=real(CCJ*CI);
```

```
      FF(J,I)=real(SI*SJ);
   end
end
NMAX1=NMAX+1;
%
%C *****  CALCULATION OF THE ARRAYS B1 AND B2  *****
%
K1=1;
K2=0;
K3=0;
K4=1;
K5=1;
K6=2;
for N=1:NMAX
   %C *****  CALCULATION OF THE ARRAYS T1 AND T2  *****
   for NN=1:NMAX
      M1MAX=min([N,NN,MORD])+1;
      for M1=1:M1MAX
         M=M1-1;
         L1=NPN6+M;
         TT1=TR11(M1,N,NN);
         TT2=TR12(M1,N,NN);
         TT3=TR21(M1,N,NN);
         TT4=TR22(M1,N,NN);
         TT5=TI11(M1,N,NN);
         TT6=TI12(M1,N,NN);
         TT7=TI21(M1,N,NN);
         TT8=TI22(M1,N,NN);
         T1=TT1+TT2;
         T2=TT3+TT4;
         T3=TT5+TT6;
         T4=TT7+TT8;
         TR1(L1,NN)=T1+T2;
         TR2(L1,NN)=T1-T2;
         TI1(L1,NN)=T3+T4;
         TI2(L1,NN)=T3-T4;
         if (M~=0)
            L1=NPN6-M;
            T1=TT1-TT2;
            T2=TT3-TT4;
            T3=TT5-TT6;
            T4=TT7-TT8;
            TR1(L1,NN)=T1-T2;
            TR2(L1,NN)=T1+T2;
            TI1(L1,NN)=T3-T4;
            TI2(L1,NN)=T3+T4;
         end
      end
   end
   %
   %C  *****  END OF THE CALCULATION OF THE ARRAYS T1 AND T2  *****
   %
   NN1MAX=NMAX1+N;
   for NN1=1:NN1MAX
      N1=NN1-1;
```

```
%
%C  *****  CALCULATION OF THE ARRAYS A1 AND A2  *****
%
[G1, CU, CD]=CCG(N,N1,NMAX,K1,K2,F,SSIGN, G1, CU, CD);
NNMAX=min(NMAX,N1+N);
NNMIN=max(1,abs(N-N1));
KN=N+NN1;
for NN=NNMIN:NNMAX
   NNN=NN+1;
   SIG=SSIGN(KN+NN);
   M1MAX=min(N,min(NN,MORD))+NPN6;
   AAR1=0.0;
   AAR2=0.0;
   AAI1=0.0;
   AAI2=0.0;
   for M1=NPN6:M1MAX
      M=M1-NPN6;
      SSS=G1(M1,NNN);
      RR1=TR1(M1,NN);
      RI1=TI1(M1,NN);
      RR2=TR2(M1,NN);
      RI2=TI2(M1,NN);
      if (M~=0)
         M2=NPN6-M;
         RR1=RR1+TR1(M2,NN)*SIG;
         RI1=RI1+TI1(M2,NN)*SIG;
         RR2=RR2+TR2(M2,NN)*SIG;
         RI2=RI2+TI2(M2,NN)*SIG;
      end
      AAR1=AAR1+SSS*RR1;
      AAI1=AAI1+SSS*RI1;
      AAR2=AAR2+SSS*RR2;
      AAI2=AAI2+SSS*RI2;

   end
   XR=FR(NN,N);
   XI=FI(NN,N);
   AR1(NN)=AAR1*XR-AAI1*XI;
   AI1(NN)=AAR1*XI+AAI1*XR;
   AR2(NN)=AAR2*XR-AAI2*XI;
   AI2(NN)=AAR2*XI+AAI2*XR;
end
%
%C  *****  END OF THE CALCULATION OF THE ARRAYS A1 AND A2 ****
%
[G2, CU, CD]=CCG(N,N1,NMAX,K3,K4,F,SSIGN, G2, CU, CD);
M1=max(-N1+1,-N);
M2=min(N1+1,N);
M1MAX=M2+NPN6;
M1MIN=M1+NPN6;
for M1=M1MIN:M1MAX
   BBR1=0.0;
   BBI1=0.0;
   BBR2=0.0;
   BBI2=0.0;
```

```
      for NN=NNMIN:NNMAX
         NNN=NN+1;
         SSS=G2(M1,NNN);
         BBR1=BBR1+SSS*AR1(NN);
         BBI1=BBI1+SSS*AI1(NN);
         BBR2=BBR2+SSS*AR2(NN);
         BBI2=BBI2+SSS*AI2(NN);
      end
      B1R(NN1,M1,N)=BBR1;
      B1I(NN1,M1,N)=BBI1;
      B2R(NN1,M1,N)=BBR2;
      B2I(NN1,M1,N)=BBI2;
   end
 end
end
%C  *****  END OF THE CALCULATION OF THE ARRAYS B1 AND B2 ****
%
%C  *****  CALCULATION OF THE ARRAYS D1,D2,D3,D4, AND D5  *****
%
for N=1:NMAX
 for NN=1:NMAX
   M1=min(N,NN);
   M1MAX=NPN6+M1;
   M1MIN=NPN6-M1;
   NN1MAX=NMAX1+min(N,NN);
   for M1=M1MIN:M1MAX
      M=M1-NPN6;
      NN1MIN=abs(M-1)+1;
      DD1=0.0;
      DD2=0.0;
      for NN1=NN1MIN:NN1MAX
         XX=SSI(NN1);
         X1=B1R(NN1,M1,N);
         X2=B1I(NN1,M1,N);
         X3=B1R(NN1,M1,NN);
         X4=B1I(NN1,M1,NN);
         X5=B2R(NN1,M1,N);
         X6=B2I(NN1,M1,N);
         X7=B2R(NN1,M1,NN);
         X8=B2I(NN1,M1,NN);
         DD1=DD1+XX*(X1*X3+X2*X4);
         DD2=DD2+XX*(X5*X7+X6*X8);
      end
      D1(M1,NN,N)=DD1;
      D2(M1,NN,N)=DD2;
   end
   MMAX=min(N,NN+2);
   MMIN=max(-N,-NN+2);
   M1MAX=NPN6+MMAX;
   M1MIN=NPN6+MMIN;
   for M1=M1MIN:M1MAX
      M=M1-NPN6;
      NN1MIN=abs(M-1)+1;
      DD3=0.0;
      DD4=0.0;
```

```
            DD5R=0.0;
            DD5I=0.0;
            M2=-M+2+NPN6;
            for NN1=NN1MIN:NN1MAX
               XX=SSI(NN1);
               X1=B1R(NN1,M1,N);
               X2=B1I(NN1,M1,N);
               X3=B2R(NN1,M1,N);
               X4=B2I(NN1,M1,N);
               X5=B1R(NN1,M2,NN);
               X6=B1I(NN1,M2,NN);
               X7=B2R(NN1,M2,NN);
               X8=B2I(NN1,M2,NN);
               DD3=DD3+XX*(X1*X5+X2*X6);
               DD4=DD4+XX*(X3*X7+X4*X8);
               DD5R=DD5R+XX*(X3*X5+X4*X6);
               DD5I=DD5I+XX*(X4*X5-X3*X6);
            end
            D3(M1,NN,N)=DD3;
            D4(M1,NN,N)=DD4;
            D5R(M1,NN,N)=DD5R;
            D5I(M1,NN,N)=DD5I;
         end
      end
   end
%C  *****  END OF THE CALCULATION OF THE D-ARRAYS *****
%
%C  *****  CALCULATION OF THE EXPANSION COEFFICIENTS *****
%
DK=LAM*LAM/(4.0*CSCA*acos(-1.0));
G1L=1.0;
L1=1;
while (abs(G1L)>1e-5)&&(L1<=L1MAX)

      G1L=0.0;
      G2L=0.0;
      G3L=0.0;
      G4L=0.0;
      G5LR=0.0;
      G5LI=0.0;
      L=L1-1;
      SL=SSI(L1)*DK;
      for N=1:NMAX
         NNMIN=max(1,abs(N-L));
         NNMAX=min(NMAX,N+L);
         if (NNMAX>=NNMIN)
            [G1, CU, CD]=CCG(N,L,NMAX,K1,K2,F,SSIGN, G1, CU, CD);
            if (L>=2); [G2, CU, CD]=CCG(N,L,NMAX,K5,K6,F, SSIGN, G2, CU, CD); end;
            NL=N+L;
            for  NN=NNMIN:NNMAX
               NNN=NN+1;
               MMAX=min(N,NN);
               M1MIN=NPN6-MMAX;
               M1MAX=NPN6+MMAX;
               SI=SSIGN(NL+NNN);
```

```
            DM1=0.0;
            DM2=0.0;
            for M1=M1MIN:M1MAX
               M=M1-NPN6;
               if(M>=0)
                  SSS1=G1(M1,NNN);
               else
                  SSS1=G1(NPN6-M,NNN)*SI;
               end
               DM1=DM1+SSS1*D1(M1,NN,N);
               DM2=DM2+SSS1*D2(M1,NN,N);
            end
            FFN=FF(NN,N);
            SSS=G1(NPN6+1,NNN)*FFN;
            G1L=G1L+SSS*DM1;
            G2L=G2L+SSS*DM2*SI;
            if (L>=2)
               DM3=0.0;
               DM4=0.0;
               DM5R=0.0;
               DM5I=0.0;
               MMAX=min(N,NN+2);
               MMIN=max(-N,-NN+2);
               M1MAX=NPN6+MMAX;
               M1MIN=NPN6+MMIN;
               for M1=M1MIN:M1MAX
                  M=M1-NPN6;
                  SSS1=G2(NPN6-M,NNN);
                  DM3=DM3+SSS1*D3(M1,NN,N);
                  DM4=DM4+SSS1*D4(M1,NN,N);
                  DM5R=DM5R+SSS1*D5R(M1,NN,N);
                  DM5I=DM5I+SSS1*D5I(M1,NN,N);
               end
               G5LR=G5LR-SSS*DM5R;
               G5LI=G5LI-SSS*DM5I;
               SSS=G2(NPN4,NNN)*FFN;
               G3L=G3L+SSS*DM3;
               G4L=G4L+SSS*DM4*SI;
            end
         end
      end
   end
   G1L=G1L*SL;
   G2L=G2L*SL;
   G3L=G3L*SL;
   G4L=G4L*SL;
   G5LR=G5LR*SL;
   G5LI=G5LI*SL;
   ALF1(L1)=G1L+G2L;
   ALF2(L1)=G3L+G4L;
   ALF3(L1)=G3L-G4L;
   ALF4(L1)=G1L-G2L;
   BET1(L1)=G5LR*2.0;
   BET2(L1)=G5LI*2.0;
   LMAX=L;
```

```
   L1=L1+1;
end
%C*************************************************************
%
%C   CALCULATION OF THE ARRAY SSIGN(N+1)=SIGN(N)
%C   0.LE.N.LE.599

   function SSIGN = Signum
      SSIGN=ones(600,1);
      for II = 2:600
         SSIGN(II) = -SSIGN(II-1);
      end
   end
%C   CALCULATION OF THE QUANTITIES F(N+1)=0.5*LN(N!)
%C   0.LE.N.LE.499
   function F = FactNum
      F=zeros(500,1);
      for II=3:500
         F(II) = F(II-1)+0.5*log(II-1);
      end
   end
end
```

```
%c*************************************************************
%
% Carries out the using general inequalities derived by van der Mee and
% Hovenier, Astron. Astrophys., vol. 228, 559-568 (1990) as a correctness
% test
%
function KONTR=Hovenr(L1,A1, A2, A3, A4, B1, B2)

for L=1:L1
   KONTR=1;
   LL=L-1;
   DL=LL*2.0+1.0;
   DDL=DL*0.480;
   AA1=A1(L);
   AA2=A2(L);
   AA3=A3(L);
   AA4=A4(L);
   BB1=B1(L);
   BB2=B2(L);
   if (LL>=1)&&abs(AA1>=DL); KONTR=2; end
   if(abs(AA2)>=DL); KONTR=2; end
   if(abs(AA3)>=DL); KONTR=2; end
   if(abs(AA4)>=DL); KONTR=2; end
   if(abs(BB1)>=DDL); KONTR=2; end
   if(abs(BB2)>=DDL); KONTR=2; end
   %        if(KONTR.EQ.2) WRITE (1,3000) LL
   C=-0.10;
   for I=1:11

      C=C+0.10;
      CC=C*C;
      C1=CC*BB2*BB2;
      C2=C*AA4;
      C3=C*AA3;
      if((DL-C*AA1)*(DL-C*AA2)-CC*BB1*BB1<=-1e-6); KONTR=2; end
      if((DL-C2)*(DL-C3)+C1<=-1E-6); KONTR=2; end
      if((DL+C2)*(DL-C3)-C1<=-1E-6); KONTR=2; end
      if((DL-C2)*(DL+C3)-C1<=-1E-6); KONTR=2; end
   end
end
end
```

```
%C*************************************************************
%
%C    CALCULATION OF THE SCATTERING MATRIX FOR GIVEN EXPANSION
%C    COEFFICIENTS
%
%C    A1,...,B2 - EXPANSION COEFFICIENTS
%C    LMAX - NUMBER OF COEFFICIENTS MINUS 1
%C    NPNA - NUMBER OF SCATTERING ANGLES
%C       THE CORRESPONDING SCATTERING ANGLES ARE GIVEN BY
%C       180*(I-1)/(NPNA-1) (DEGREES), WHERE I NUMBERS THE ANGLES
function [F11, F22, F33, F44, F12, F34]=MatrBi(A1, A2, A3, A4, B1, B2, LMAX, NPNA)
    N=NPNA;
    DN=1.0/(N-1);
    DA=acos(-1.0)*DN;
    DB=180.0*DN;
    L1MAX=LMAX+1;
    TB=-DB;
    TAA=-DA;
    D6=sqrt(6.0)*0.250;
    for I1=1:N
      TAA=TAA+DA;
      TB=TB+DB;
      U=cos(TAA);
      F11=0.0;
      F2=0.0;
      F3=0.0;
      F44=0.0;
      F12=0.0;
      F34=0.0;
      P1=0.0;
      P2=0.0;
      P3=0.0;
      P4=0.0;
      PP1=1.0;
      PP2=0.250*(1.0+U)*(1.0+U);
      PP3=0.250*(1.0-U)*(1.0-U);
      PP4=D6*(U*U-1.0);
      for L1=1:L1MAX
        L=L1-1;
        DL=(L);
        DL1=(L1);
        F11=F11+A1(L1)*PP1;
        F44=F44+A4(L1)*PP1;
        if (L1~=LMAX)
        PL1=(2*L+1);
        P=(PL1*U*PP1-DL*P1)/DL1;
        P1=PP1;
        PP1=P;
        end
        if (L>=2)
        F2=F2+(A2(L1)+A3(L1))*PP2;
        F3=F3+(A2(L1)-A3(L1))*PP3;
        F12=F12+B1(L1)*PP4;
        F34=F34+B2(L1)*PP4;
        if (L~=LMAX)
```

```
      PL2=(L*L1)*U;
      PL3=(L1*(L*L-4));
      PL4=1D0/(L*(L1*L1-4));
      P=(PL1*(PL2-4D0)*PP2-PL3*P2)*PL4;
      P2=PP2;
      PP2=P;
      P=(PL1*(PL2+4D0)*PP3-PL3*P3)*PL4;
      P3=PP3;
      PP3=P;
      P=(PL1*U*PP4-sqrt((L*L-4))*P4)/sqrt((L1*L1-4));
      P4=PP4;
      PP4=P;
      end
      end
    end

    F22=(F2+F3)*0.50;
    F33=(F2-F3)*0.50;
    % Following lines were commented out and are now restored.
   F22=100D0*F22/F11;
   F33=100D0*F33/F11;
   F44=100D0*F44/F11;
   F12=-100D0*F12/F11;
   F34=100D0*F34/F11;
end
```

```
%C
%C*************************************************************
%C
%     SUBROUTINE MIE1(X,SN,SK,NSTOP,QEXT,QSCA,AN,CN,IRAY)
%     IMPLICIT REAL*8 (A-H,O-Z)
%     PARAMETER(NOD=60,NOMAX2=2*NOD)
%     COMPLEX*16 Y,RI,XIP,PCP,DA,DB,PC(0:NOMAX2),XI(0:NOMAX2),
%    1 AN(*),NA,NB
%     REAL*8 CN(*)
%C
% Calculates Mie scattering coefficients for a sphere, where:
% X is the wavenumber times the radius
% SN and SK are the real & imaginary parts of the refractive index of the sphere
% NSTOP is the numbr of terms
% IRAY = 0 in this code.
% NM is the refractive index of the medium
function [QEXT,QSCA,AN,CN] = Mie1(X, SN, SK, NSTOP, IRAY)
RI=SN + sqrt(-1)*SK;
Y=X*RI;
% Returns ns*x*j_(n)(ns*x)
PC = PSIC(NSTOP, Y);
% Scattered Hankel function on surface, nm*x*h_(n)(x*nm)
XI = BesselFunc(NSTOP, X);
QSCA=0.0;
QEXT=0.0;
for N=1:NSTOP
   NNN=N+1; % Avoids indexing issues.
   N1=N+N-1; % These are OK as they start from 1.
   N2=N+N;
   PRN=real(XI(NNN));
   % Derivative of ns*x*j_n(x*ns)
   PCP=PC(NNN-1)-N*PC(NNN)/Y;
   % Derivative of nm*x*h_n(x*nm)
   XIP=XI(NNN-1)-N*XI(NNN)/X;
   % Derivative of nm*x*j_n(x*nm), i.e. incoming.
   PRP=real(XIP);
   DA=RI*XIP*PC(NNN)-XI(NNN)*PCP;
   DB=RI*XI(NNN)*PCP-XIP*PC(NNN);
   NA=RI*PRP*PC(NNN)-PRN*PCP;
   NB=RI*PRN*PCP-PRP*PC(NNN);
   AN(N1)=NA/DA;
   AN(N2)=NB/DB;
   if (NA==0)
      DN1=0.;
   else
      DN1=1./NA/(NA');
   end
   if (NB==0.)
      CN1=0.;
   else
      CN1=1./NB/(NB');
   end
   NA=sqrt(-1)*PCP*(PC(N)');
   CN(N1)=NA*DN1*(RI');
   CN(N2)=NA*CN1*RI;
```

```
   %C       CN(N1)=DCMPLX(0.D0,1.D0)*RI/NA
   %C       CN(N2)=-DCMPLX(0.D0,1.D0)*RI/NB
   QSCA=QSCA+(N+N+1)*(abs(AN(N1))*abs(AN(N1))+abs(AN(N2))*abs(AN(N2)));
   QEXT=QEXT+(N+N+1)*real(AN(N1)+AN(N2));
end
QSCA=2./X/X*QSCA;
QEXT=2./X/X*QEXT;
if (NSTOP==1)&&(IRAY>0)
   AN(1)=-sqrt(-1)*2.*X*X*X*(RI-1.)*(RI+1.)/(RI*RI+2.)/3.;
end
if (IRAY==2)
   AN(1)=AN(1)/(1.+AN(1));
end

   function PSI = PSIC(N, DS)
      % This returns DS*j_n(DS)
      NS = floor(abs(DS)+4.0*(abs(DS)^0.33333)+17.0);
      PSINS=0;
      for II=NS-1:-1:N
         SN0=(II+1)/DS;
         PSINS=SN0-1.0/(PSINS+SN0);
      end
      PSI(N+1)=PSINS;
      PSI(N)=N/DS-1.0/(PSINS+N/DS);
      for II=N-1:-1:2
         SN0=(II)/DS;
         PSI(II)=SN0-1.0/(PSI(II+1)+SN0);
      end
      PSINS=sin(DS);
      PSI(1)=PSINS;
      for II=1:N
         PSINS=PSINS/(II/DS+PSI(II+1));
         PSI(II+1)=PSINS;
      end
   end
end
```

```
%     SUBROUTINE VWHAC2(ITYPE,ICALL,R,NTOT,LTOT,MS,MOLD,NM,LM,
%    1          CM,CN,CN1,NCD,C,A,B)
function [C, MOLD, CM, CN, CN1, NMIN] = VWHAC2_0(ITYPE,R,NTOT,LTOT,MS,MOLD, C,
CM, CN, CN1)
%
% THIS IS ICALL=0 but NOT MOLD = -1 (unused condition left in just in case)
%
%C
%C  COMPUTES SCALAR ADDITION COEFFICIENTS FOR AXIAL TRANSLATION
%C
%C  INPUT: ITYPE: =1: RETURNS J MATRIX
%C              =3: RETURNS H MATRIX
%C         ICALL: = 0, INITIALIZE AND/OR PERFORM RECURRENCE IN DEGREE
%C                   M TO CALCULATE SCALAR COEFFICIENTS (C)
%C              = 1, CALCULATE VECTOR COEFFICIENTS A,B FROM SCALAR C
%C                   FOR ORDER NM AND LM AND DEGREE M.
%C         R: AXIAL DISTANCE
%C         NTOT: ORDER N MAX
%C         LTOT: ORDER L MAX
%C         MS: DEGREE M OF COEFFICIENTS, MS>=0
%C         NCD: DIMENSION OF C ARRAY
%C  OUTPUT:A,B:  A,B VECTOR COEFFICIENTS: A^MN_ML ETC
%C         C: ARRAY OF SCALAR COEFFICIENTS
%C
%C  WRITTEN 5/5/94
%C
%C  USAGE:  START WITH MS=0, CALL VWHAC2 WITH ICALL=0: RETURNS
%C         SCALAR C ARRAY.  THEN USE ICALL=1 (WITH SAME M) TO CALC
%C         A & B VALUES FOR DIFFERENT NM AND LM'S.  NEXT, SET MS=1,
%C         ICALL=0, AND CALC C'S.  THEN SET ICALL=1, GET A,B'S, AND SO
%C         ON, UNTIL M=NTOT OR LTOT.
%C

   NMIN=min(NTOT,LTOT);
   NMAX=max(NTOT,LTOT);
   MSA=abs(MS);
   if(MSA<MOLD); MOLD=-1; end
   NT2=NTOT+LTOT+1;
   %C
   %C  IF M=0, INITIALIZE, IF M>0, USE RECURRENCE FOR PREVIOUS M
   %C
   for M=MOLD+1:MSA
      MM=M+1;
      if (M==0)
         XI = BesselFunc(NT2, R);
         for L=0:NT2
            LL=L+1;
            if (ITYPE==3)
               CM(LL)=((-1)^L)*(L+L+1)*XI(LL)/R;
            else
               if (ITYPE==1)
                  CM(LL)=((-1)^L)*(L+L+1)*real(XI(LL))/R;
               end
            end
            if(L<=NMAX+1); C(1,LL)=CM(LL); end
```

```
            CN(LL)=CM(LL);
            CN1(LL)=0.;
          end
        else
          %CT1=CM(M-1);
          CT1=CM(MM-1); % Because nothing starts from zero.
          for L=M:NT2-M
            LL=L+1;
            CT=CM(LL);
            CM(LL)=(M+M-1)*(CT1/real(L+L-1)+CM(LL+1)/real(L+L+3));
            CT1=CT;
            if(L<=NMAX+1); C(MM,LL)=CM(LL); end
            CN(LL)=CM(LL);
            CN1(LL)=0.;
          end
        end
        %C
        %C  CALCULATE FOR N=M+1
        %C
        CN1(MM)=CN(MM);
        CN(MM)=-(M+M+1)*(M+M+1)/real(M+M+3)*CN(MM+1);
        C(MM+1,MM)=CN(MM);

        for L=M+1:NT2-M-1
          LL=L+1;
          CN1(LL)=CN(LL);
          CN(LL)=(M+M+1)*((L-M)/(L+L-1)*CN1(LL-1)-(L+M+1)/(L+L+3)*CN(LL+1));
          if(L<=NMAX+1); C(MM+1,LL)=CN(LL); end
        end

        %C
        %C  RECURRENCE FOR N=M+2,M+3,... NTOT
        %C

        for N=M+1:NMIN
          NN=N+1;
          CT=CN1(MM);
          CN1(MM)=CN(MM);
          CN(MM)=((N+M)*CT-(N+N+1)*(M+M+1)/real(M+M+3)*CN(MM+1))/real(N-M+1);
          C(NN+1,MM)=CN(MM);
          for L=M+1:NT2-N-1
            LL=L+1;
            CT=CN1(LL);
            CN1(LL)=CN(LL);
            CN(LL)=((N+M)*CT+(N+N+1)*((L-M)/real(L+L-1)*CN1(LL-1)-
(L+M+1)/real(L+L+3)*CN(LL+1)))/real(N-M+1);
            if(L<=NMAX+1);  C(NN+1,LL)=CN(LL); end
          end
        end
      end
      %C
      MOLD=MSA;
```

```
%     SUBROUTINE VWHAC2(ITYPE,ICALL,R,NTOT,LTOT,MS,MOLD,NM,LM,
%    1          CM,CN,CN1,NCD,C,A,B)
function [C, MOLD, CM, CN, CN1, NMIN] = VWHAC2_00(ITYPE,R,NTOT,LTOT,MS,MOLD)
%
% THIS IS ICALL=0 AND MOLD=-1
%
%C
%C  COMPUTES SCALAR ADDITION COEFFICIENTS FOR AXIAL TRANSLATION
%C
%C  INPUT: ITYPE: =1: RETURNS J MATRIX
%C              =3: RETURNS H MATRIX
%C        ICALL: = 0, INITIALIZE AND/OR PERFORM RECURRENCE IN DEGREE
%C                  M TO CALCULATE SCALAR COEFFICIENTS (C)
%C              = 1, CALCULATE VECTOR COEFFICIENTS A,B FROM SCALAR C
%C                  FOR ORDER NM AND LM AND DEGREE M.
%C        R: AXIAL DISTANCE
%C        NTOT: ORDER N MAX
%C        LTOT: ORDER L MAX
%C        MS: DEGREE M OF COEFFICIENTS, MS>=0
%C        NCD: DIMENSION OF C ARRAY
%C  OUTPUT:A,B:  A,B VECTOR COEFFICIENTS: A^MN_ML ETC
%C        C: ARRAY OF SCALAR COEFFICIENTS
%C
%C  WRITTEN 5/5/94
%C
%C  USAGE:  START WITH MS=0, CALL VWHAC2 WITH ICALL=0: RETURNS
%C        SCALAR C ARRAY.  THEN USE ICALL=1 (WITH SAME M) TO CALC
%C        A & B VALUES FOR DIFFERENT NM AND LM'S.  NEXT, SET MS=1,
%C        ICALL=0, AND CALC C'S.  THEN SET ICALL=1, GET A,B'S, AND SO
%C        ON, UNTIL M=NTOT OR LTOT.
%C

   NMIN=min(NTOT,LTOT);
   NMAX=max(NTOT,LTOT);
   MSA=abs(MS);
   if(MSA<MOLD); MOLD=-1; end
   NT2=NTOT+LTOT+1;
   %C
   %C  IF M=0, INITIALIZE, IF M>0, USE RECURRENCE FOR PREVIOUS M
   %C
   for M=MOLD+1:MSA
      MM=M+1;
      if (M==0)
         XI = BesselFunc(NT2, R);
         for L=0:NT2
            LL=L+1;
            if (ITYPE==3)
               CM(LL)=((-1)^L)*(L+L+1)*XI(LL)/R;
            else
               if (ITYPE==1)
                  CM(LL)=((-1)^L)*(L+L+1)*real(XI(LL))/R;
               end
            end
            if(L<=NMAX+1); C(1,LL)=CM(LL); end
            CN(LL)=CM(LL);
```

```
            CN1(LL)=0.;
          end
        else
          %CT1=CM(M-1);
          CT1=CM(MM-1); % Because nothing starts from zero.
          for L=M:NT2-M
            LL=L+1;
            CT=CM(LL);
            CM(LL)=(M+M-1)*(CT1/real(L+L-1)+CM(LL+1)/real(L+L+3));
            CT1=CT;
            if(L<=NMAX+1); C(MM,LL)=CM(LL); end
            CN(LL)=CM(LL);
            CN1(LL)=0.;
          end
        end
        %C
        %C  CALCULATE FOR N=M+1
        %C
        CN1(MM)=CN(MM);
        CN(MM)=-(M+M+1)*(M+M+1)/real(M+M+3)*CN(MM+1);
        C(MM+1,MM)=CN(MM);

        for L=M+1:NT2-M-1
          LL=L+1;
          CN1(LL)=CN(LL);
          CN(LL)=(M+M+1)*((L-M)/(L+L-1)*CN1(LL-1)-(L+M+1)/(L+L+3)*CN(LL+1));
          if(L<=NMAX+1); C(MM+1,LL)=CN(LL); end
        end

        %C
        %C  RECURRENCE FOR N=M+2,M+3,... NTOT
        %C

        for N=M+1:NMIN
          NN=N+1;
          CT=CN1(MM);
          CN1(MM)=CN(MM);
          CN(MM)=((N+M)*CT-(N+N+1)*(M+M+1)/real(M+M+3)*CN(MM+1))/real(N-M+1);
          C(NN+1,MM)=CN(MM);
          for L=M+1:NT2-N-1
            LL=L+1;
            CT=CN1(LL);
            CN1(LL)=CN(LL);
            CN(LL)=((N+M)*CT+(N+N+1)*((L-M)/real(L+L-1)*CN1(LL-1)-
(L+M+1)/real(L+L+3)*CN(LL+1)))/real(N-M+1);
            if(L<=NMAX+1);  C(NN+1,LL)=CN(LL); end
          end
        end
      end
      %C
      MOLD=MSA;
```

```
%     SUBROUTINE VWHAC2(ITYPE,ICALL,R,NTOT,LTOT,MS,MOLD,NM,LM,
%    1          CM,CN,CN1,NCD,C,A,B)
function [A, B] = VWHAC2_1(R,MS,NM,LM,C, NMIN, FAC)
%
% THIS IS ICALL=1
%
%C
%C  COMPUTES SCALAR ADDITION COEFFICIENTS FOR AXIAL TRANSLATION
%C
%C  INPUT: ITYPE: =1: RETURNS J MATRIX
%C              =3: RETURNS H MATRIX
%C        ICALL: = 0, INITIALIZE AND/OR PERFORM RECURRENCE IN DEGREE
%C                  M TO CALCULATE SCALAR COEFFICIENTS (C)
%C               = 1, CALCULATE VECTOR COEFFICIENTS A,B FROM SCALAR C
%C                  FOR ORDER NM AND LM AND DEGREE M.
%C        R: AXIAL DISTANCE
%C        NTOT: ORDER N MAX
%C        LTOT: ORDER L MAX
%C        MS: DEGREE M OF COEFFICIENTS, MS>=0
%C        NCD: DIMENSION OF C ARRAY
%C  OUTPUT:A,B:  A,B VECTOR COEFFICIENTS: A^MN_ML ETC
%C        C: ARRAY OF SCALAR COEFFICIENTS
%C
%C  WRITTEN 5/5/94
%C
%C  USAGE:  START WITH MS=0, CALL VWHAC2 WITH ICALL=0: RETURNS
%C         SCALAR C ARRAY.  THEN USE ICALL=1 (WITH SAME M) TO CALC
%C         A & B VALUES FOR DIFFERENT NM AND LM'S.  NEXT, SET MS=1,
%C         ICALL=0, AND CALC C'S.  THEN SET ICALL=1, GET A,B'S, AND SO
%C         ON, UNTIL M=NTOT OR LTOT.
%C
%
% NB: All FAC terms have had +1 added to indices since our vector numbering
% starts from 1 not 0.
   M=abs(MS);
   if (MS>=0)
      if (NM<=NMIN)
         N=NM;
         L=LM;
         FTERM=1.;
      else
         L=NM;
         N=LM;
         FTERM=((-1)^(N+L))*(N+N+1)*L*(L+1)*FAC(N-
M+1)*FAC(L+M+1)/((L+L+1)*N*(N+1))/FAC(N+M+1)/FAC(L-M+1);
      end
   else
      if (NM<=NMIN)
         N=NM;
         L=LM;
         FTERM=FAC(N-M+1)/FAC(N+M+1)*FAC(L+M+1)/FAC(L-M+1);
      else
         L=NM;
         N=LM;
         FTERM=((-1)^(N+L))*(N+N+1)*L*(L+1)/((L+L+1)*N*(N+1));
```

```
      end
   end
   NN=N+1; LL=L+1;
   C1=(L+M+1)*C(NN,LL+1)/((L+1)*(L+L+3));
   if(M>L-1)
      C2=0.;
   else
      C2=(L-M)*C(NN,LL-1)/(L*(L+L-1));
   end
   A=(R*(C1+C2)+C(NN,LL))*FTERM;
   B=sqrt(-1)*FTERM*R*MS*C(NN,LL)/(L*(L+1));
end
```