COMMUNICATIONS

Pascal program to perform Mie calculations

Jaap R. Zijp
Jaap J. ten Bosch
State University of Groningen
Laboratory for Materia Technica
Faculty of Medicine
Antonius Deusinglaan 1
9713 AV Groningen, The Netherlands

Subject terms: scattering; Mie theory; phase functions.
Optical Engineering 32(7), 1691–1695 (July 1993).

1 Introduction

The scattering of light by small spherical particles was originally described by Mie,[1] but also extensively set out by van de Hulst.[2] The calculations that consider the scattering of light by such particles are mostly called Mie calculations. Spherical scatterers are frequently occurring in nature, e.g., in aerosols and suspensions of biological cells. Many investigators dealing with them perform Mie calculations to compare their measured data with calculated ones.

In Mie theory the incident radiation is separated into two components, one perpendicular to and one parallel with the scattering plane. The scattering plane is defined as the plane through the incident and the scattering directions. The irradiance of the scattered radiation depends on the scattering angle θ. We use the terms and symbols as prescribed by international conventions.[3] The functions $i_1(\theta)$ and $i_2(\theta)$ (Ref. 2, p. 35) describe the angular dependency of the perpendicular and parallel polarized components, respectively. The scattered irradiance of linear polarized incident light in any direction can be calculated by:

$$E(\theta,\phi)=\frac{E_0}{k^2r^2}[i_1(\theta)\ \sin^2\phi+i_2(\theta)\ \cos^2\phi]\ ,$$

where

E_0 = the irradiance of the incident radiation in watts per meter^2

E = the irradiance of the scattered radiation in watts per meter^2 at distance r in meters

k = the wave number in meters^{-1}

ϕ = the angle between the polarization direction and the scattering plane.

We present a computer program in standard ISO PASCAL[4] to perform Mie calculations. Several investigators used our program,[5,6] and others showed their interest, which is why we present its source code here.

2 Input and Output

The input parameters are the size parameter

$$x=2\pi n_{med}\ a/\lambda_{vac}$$

(n_{med} is the refractive index of the medium surrounding the particles, a is the radius of the particles, and λ_{vac} is the wavelength in vacuum) and the relative refractive index

$$m=n_{part}/n_{med}$$

(n_{part} is the refractive index of the particle). When $x=0$ is given as input, the program asks for the diameter of the particle and λ_{vac} (both in nanometers) and n_{med}, and calculates x for these data. When $m=0$ is given as input, the program asks for n_{part} and n_{med}.

The output of the program is: first, the functions $i_1(\theta)$ and $i_2(\theta)$, which are stored in the arrays $i1[i]$ and $i2[i]$, $0 \leqslant i \leqslant 720$, i corresponding to θ so that a resolution of 0.25 deg is obtained; second, the efficiency factor for scattering Q_{sca}, which is stored in the global variable *qsca*; and third, the anisotropy factor g (stored in the global variable g), which is the mean cosine of the scattering angle θ for naturally polarized incident light.

We followed the treatment by van de Hulst (Ref. 2, 114 to 130), and the special functions contained therein are presented by Abramowitz and Stegun in Ref. 7. Comments in the program text designated as HLST and HBMF refer to these books respectively.

Paper CM1023 received Feb. 3, 1993; accepted for publication Feb. 5, 1993.

3 Program Listing

```
program ztbmie;
type complex=record r,i:real; end;

const nmax=200; nmaxplusone=201;
   minreal=1.0E-38; maxreal=0.99E+38; precision=1.0E-06;

var i,n,maxn:integer;
   x,m,g,Qsca,theta:real;
   pie,tau,psi,psiprime,psiy,psiprimey:array [1..nmax] of real;
   zeta,zetaprime,zetay,zetaprimey:array [1..nmaxplusone] of complex;
   a,b:array [0..nmax] of complex;
   i1,i2:array [0..720] of real;

(************************************************************************)
(*                  ARITHMATIC FUNCTIONS AND PROCEDURES                 *)
(************************************************************************)

function pwrrs(x,y:real):real; (* x raised to the power y *)
   var scrint:integer;         (* defined for x>0 or (x<0 and y=integer) *)

   function is_int(y:real):boolean;
      begin is_int:=(y-trunc(y)=0.0); end;

   begin
   if ( (x<0)and(not is_int(y)) )
     then begin pwrrs:=0.0; writeln('error in function pwrrs'); end;
   if x=0 then begin pwrrs:=0.0; end;
   if ( (x<0) and (is_int(y)) )
      then begin
      if odd(round(y)) then pwrrs:=-pwrrs(-x,y)
                       else pwrrs:= pwrrs(-x,y);
      end;
   if x>0 then pwrrs:=exp(y*ln(x));
   end; (* pwrrs *)

function cabs(z:complex):real;               (* cabs:=|z| *)
   begin  cabs:=sqrt(sqr(z.r)+sqr(z.i)); end;

function creal(z:complex):real;               (* creal:=z.r *)
   begin  creal:=z.r; end;

procedure cadd(z1,z2:complex;var sum:complex); (* sum:=z1+z2 *)
   begin sum.r:=z1.r+z2.r; sum.i:=z1.i+z2.i; end;

procedure cmult(z1,z2:complex;var prod:complex); (* prod:=z1*z2 *)
   begin prod.r:=z1.r*z2.r-z1.i*z2.i;
         prod.i:=z1.r*z2.i+z1.i*z2.r; end;

procedure cdiv(z1,z2:complex;var ratio:complex); (* ratio:=z1/z2 *)
   begin
   if ((z2.r=0.0) and (z2.i=0.0))
      then begin
      ratio.r:=maxreal; ratio.i:=maxreal;
      write('error in cdiv');
      end
      else begin
      ratio.r:=(z1.r*z2.r+z1.i*z2.i)/(z2.r*z2.r+z2.i*z2.i);
      ratio.i:=(z1.i*z2.r-z1.r*z2.i)/(z2.r*z2.r+z2.i*z2.i);
      end;
   end; (* cdiv *)

(************************************************************************)
(*                        AUXILIARY PROCEDURES                          *)
(************************************************************************)

procedure calc_psi_zeta(x:real; maxn:integer);
   var n:integer;
   begin
   (* psi[n](x) = x*j[n](x)  HLST p123 , HBMF 10.1.11 *)
   psi[1]:=sin(x)/x-cos(x);
   psi[2]:=(3*pwrrs(x,-2)-1)*sin(x)-3*cos(x)/x;
   if maxn>2 then for n:=3 to maxn do
      psi[n]:=(2*n-1)*psi[n-1]/x-psi[n-2];              (* HBMF 10.1.19 *)

   (* psi'[n](x) *)
   psiprime[1]:=(1-pwrrs(x,-2))*sin(x)+cos(x)/x;        (* HBMF 10.1.21 *)
   if maxn>1 then for n:=2 to maxn do
         psiprime[n]:=psi[n-1]-n*psi[n]/x;
```

```
(* zeta[n](x)=x*h(2)[n](x) complex HLST p123 *)
zeta[1].r:=sin(x)/x-cos(x);                              (* HBMF 10.1.17 *)
zeta[1].i:=sin(x)+cos(x)/x;
zeta[2].r:=(-1+3*pwrrs(x,-2))*sin(x) - 3/x*cos(x);
zeta[2].i:=3/x*sin(x) + (-1+3*pwrrs(x,-2))*cos(x);
for n:=3 to maxn+1 do
   begin
   zeta[n].r:=(2*n-1)*zeta[n-1].r/x - zeta[n-2].r;
   zeta[n].i:=(2*n-1)*zeta[n-1].i/x - zeta[n-2].i;
   end;

(* zeta'[n](x) complex *)
for n:=1 to maxn do
   begin
   zetaprime[n].r:=(n+1)*zeta[n].r/x - zeta[n+1].r;
   zetaprime[n].i:=(n+1)*zeta[n].i/x - zeta[n+1].i;
   end;
end; (* calc_psi_zeta *)

procedure mie_var(n:integer);
   (* calculates psi(x),psi'(x),zeta(x),zeta'(x) *)
   begin  (* and psi(y),psi'(y),zeta(y),zeta'(y) *)
   calc_psi_zeta(x*m,n);
   psiy[n]      :=psi[n];
   psiprimey[n] :=psiprime[n];
   zetay[n]     :=zeta[n];
   zetaprimey[n]:=zetaprime[n];
   calc_psi_zeta(x,n);
   end; (* mie_var *)

procedure calc_pie_tau(theta:real);
   var n:integer;
   begin
   (* pie[n](cos(theta))=dp[n]/dcos(theta) HLST p124 and HBMF p342 *)
   pie[1]:=-1;
   pie[2]:=-3*cos(theta);
   if maxn>2 then for n:=3 to maxn do
      pie[n]:=(((2*n-1)*cos(theta)*pie[n-1]-n*pie[n-2]))/(n-1);

   (* tau[n]=dP1[n]/d(theta)=dP1[n]/dcos(theta)*sin(theta) HLST p124 *)
   tau[1]:=-cos(theta);
   if maxn>1 then for n:=2 to maxn do
      tau[n]:=(n*cos(theta)*pie[n]-(n+1)*pie[n-1]);     (* HBMF 8.5.4 *)
   end; (* calc_pie_tau *)

(*************************************************************************)
(*                        PROCEDURES FOR MAIN PROGRAM                    *)
(*************************************************************************)

procedure nulling;
   begin
   for i:= 0 to 720 do begin i1[i]:=0.0; i2[i]:=0.0; end;
   for i:=1 to nmax do
      begin
      psiy[i]:=0.0;      psi[i]:=0.0;
      psiprimey[i]:=0.0; psiprime[i]:=0.0;
      end;
   for i:=0 to nmaxplusone do
      begin
      zetay[i].r:=0.0;       zeta[i].r:=0.0;
      zetay[i].i:=0.0;       zeta[i].i:=0.0;
      zetaprimey[i].r:=0.0;  zetaprime[i].r:=0.0;
      zetaprimey[i].i:=0.0;  zetaprime[i].i:=0.0;
      end;
   end;

procedure read_x_m;
   var d,lambda,n1,n2:real;
   begin
   n1:=0.0;
   write('x = ') ;readln(x);
   if x=0
      then begin
      write('lambda (nm) = '); readln(lambda);
      write('d      (nm) = '); readln(d);
      write('rfer.index surrounding = '); readln(n1);
      x:=2*pi*n1*d/(2*lambda);
      end;
   write('m = '); readln(m);
   if m=0
      then begin
```

```
      write('rfer.index    particle = '); readln(n2);
      if n1=0.0
         then begin
         write('rfer.index surrounding = '); readln(n1);
         end;
      m:=n2/n1;
      end;
   end; (* read_x_m *)

procedure mie_an_bn;    (* calculates a[n],b[n] HLST p123 *)
   var maxabsa,maxabsb:real; (* maxima of a[n] and b[n]   *)
     num,den:complex;        (* numerator and denominator *)

   procedure calc_an;
      begin
      num.r:=psiprimey[n]*psi[n]-m*psiy[n]*psiprime[n];
      num.i:=0;
      den.r:=psiprimey[n]*zeta[n].r-m*psiy[n]*zetaprime[n].r;
      den.i:=psiprimey[n]*zeta[n].i-m*psiy[n]*zetaprime[n].i;
      cdiv(num,den,a[n]);
      end;

   procedure calc_bn;
      begin
      num.r:=m*psiprimey[n]*psi[n]-psiy[n]*psiprime[n];
      num.i:=0;
      den.r:=m*psiprimey[n]*zeta[n].r-psiy[n]*zetaprime[n].r;
      den.i:=m*psiprimey[n]*zeta[n].i-psiy[n]*zetaprime[n].i;
      cdiv(num,den,b[n]);
      end;

      begin
      maxabsa:=minreal; maxabsb:=minreal;
      n:=0;
      repeat
         inc(n);
         mie_var(n);
         calc_an;
         if cabs(a[n])>maxabsa then maxabsa:=cabs(a[n]);
         calc_bn;
         if cabs(b[n])>maxabsb then maxabsb:=cabs(b[n])
         until (n>1)and(cabs(a[n])/maxabsa<precision) and
                       (cabs(b[n])/maxabsb<precision);
      maxn:=n;
      inc(n);
      mie_var(n);
      calc_an;  (* a[n+1] and b[n+1] needed to calculate g *)
      calc_bn;
      end; (* mie_an_bn *)

   procedure mie_sigma(theta:real);
      (* calculates i1 en i2  HLST p35 by HLST p125 *)
      var s1,s2:complex;
      begin
      calc_pie_tau(theta);
      s1.r:=0.0  ;s1.i:=0.0;   s2.r:=0.0  ;s2.i:=0.0;
      for n:=1 to maxn do
         begin
         s1.r:=s1.r + (2*n+1) * (a[n].r*pie[n]+b[n].r*tau[n]) / (n*(n+1));
         s1.i:=s1.i + (2*n+1) * (a[n].i*pie[n]+b[n].i*tau[n]) / (n*(n+1));

         s2.r:=s2.r + (2*n+1) * (b[n].r*pie[n]+a[n].r*tau[n]) / (n*(n+1));
         s2.i:=s2.i + (2*n+1) * (b[n].i*pie[n]+a[n].i*tau[n]) / (n*(n+1));
         end;
      i1[i]:=sqr(s1.r) + sqr(s1.i);
      i2[i]:=sqr(s2.r) + sqr(s2.i);
      end; (* mie_sigma *)

   procedure calc_qsca; (* HLST p128 *)
      begin
      qsca:=0.0;
      for n:=maxn downto 1 do
         qsca:=qsca+2*(2*n+1)*(sqr(cabs(a[n]))+sqr(cabs(b[n])))/sqr(x);
      end;

   procedure calc_g; (* HLST p128 *)
      var scr1,scr2,scr3:complex;
      begin
      g:=0.0;
      for n:=maxn downto 1 do
```

```
    begin
    scr1.r:=a[n+1].r; scr1.i:=-a[n+1].i; cmult(a[n],scr1,scr1);
    scr2.r:=b[n+1].r; scr2.i:=-b[n+1].i; cmult(b[n],scr2,scr2);
    scr3.r:=b[n].r;   scr3.i:=-b[n].i;   cmult(a[n],scr3,scr3);
    cadd(scr1,scr2,scr1);
    g:=g+n*(n+2)*creal(scr1)/(n+1) + (2*n+1)*creal(scr3)/(n*(n+1));
    end;
g:=4*g/(sqr(x)*qsca);
end; (* calc_g *)

BEGIN (* main *)
nulling;
read_x_m;
mie_an_bn;
for i:=0 to 720 do mie_sigma(i*pi/720);
calc_qsca;
calc_g;
(* at this stage i1[0..720] and i2[0..720] contain the phase function *)
(* qsca is the relative scattering crosssection, g the anisotropy factor *)

END.  (* main *)
```

4 Performance

The program was tested with a Turbo Pascal compiler on a PC by Graaff et al.[5] By comparison with other data, Graaff et al. showed that the values of $i1[i]$, $i2[i]$, Q_{sca}, and g are accurate within 0.0003%. The running time depends on the values x and m. On a 33-MHz 486 PC we measured 0.27 s ($x = 0.10$, $m = 1.025$) and 9.8 s ($x = 50.0$, $m = 1.50$). Decreasing the accuracy of the output by setting the constant *precision* to a larger value does not substantially increase the speed of the program.

5 Conclusion

The computer program presented is easy and accurate, and may be valuable for many scientists.

Acknowledgment

The authors would like to thank R. Graaff for helping to bring the presented program to its final form, and for performing the comparisons with other calculations.

References

1. G. Mie, "Beiträge zur optik trüber medien, speziell kolloidaler metallösingen," *Annalen der Physik* **25**, 377–445 (1908).
2. H. C. van de Hulst, *Light Scattering by Small Particles,* Dover Publications Inc., New York (1981).
3. Commission Internationale de 'l Éclairage, *Vocabulaire International de 'l Éclairage,* CIE Publ. No. 17.4 (1987).
4. K. Jensen and N. Wirth, *PASCAL User Manual and Report, ISO PASCAL Standard,* 3rd ed., Springer Verlag, New York (1985).
5. R. Graaff, J. G. Aarnoudse, J. R. Zijp, P. M. A. Sloot, F. F. M. de Mul, J. Greve, and M. H. Koelink, "Reduced light-scattering properties for mixtures of spherical particles: a simple approximation derived from Mie calculations," *Appl. Opt.* **31**, 1370–1376 (1992).
6. H. J. van Staveren, C. J. M. Moes, J. van Marle, S. A. Prahl, and M. J. C. van Gemert, "Light scattering in Intralipid-10% in the wavelength range of 400–1100 nm," *Appl. Opt.* **30**, 4507–4514 (1991).
7. M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions,* Dover Publications Inc., New York (1965).