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

1 Introduction

The scattering of light by small spherical particles was originally described by Mie, but also extensively set out by van de Hulst. 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. The functions i₁(θ) and i₂(θ) (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:

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

where
\[
\begin{align*}
E_0 & = \text{the irradiance of the incident radiation in watts per meter}^2 \\
E & = \text{the irradiance of the scattered radiation in watts per meter}^2 \text{ at distance } r \text{ in meters} \\
k & = \text{the wave number in meters}^{-1} \\
ϕ & = \text{the angle between the polarization direction and the scattering plane.}
\end{align*}
\]

We present a computer program in standard ISO PASCAL to perform Mie calculations. Several investigators used our program, 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

\[x = 2\pi n_{med} \frac{a}{\lambda_{vac}}\]

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

\[m = n_{part} / n_{med}\]

\((n_{part} \text{ is the refractive index of the particle}).\) When \(x = 0\) is given as input, the program asks for the diameter of the particle and \(\lambda_{vac}\) (both in nanometers) 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 \(i_1(θ)\) and \(i_2(θ)\), which are stored in the arrays \(i1[i]\) and \(i2[i]\), \(0 ≤ i ≤ 720\), \(i\) corresponding to \(θ\) so that a resolution of 0.25 deg is obtained; second, the efficiency factor for scattering \(Q_{scat}\), 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.
program ztbmie;
type complex=record r,i:real; end;
constant nmax=200; nmaxplusone=201;
minreal=1.0E-38; maxreal=0.99E+38; precision=1.0E-06;

var i,n, maxn: integer;
x,m,g,Qca,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;
il,i2: array [0..720] of real;

(* ARITHMETIC FUNCTIONS AND PROCEDURES *)

function pwrre(x,y: real): real; (* x raised to the power y *)
var ecrint: integer;
begin
  is_int := (y = trunc(y) = 0.0);
  if (x < 0) and not is_int(y) then begin
    if y = 0 then begin pwrre := 0.0; end;
    pwrre := pwrre(-x, y);
  end;
  if x = 0 then pwrre := 0.0;
  if x > 0 then begin
    if (x < 0) and is_int(y) then begin
      if odd(round(y)) then
        pwrre := pwrre(-x, y)
      else
        pwrre := pwrre(-x, y);
    end;
    pwrre := exp(y * ln(x));
  end;
end;

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

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

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

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

procedure cdiv(zl, z2: complex; var ratio: complex);
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;

(* AUXILIARY PROCEDURES *)

procedure calc_psi_zeta(x: real; maxn: integer);
var n: integer;
begin
  (* psi[n](x) = x^n/sin(x) * HLM 123, HBMF 10.1.11 *)
  psi[1] := sin(x) / x;
  psi[2] := (3 * pwrre(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] := (1 - pwrre(x, -2)) * sin(x) + cos(x) / x;
  if maxn > 1 then for n := 2 to maxn do
    psiprime[n] := psi'[n - 1] - psi'[n - 2]; (* HBMF 10.1.21 *)
PASCAL PROGRAM TO PERFORM MIE CALCULATIONS

(* zeta(n)(x)=x*h(2)(n)(x) complex HLST p123 *)

zeta[1].r:=sin(x)/x-cos(x);
zeta[1].i:=sin(x)*cos(x)/x;
zeta[2].r:=(-1+3*pwr(x,-2))*sin(x) - 3/x*cos(x);
zeta[2].i:=3/x*sin(x) + (-1+3*pwr(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_zeta *)

procedure mie_var(n:integer);
(* calculates psi(x),psi'(x),zeta(x),zeta'(x)*)
begin
  calc_psizeta(x,n);
  zetay[n].r:=psi[n].r;
  zetay[n].i:=psi[n].i;
  zetaprimey[n].r:=zetaprime[n].r;
  zetaprimey[n].i:=zetaprime[n].i;
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]:=(1/(2*n-1)*cos(theta)*pie[n-1]-n*pie[n-2])/(n-1);
  (* tau(n)=-dPl[n]/d(theta)=dPl(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-1]-n*pie[n-2])/(n-1);
end; (* calc_pie_tau *)

procedure nulling;
begin
  for i:=0 to 720 do begin il[i]:=0.0; i2[i]:=0.0; end;
  for i:=1 to mmax do
    begin
      psiy[i] :=psi[i];
      psiprimey[i] :=psiprime[i];
      zetay[i] :=zeta[i];
      zetaprimey[i] :=zetaprime[i];
    end;
  for i:=0 to mmaxplusone 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,nl,n2:real;
begin
  nl:=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(nl);
      x:=2*pi*nl*d/(2*lambda);
    end;
  write('m = '); readln(m);
  if m=0
    then begin

end;
ZJP and TEN BOSCH

write('rfer.index particle = ''); readln(n2);
if n1=0.0
then begin
write('rfer.index surrounding = ''); readln(n1);
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:=psi[n]*psi[n]*psiy[n]*psiprime[n];
num.i:=0;
den.r:=psiprime[n]*psiy[n]*zetaprime[n];
den.i:=psi[n]*zetaprime[n];
cdiv(num,den,a[n]);
end;

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

begin
maxabsa:=minreal; maxabsb:=minreal;
maxn:=n;
maxn:=n;
end; (* mie_an_bn *)

procedure mie_sigma(theta:real);
(* calculates i1 en i2 HLST p35 by HLST p125 *)
var sl,s2:complex;
begin
for n:=maxn downto 1 do
begin
sl.r:=sqr(sl.r) + sqrt(sl.i); i2[i]:=sqrt(s2.r) + sqrt(s2.i);
end;
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 *)
begin
sqr(g):=0.0;
for n:=maxn downto 1 do
sqr(g):=sqr(g)+2*(2*n+1)*sqr(cabs(a[n])) / sqr(X);
end;
PASCAL PROGRAM TO PERFORM MIE CALCULATIONS

begin
sr1.r:=a[n+1].r; sr1.i:=-a[n+1].i; cmult(a[n],sr1,sr1);
sr2.r:=b[n+1].r; sr2.i:=-b[n+1].i; cmult(b[n],sr2,sr2);
sr3.r:=b[n].r; sr3.i:=-b[n].i; cmult(a[n],sr3,sr3);
cadd(sr1,sr2,sr1);
g:=g+n*(n+2)*creal(sr1)/(n+1) + (2*n+1)*creal(sr3)/(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 i[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. By comparison with other data, Graaff et al. showed that the values of $i1[i]$, $i2[i]$, $Q_{scat}$, 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