Mie Scattering and Absorption (Version 1)

Christian Mätzler

Institute of Applied Physics
University of Bern, Switzerland
matzler@iap.unibe.ch

June 2002

Editor: Stewart Nash

Contents

Abstract

A set of Mie functions has been developed in Python to compute the four Mie coefficients a_n, b_n, c_n and d_n, efficiencies of extinction, scattering, backscattering and absorption, the asymmetry parameter, and the two angular scattering functions S_1 and S_2. In addition to the scattered field, also the absolute-square of the internal field is computed and used to get the absorption efficiency in a way independent from the scattered field. This allows to test the computational accuracy. This first version of MATLAB Mie Functions is limited to homogeneous dielectric spheres without change in the magnetic permeability between the inside and outside of the particle. Required input parameters are the complex refractive index, $m = m' + im''$, of the sphere (relative to the ambient medium) and the size parameter, $x = ka$, where a is the sphere radius and k the wave number in the ambient medium.

1 Introduction

This report is a description of Mie-Scattering and Mie-Absorption programs written in the numeric computation and visualisation software, MATLAB (Math Works, 1992), for the improvement of radiative-transfer codes, especially to account for rain and hail in the microwave range and for aerosols and clouds in the submillimeter, infrared and visible range. Excellent descriptions of Mie Scattering were given by van de Hulst (1957) and by Bohren and Huffman (1983). The present programs are related to the formalism of

Bohren and Huffman (1983). In addition an extension (Section 2.5) is given to describe the radial dependence of the internal electric field of the scattering sphere and the absorption resulting from this field. Except for Section 2.5, equation numbers refer to those in Bohren and Huffman (1983), in short BH, or in case of missing equation numbers, page numbers are given. For a description of computational problems in the Mie calculations, see the notes on p. 126-129 and in Appendix A of BH.

2 Formulas for a homogeneous sphere

2.1 Mie coefficients and Bessel functions

Python function: mie_abcd
The key parameters for Mie calculations are the Mie coefficients a_n and b_n to compute the amplitudes of the scattered field, and c_n and d_n for the internal field, respectively. The computation of these parameters has been the most challenging part in Mie computations due to the involvement of spherical Bessel functions up to high order. With MATLAB's built-in double-precision Bessel functions, the computation of the Mie coefficients has so far worked well up to size parameters exceeding 10,000; the coefficients are given in BH on p.100:

$$a_n = \frac{m^2 j_n(mx)[xj_n(x)]' - \mu_1 j_n(x)[mxj_n(mx)]'}{m^2 j_n(mx)[xh_n^{(1)}(x)]' - \mu_1 h_n^{(1)}(x)[mxj_n(mx)]'} \tag{1}$$

$$b_n = \frac{\mu_1 j_n(mx)[xj_n(x)]' - j_n(x)[mxj_n(mx)]'}{\mu_1 j_n(mx)[xh_n^{(1)}(x)]' - h_n^{(1)}(x)[mxj_n(mx)]'} \tag{2}$$

$$c_n = \frac{\mu_1 j_n(x)[xh_n^{(1)}(x)]' - \mu_1 h_n^{(1)}(x)[xj_n(x)]'}{\mu_1 j_n(mx)[xhh_n^{(1)}(x)]' - h_n^{(1)}(x)[mxj_n(mx)]'} \tag{3}$$

$$d_n = \frac{\mu_1 m j_n(x)[xh_n^{(1)}(x)]' - \mu_1 m h_n^{(1)}(x)[xj_n(x)]'}{m j_n(mx)[xh_n^{(1)}(x)]' - \mu_1 h_n^{(1)}(x)[mxj_n(mx)]'} \tag{4}$$

where m is the refractive index of the sphere relative to the ambient medium, $x = ka$ is the size parameter, a the radius of the sphere and $k = 2\pi/\lambda$ is the wave number and λ the wavelength in the ambient medium. In deviation from BH, μ_1 is the ratio of the magnetic permeability of the sphere to the magnetic permeability of the ambient medium (corresponding to μ_1/μ in

BH). The functions $j_n(z)$ and $h_n^{(1)}(z) = j_n(z) + iy_n(z)$ are spherical Bessel functions of order n ($n = 1, 2, \ldots$) and of the given arguments, $z = x$ or mx, respectively, and primes mean derivatives with respect to the argument. The derivatives follow from the spherical Bessel functions themselves, namely

$$\begin{aligned} [zj_n(z)]' &= zj_{n-1}(z) - nj_n(z) \\ [zh_n^{(1)}(z)]' &= zh_{n-1}^{(1)}(z) - nh_n^{(1)}(z) \end{aligned} \tag{5}$$

For completeness, the following relationships between Bessel and spherical Bessel functions are given:

$$j_n(z) = \sqrt{\frac{\pi}{2z}} J_{n+0.5}(z) \tag{6}$$

$$y_n(z) = \sqrt{\frac{\pi}{2z}} Y_{n+0.5}(z) \tag{7}$$

Here, J_ν and Y_ν are Bessel functions of the first and second kind. For $n = 0$ and 1 the spherical Bessel functions are given (BH, p. 87) by

$$\begin{aligned} j_0(z) &= \sin z/z \\ j_1(z) &= \sin z/z^2 - \cos z/z \\ y_0(z) &= -\cos z/z \\ y_1(z) &= -\cos z/z^2 - \sin z/z \end{aligned} \tag{8}$$

and the recurrence formula

$$f_{n-1}(z) + f_{n+1}(z) = \frac{2n+1}{z} f_n(z) \tag{9}$$

where f_n is any of the functions j_n and y_n. Taylor-series expansions for small arguments of j_n and y_n are given on p. 130 of BH. The spherical Hankel functions are linear combinations of j_n and y_n. Here, the first type is required

$$h_n^{(1)}(z) = j_n(z) + iy_n(z) \tag{10}$$

The following related functions are also used in Mie theory (although we try to avoid them here):

$$\begin{aligned} \psi_n(z) &= zj_n(z) \\ \chi_n(z) &= -zy_n(z) \\ \xi_n(z) &= zh_n^{(1)}(z) \end{aligned} \tag{11}$$

Often $\mu_1 = 1$; then, (4) simplify to

$$a_n = \frac{m^2 j_n(mx)[x j_n(x)]' - j_n(x)[mx j_n(mx)]'}{m^2 j_n(mx)[x h_n^{(1)}(x)]' - h_n^{(1)}(x)[mx j_n(mx)]'} \tag{12}$$

$$b_n = \frac{j_n(mx)[x j_n(x)]' - j_n(x)[mx j_n(mx)]'}{j_n(mx)[x h_n^{(1)}(x)]' - h_n^{(1)}(x)[mx j_n(mx)]'} \tag{13}$$

$$c_n = \frac{j_n(x)[x h_n^{(1)}(x)]' - h_n^{(1)}(x)[x j_n(x)]'}{j_n(mx)[x h_n^{(1)}(x)]' - h_n^{(1)}(x)[mx j_n(mx)]'} \tag{14}$$

$$d_n = \frac{m j_n(x)[x h_n^{(1)}(x)]' - m h_n^{(1)}(x)[x j_n(x)]'}{m^2 j_n(mx)[x h_n^{(1)}(x)]' - h_n^{(1)}(x)[mx j_n(mx)]'} \tag{15}$$

The parameters used in radiative transfer depend on a_n and b_n, but not on c_n and d_n. The latter coefficients are needed when the electric field inside the sphere is of interest, e.g. to test the field penetration in the sphere, to study the distribution of heat sources or to compute absorption. The absorption efficiency $Q_{\rm abs}$, however, can also be computed from the scattered radiation, Equation 18 and Equation 20 to be shown below.

2.2 Mie efficiencies and cross sections

Python functions:mie, mie_xscan The efficiencies Q_i for the interaction of radiation with a scattering sphere of radius a are cross sections σ_i (called C_i in BH) normalised to the particle cross section, πa^2, where i stands for extinction ($i = \text{ext}$), absorption ($i = \text{abs}$), scattering ($i = \text{sca}$), backscattering ($i = \text{b}$), and radiation pressure ($i = \text{pr}$), thus

$$Q_i = \frac{\sigma_i}{\pi a^2} \tag{16}$$

Energy conservation requires that

$$Q_{\rm ext} = Q_{\rm sca} + Q_{\rm abs} \tag{17}$$

$$\sigma_{\rm ext} = \sigma_{\rm sca} + \sigma_{\rm abs} \tag{18}$$

The scattering efficiency $Q_{\rm sca}$ follows from the integration of the scattered power over all directions, and the extinction efficiency $Q_{\rm ext}$ follows from the Extinction Theorem (Ishimaru, 1978, p. 14, van de Hulst, 1957, p. 31), also

called Forward-Scattering Theorem, leading to:

$$Q_{\rm sca} = \frac{2}{x^2}\sum_{n=1}^{\infty}(2n+1)(|a_n|^2+|b_n|^2) \tag{19}$$

$$Q_{\rm ext} = \frac{2}{x^2}\sum_{n=1}^{\infty}(2n+1){\rm Re}(a_n+b_n) \tag{20}$$

and $Q_{\rm abs}$ follows from (18). All infinite series can be truncated after $n_{\rm max}$ terms. For this number Bohren and Huffman (1983) proposed the value

$$n_{\rm max} = x + 4x^{1/3} + 2 \tag{21}$$

and this value is used here as well. Furthermore, the asymmetry parameter $g = \langle\cos\theta\rangle$ indicates the average cosine of the scattering angle θ with respect to power; it is used in Two-Stream Models (Meador and Weaver, 1980), and it is related to the efficiency $Q_{\rm pr}$ of radiation pressure:

$$Q_{\rm pr} = Q_{\rm ext} - Q_{\rm sca}\langle\cos\theta\rangle \tag{22}$$

$$Q_{\rm sca}\langle\cos\theta\rangle = \frac{4}{x^2}\left\{\sum_{n=1}^{\infty}\frac{n(n+2)}{n+1}{\rm Re}(a_n a_{n+1}^* + b_n b_{n+1}^*) + \sum_{n=1}^{\infty}\frac{2n+1}{n(n+1)}{\rm Re}(a_n b_n^*)\right\} \tag{23}$$

Finally, the backscattering efficiency Q_b, applicable to monostatic radar, is given by

$$Q_{\rm b} = \frac{1}{x^2}\left|\sum_{n=1}^{\infty}(2n+1)(-1)^n(a_n-b_n)\right|^2 \tag{24}$$

2.3 The scattered far field

Python functions: mie_s12, mie_pt, mie_thetascan If the detailed shape of the angular scattering pattern is required, e.g. to get the phase matrix or phase function for radiative-transfer calculations (Chandrasekhar, 1960), the scattering functions S_1 and S_2 are required. These functions describe the scattered field $\mathbf{E}_{\rm s}$. The scattered far field in spherical coordinates ($E_{{\rm s}\theta}$, $E_{{\rm s}\phi}$) for a unit-amplitude incident field (where the time variation $\exp(-i\omega t)$ has

been omitted) is given by

$$E_{s\theta} = \frac{e^{ikr}}{-ikr} \cos\phi \cdot S_2(\cos\theta) \tag{25}$$

$$E_{s\phi} = \frac{e^{ikr}}{ikr} \sin\phi \cdot S_1(\cos\theta) \tag{26}$$

with the scattering amplitudes S_1 and S_2

$$S_1(\cos\theta) = \sum_{n=1}^{\infty} \frac{2n+1}{n(n+1)} (a_n \pi_n + b_n \tau_n) \tag{27}$$

$$S_2(\cos\theta) = \sum_{n=1}^{\infty} \frac{2n+1}{n(n+1)} (a_n \tau_n + b_n \pi_n) \tag{28}$$

$E_{s\theta}$ is the scattered far-field component in the scattering plane, defined by the incident and scattered directions, and $E_{s\phi}$ is the orthogonal component. The angle ϕ is the angle between the incident electric field and the scattering plane. The functions $\pi_n(\cos\theta)$ and $\tau_n(\cos\theta)$ describe the angular scattering patterns of the spherical harmonics used to describe S_1 and S_2 and follow from the recurrence relations

$$\begin{aligned} \pi_n &= \frac{2n-1}{n-1} \cos\theta \cdot \pi_{n-1} - \frac{n}{n-1} \pi_{n-2} \\ \tau_n &= n \cos\theta \cdot \pi_n - (n+1) \pi_{n-1} \end{aligned} \tag{29}$$

starting with (Deirmendjian, 1969, p. 15)

$$\begin{aligned} \pi_0 &= 0 \\ \pi_1 &= 1 \\ \pi_2 &= 3\cos\theta \\ \tau_0 &= 0 \\ \tau_1 &= \cos\theta \\ \tau_2 &= 3\cos 2\theta \end{aligned} \tag{30}$$

2.4 The internal field

Python function: NONE (see mie_esquare) The internal field $\mathbf{E}_1$ for an incident field with unit amplitude is given by

$$\mathbf{E}_1 = \sum_{n=1}^{\infty} \frac{2n+1}{n(n+1)} \left(c_n \mathbf{M}_{o1n}^{(1)} - d_n \mathbf{N}_{e1n}^{(1)} \right) \tag{31}$$

where the vector-wave harmonic fields are given in spherical (r, θ, ϕ) coordinates by

$$\mathbf{M}^{(1)}_{o1n} = \begin{pmatrix} 0 \\ \cos\phi \cdot \pi_n(\cos\theta) j_n(rmx) \\ -\sin\phi \cdot \tau_n(\cos\theta) j_n(rmx) \end{pmatrix} \tag{32}$$

$$\mathbf{N}^{(1)}_{e1n} = \begin{pmatrix} n(n+1)\cos\phi \cdot \sin\theta \cdot \pi_n(\cos\theta) \frac{j_n(rmx)}{rmx} \\ \cos\phi \cdot \tau_n(\cos\theta) \frac{[rmx j_n(rmx)]'}{rmx} \\ -\sin\phi \cdot \pi_n(\cos\theta) \frac{[rmx j_n(rmx)]'}{rmx} \end{pmatrix} \tag{33}$$

$$\tag{34}$$

and the coordinate system is defined as for the scattered field. The vector-wave functions $\mathbf{N}$ and $\mathbf{M}$ are orthogonal with respect to integration over directions. Furthermore for different values of n, the $\mathbf{N}$ functions are orthogonal, too, and the same is true for the $\mathbf{M}$ functions.

2.5 Computation of Q_{abs}, based on the internal field

Python functions: mie_esquare, mie_abs The absorption cross section of a particle with dielectric (i.e. Ohmic) losses is given by (Ishimaru, 1978, p. 17)

$$\sigma_{\text{abs}} = k\epsilon'' \int_V |\mathbf{E}_1|^2 \, dV \tag{35}$$

where ϵ'' is the imaginary part of the relative dielectric constant of the particle (here with respect to the ambient medium). Thanks to the orthogonality of the spherical vector-wave functions this integral becomes in spherical coordinates

$$\sigma_{\text{abs}} = k\epsilon''\pi \sum_{n=1}^{\infty} \int_{-1}^{+1} d(\cos\theta) \int_0^a r^2 \, dr \left(|c_n|^2 (m_\theta + m_\phi) + |d_n|^2 (n_r + n_\theta + n_\phi) \right) \tag{36}$$

and the integration over azimuth ϕ has already been performed, leading to the factor π. The functions in the integrand are absolute-square values of

the series terms of the components of the vector-waves (34)

$$\begin{aligned} m_\theta &= g_n \pi_n^2(\cos\theta) \cdot |j_n(z)|^2 \\ m_\phi &= g_n \tau_n^2(\cos\theta) \cdot |j_n(z)|^2 \\ n_r &= g_n \sin^2\theta \cdot \pi_n^2(\cos\theta) \left| \frac{j_n(z)}{z} \right|^2 \\ n_\theta &= g_n \tau_n^2(\cos\theta) \left| \frac{(z j_n(z))'}{z} \right|^2 \\ n_\phi &= g_n \pi_n^2(\cos\theta) \left| \frac{(z j_n(z))'}{z} \right|^2 \end{aligned} \tag{37}$$

Here $z = mrk$, and g_n stands for

$$g_n \left(\frac{2n+1}{n(n+1)} \right)^2 \tag{38}$$

for the integrals over $\cos\theta$, analytic solutions can be obtained. First, from BH we find

$$\int_{-1}^{-1} (\pi_n^2(\cos theta) + (\tau_n^2(\cos theta))\, d(\cos\theta) = \frac{2n^2(n+1)^2}{2n+1} \tag{39}$$

and second, from (4.46) in BH and Equation 8.14.13 of Abramowitz and Stegun (1965), we get

$$\begin{aligned} \int_{-1}^{-1} (\sin^2\theta \cdot \pi_n^2(\cos\theta))\, d(\cos\theta) &= \int_{-1}^{-1} (P_n^1(\cos\theta))^2\, d(\cos\theta) \\ &= \frac{2(n+1)}{2n+1} \end{aligned} \tag{40}$$

leading to the two parts (41) and (42) of the angular integral in (36)

$$\begin{aligned} m_n &= \int_{-1}^{1} (m_\theta + m_\phi)\, d(\cos\theta) \\ &= 2(2n+1)|j_n(z)|^2 \end{aligned} \tag{41}$$

$$\begin{aligned} n_n &= \int_{-1}^{1} (n_r + n_\theta + n_\phi)\, d(\cos\theta) \\ &= 2n(2n+1) \left\{ (n+1) \left| \frac{j_n(z)}{z} \right|^2 + \left| \frac{(z j_n(z))'}{z} \right|^2 \right\} \end{aligned} \tag{42}$$

Now, the absorption cross section follows from integration over the radial distance r inside the sphere up to the sphere radius a:

$$\sigma_{\text{abs}} = k\epsilon''\pi \sum_{n=1}^{\infty} \int_0^a \left(m_n|c_n|^2 + n_n|d_n|^2\right) \cdot r^2 \, dr \tag{43}$$

The integrand contains the radial dependence of the absolute-square electric field $\langle|\mathbf{E}|^2\rangle$ averaged over spherical shells (all θ and ϕ, constant r):

$$\langle|\mathbf{E}|^2\rangle = \frac{1}{4} \sum_{n=1}^{\infty} \left(m_n|c_n|^2 + n_n|d_n|^2\right) \tag{44}$$

and in terms of this quantity, the absorption efficiency becomes

$$Q_{\text{abs}} = \frac{4\epsilon''}{x^2} \int_0^x \langle|\mathbf{E}|^2\rangle x'^2 \, dx' \tag{45}$$

where $x' = rk = z/m$. Note that (44) is dimensionless because of the unit-amplitude incident field; In case of Rayleigh scattering ($x \ll 1$) the internal field is constant, and the corresponding squared-field ratio (44) is given by

$$\frac{9}{|m^2 + 2|^2} \tag{46}$$

This quantity can be used to test the accuracy of the function, mie_esquare, for small size parameters. In addition, Equation 45 can be used to test the accuracy of the computation of Q_{abs} from the difference, $Q_{\text{ext}} - Q_{\text{sca}}$ (20). Finally, it should be remembered that all infinite series can be terminated after $n_{\max}$ terms.

3 The Python Programs

3.1 Commments

3.1.1 Restrictions

1. The present restrictions are situations with $\mu_1 = 1$; this means that the permittivity of the sphere relative to the ambient medium is given by $\epsilon = m^2$.
2. There is a maximum x value for stable and correct computation of Mie scattering. For a small or moderate imaginary refractive index m'', the maximum size parameter ranges somewhere between 10^4 and 10^5, however, for $m'' \gg 1$, the maximum size parameter is strongly diminished.

3.1.2 Computation of Bessel Functions

The ordinary Bessel Functions $J_\nu(z)$ and $Y_\nu(z)$ are standard functions in MATLAB. The spherical Bessel Functions used here follow from (4.9-10) of BH.

3.1.3 Computation of angular functions

The angular functions, π_n and τ_n, are computed from the recurrence relations (4.47) of BH with the initial functions given for $n = 1$ and 2.

3.1.4 Optimisation strategy

The programs were optimised rather with respect to computation speed than memory space; all required function values are computed only once and then stored for further use in vectors of dimension $n_{\max}$.

3.2 The Function Mie_abcd

The following text lists the basic program to compute the **Mie Coefficients** a_n, b_n, c_n, d_n and to produce a matrix of $n_{\max}$ column vectors $[[an][bn][cn][dn]]$:

```
import mpmath as mp

def mie_abcd(m, x):
    """
    Computes Mie coefficients a_n, b_n, c_n, d_n using mpmath.
    Translation of MATLAB code by C. Mätzler (2002).
    """

    # Maximum order
    nmax = round(2 + x + 4 * x**(1/3))
    n = [i for i in range(1, nmax + 1)]
    nu = [i + 0.5 for i in n]

    z = m * x
    m2 = m * m

    # Helper scaling factors
```

```
    sqx = mp.sqrt(0.5 * mp.pi / x)
    sqz = mp.sqrt(0.5 * mp.pi / z)

    # Riccati{Bessel functions (element wise)
    bx = [mp.besselj(v, x) * sqx for v in nu]
    bz = [mp.besselj(v, z) * sqz for v in nu]
    yx = [mp.bessely(v, x) * sqx for v in nu]
    hx = [bx[i] + 1j*yx[i] for i in range(nmax)]

    # Shifted values
    b1x = [mp.sin(x)/x] + bx[:nmax-1]
    b1z = [mp.sin(z)/z] + bz[:nmax-1]
    y1x = [-mp.cos(x)/x] + yx[:nmax-1]
    h1x = [b1x[i] + 1j*y1x[i] for i in range(nmax)]

    # Axial functions
    ax  = [x * b1x[i] - n[i] * bx[i] for i in range(nmax)]
    az  = [z * b1z[i] - n[i] * bz[i] for i in range(nmax)]
    ahx = [x * h1x[i] - n[i] * hx[i] for i in range(nmax)]

    # Mie coefficients
    an = [(m2*bz[i]*ax[i] - bx[i]*az[i])/(m2*bz[i]*ahx[i] - hx[i]*az[i]) for i in
    bn = [(bz[i]*ax[i] - bx[i]*az[i])/(bz[i]*ahx[i] - hx[i]*az[i]) for i in range(
    cn = [(bx[i]*ahx[i] - hx[i]*ax[i])/(bz[i]*ahx[i] - hx[i]*az[i]) for i in range
    dn = [m*(bx[i]*ahx[i] - hx[i]*ax[i])/(m2*bz[i]*ahx[i] - hx[i]*az[i]) for i in

    # Output as list of lists (same structure: 4 × nmax)
    return [an, bn, cn, dn]
```

3.3 The Function Mie

The following text lists the Program to compute the **Mie Efficiencies**:

```
def mie(m, x):
    """
    Python/Numpy version of the MATLAB function:
    [real(m) imag(m) x qext qsca qabs qb asy qratio]
    """
```

```
# If x == 0: singularity case (same as MATLAB)
if x == 0:
    return np.array([m.real, m.imag, 0, 0, 0, 0, 0, 0, 1.5], dtype=float)
# Normal case (x > 0)
nmax = int(round(2 + x + 4 * x ** (1/3)))
n = np.arange(1, nmax + 1)
cn = 2 * n + 1
c1n = n * (n + 2) / (n + 1)
c2n = cn / (n * (n + 1))
x2 = x * x

# Retrieve a_n and b_n coefficients
f = mie_abcd(m, x)  # expects shape (2, nmax)
an = f[0, :]
bn = f[1, :]

# Real/imaginary parts
anp = an.real
anpp = an.imag
bnp = bn.real
bnpp = bn.imag

# g1 shifted index family for asymmetry parameter
g1 = np.zeros((4, nmax))
g1[0, :-1] = anp[1:]
g1[1, :-1] = anpp[1:]
g1[2, :-1] = bnp[1:]
g1[3, :-1] = bnpp[1:]

# Efficiency calculations
dn = cn * (anp + bnp)
qext = 2 * np.sum(dn) / x2

en = cn * (anp**2 + anpp**2 + bnp**2 + bnpp**2)
qsca = 2 * np.sum(en) / x2

qabs = qext - qsca
```

```
    # Backscattering
    fn = (an - bn) * cn
    gn = (-1)**n
    back = np.sum(fn * gn)
    qb = (back * np.conj(back)).real / x2

    # Asymmetry parameter
    asy1 = c1n * (anp * g1[0] + anpp * g1[1] + bnp * g1[2] + bnpp * g1[3])
    asy2 = c2n * (anp * bnp + anpp * bnpp)
    asy = (4 / x2) * np.sum(asy1 + asy2) / qsca

    qratio = qb / qsca

    return np.array([m.real, m.imag, x, qext, qsca, qabs, qb, asy, qratio], dtype=
```

3.4 The Function Mie_S12

The following text lists the program to compute the two **complex scattering amplitudes** S_1 and S_2:

```
def mie_s12(m, x, u):
    """
    Compute Mie scattering functions S1 and S2.

    Parameters:
        m : complex
            Complex refractive index ratio
        x : float
            Size parameter (k0 * a)
        u : float or array-like
            Cosine of scattering angle (cos(theta))

    Returns:
        numpy.ndarray
            2 x N array, with S1 and S2 scattering amplitudes
    """

    # Ensure u is array for vectorized operations
```

```
    u = np.atleast_1d(u)

    # Order limit
    nmax = int(round(2 + x + 4 * x ** (1/3)))

    # Mie coefficients (shape: (2, nmax))
    abcd = Mie_abcd(m, x)
    an = abcd[0, :]   # shape (nmax,)
    bn = abcd[1, :]

    # Angular functions \pi_n and \tau_n (shape: (2, nmax, len(u)))
    pt = Mie_pt(u, nmax)
    pin = pt[0, :, :]  # \pi_n(u)
    tin = pt[1, :, :]  # \tau_n(u)

    # n = 1...nmax
    n = np.arange(1, nmax + 1)
    n2 = (2 * n + 1) / (n * (n + 1))

    # Apply normalization factor to angular functions
    pin = pin * n2[:, None]
    tin = tin * n2[:, None]

    # Compute S1 and S2 via vectorized summation over n
    S1 = np.sum(an[:, None] * pin + bn[:, None] * tin, axis=0)
    S2 = np.sum(an[:, None] * tin + bn[:, None] * pin, axis=0)

    return np.vstack((S1, S2))
```

3.5 The Function Mie_xscan

The following text lists the program to compute the a matrix of **Mie efficiencies** and to **plot** them as a function of x:

```
import numpy as np
import matplotlib.pyplot as plt

def mie_xscan(m, nsteps, dx):
```

```
"""
Compute and plot Mie efficiencies for a range of size parameters.

Parameters:
    m : complex
        Complex refractive index ratio
    nsteps : int
        Number of x increments
    dx : float
        Step size in x (size parameter)

Returns:
    numpy.ndarray
        Array of results, each row:
        [real(m), imag(m), x, qext, qsca, qabs, qb, asy, qratio]
"""

# Create array of x values starting at 0
x = np.arange(nsteps) * dx  # shape: (nsteps,)

# Allocate result array
results = np.zeros((nsteps, 9))

# Compute Mie results at each x
for j in range(nsteps):
    results[j, :] = Mie(m, x[j])

# Plot results (columns 3→8 correspond to qext...qratio)
plt.figure()
plt.plot(results[:, 2], results[:, 3:9])
plt.legend(['Qext', 'Qsca', 'Qabs', 'Qb', '<cos\theta>', 'Qb/Qsca'])
plt.title(f"Mie Efficiencies, m = {m.real:.3g} + {m.imag:.3g}i")
plt.xlabel("Size parameter x")
plt.grid(True)
plt.tight_layout()
plt.show()

return results
```

3.6 The Function Mie_thetascan

The following text lists the program to compute the matrix of **Mie scattering intensities** $|S_1|^2$ and $|S_2|^2$ as a function of $u = \cos\theta$, and to display the result as a polar diagram of θ with $|S_1|^2$ in the upper half circle ($0 < \theta < \pi$) and $|S_2|^2$ in the lower half circle ($\pi < \theta < 2\pi$). Both functions are symmetric with respect to both half circles:

```
def mie_tetascan(m, x, nsteps):
    """
    Computation and polar plot of Mie Power Scattering function for:
      m = complex refractive-index ratio
      x = size parameter
      nsteps = number of angular steps
    Translation from MATLAB code by C. Mätzler (2002)
    """

    m1 = np.real(m)
    m2 = np.imag(m)

    # Create theta array
    teta = np.linspace(0, np.pi, nsteps)

    SL = np.zeros(nsteps)
    SR = np.zeros(nsteps)

    for j in range(nsteps):
        u = np.cos(teta[j])
        a = mie_s12(m, x, u)
        SL[j] = np.real(a[0] * np.conj(a[0]))
        SR[j] = np.real(a[1] * np.conj(a[1]))

    # Mirror data like MATLAB code
    y_theta = np.concatenate((teta, teta + np.pi))
    y_vals = np.concatenate((SL, SR[::-1]))

    y = np.column_stack((y_theta, y_vals))

    # Polar Plot
```

```
    plt.figure()
    ax = plt.subplot(111, projection='polar')
    ax.plot(y[:, 0], y[:, 1])
    ax.set_title(f"Mie angular scattering: m={m1}+{m2}i, x={x}")
    ax.set_xlabel("Scattering Angle")
    plt.show()

    return y
```

3.7 The Function Mie_pt

The following text lists the program to compute a matrix of π_n and τ_n functions for $n = 1$ to $n_{\max}$:

```
import numpy as np

def mie_pt(u, nmax):
    """
    Compute the angular functions pi_n and tau_n used in Mie theory.
    Translated from MATLAB version by C. Mätzler (2002)

    Parameters:
        u : float  (cos(theta), must be in [-1, 1])
        nmax : int

    Returns:
        result : ndarray shape (2, nmax)
                 result[0, :] = pi_n
                 result[1, :] = tau_n
    """

    p = np.zeros(nmax)
    t = np.zeros(nmax)

    # Initial conditions
    p[0] = 1.0
    t[0] = u
    if nmax > 1:
```

```
        p[1] = 3 * u
        t[1] = 3 * np.cos(2 * np.arccos(u))

    # Recurrence
    for n1 in range(3, nmax + 1):
        n = n1 - 1  # MATLAB indexing shift
        p1 = (2*n1 - 1)/(n1 - 1) * p[n - 1] * u
        p2 = n1/(n1 - 1) * p[n - 2]
        p[n] = p1 - p2

        t1 = n1 * u * p[n]
        t2 = (n1 + 1) * p[n - 1]
        t[n] = t1 - t2

    return np.vstack((p, t))
```

3.8 The Function Mie_esquare

The following text lists the program to **compute** and **plot** the (θ, ϕ) averaged **absolute-square E-field** as a function of $x' = rk$ (for $r < 0 < a$):

```
mp.mp.dps = 50  # high precision recommended for Mie series computations

def mie_esquare(m, x, nj:
    """
    Mean-absolute-square internal electric field inside a sphere (Mie theory)
    Compute and plot mean-square internal electric field |E|^2 inside a sphere.

    Parameters:
        m  : complex refractive index ratio
        x  : size parameter (k0 * radius)
        nj : number of radial sample points

    Returns:
        een : numpy array of |E|^2 samples, length nj+1
    MATLAB Source: C. Mätzler (2002)
    """
```

```
# Determine max series term
nmax = int(round(2 + x + 4 * x**(1/3)))
n = np.arange(1, nmax + 1)
nu = n + 0.5

m1 = mp.re(m)
m2 = mp.im(m)

# Fetch cn & dn from Mie coefficients
abcd = mie_abcd(m, x)  # returns NumPy or Python lists convertible to mp
cn = np.array(abcd[2, :], dtype=object)
dn = np.array(abcd[3, :], dtype=object)

cn2 = np.abs(cn)**2
dn2 = np.abs(dn)**2

dx = x / nj
en = np.zeros(nj, dtype=float)

# Loop over radial steps
for j in range(1, nj + 1):
    xj = dx * j
    z = m * xj  # complex

    sqz = mp.sqrt(0.5 * mp.pi / z)

    # spherical j_n(z) = sqrt(pi/(2*z)) * J_{n+1/2}(z)
    bz = np.array([mp.besselj(nu[k], z) * sqz for k in range(len(n))], dtype=c
    bz2 = np.abs(bz)**2

    # j_0(z) = sin(z)/z
    b1z = np.zeros_like(bz, dtype=object)
    b1z[0] = mp.sin(z) / z
    b1z[1:] = bz[:-1]

    az = b1z - (n * bz) / z
    az2 = np.abs(az)**2
```

```
            z2 = np.abs(z)**2
            n1 = n * (n + 1)
            n2 = 2 * (2 * n + 1)

            mn = np.real(bz2 * n2)
            nn1 = az2
            nn2 = bz2 * n1 / z2
            nn = n2 * np.real(nn1 + nn2)

            en[j - 1] = 0.25 * (cn2 @ mn + dn2 @ nn)

    # Build radial coordinate array
    xxj = np.linspace(0, x, nj + 1)
    een = np.concatenate(([en[0]], en))

    # Plot results
    plt.figure()
    plt.plot(xxj, een)
    plt.title(f'Squared |E|² inside Sphere | m={m1}+{m2}i, x={x}')
    plt.xlabel('r k')
    plt.ylabel('|E|²')
    plt.grid(True)
    plt.legend(['Radial Dependence of (abs(E))²'])
    plt.show()

    return een
```

3.9 The Function Mie_abs

The following text lists the program to compute the **absorption efficiency**, based on Equation 45:

```
import numpy as np
import mpmath as mp

def mie_abs(m, x):
    """
    Absorption efficiency Qabs for a sphere using Mie theory.
```

```
Translation of MATLAB code by C. Mätzler (2002).

Parameters
----------
m : complex
    Refractive index (relative)
x : float
    Size parameter

Returns
-------
Qabs : float
    Absorption efficiency
"""

# Determine number of internal field samples
nj = 5 * round(2 + x + 4 * x**(1/3)) + 160

e2 = mp.im(m * m)
dx = x / nj
x2 = x * x

# Radial positions: 0, dx, 2dx, ..., x
xj = np.linspace(0, x, nj + 1)

# Compute |E|^2 internal field profile
en = mie_esquare(m, x, nj)  # assumes Mie_Esquare internally calls Mie_abcd

# End-term correction
en1 = 0.5 * en[-1] * x2

# Compute radial weighted field: en(r) * r^2 - correction
enx = en * (xj * xj) - en1

# Trapezoidal radial integration
inte = dx * np.sum(enx)

# Absorption Efficiency
```

```
Qabs = 4 * e2 * inte / x2

return float(mp.re(Qabs))
```

4 Examples and Tests

4.1 The situation of $x = 1, m = 5 + 0.4i$

4.2 Large size parameters

4.3 Large refractive index

5 Conclusion, and outlook to further developments