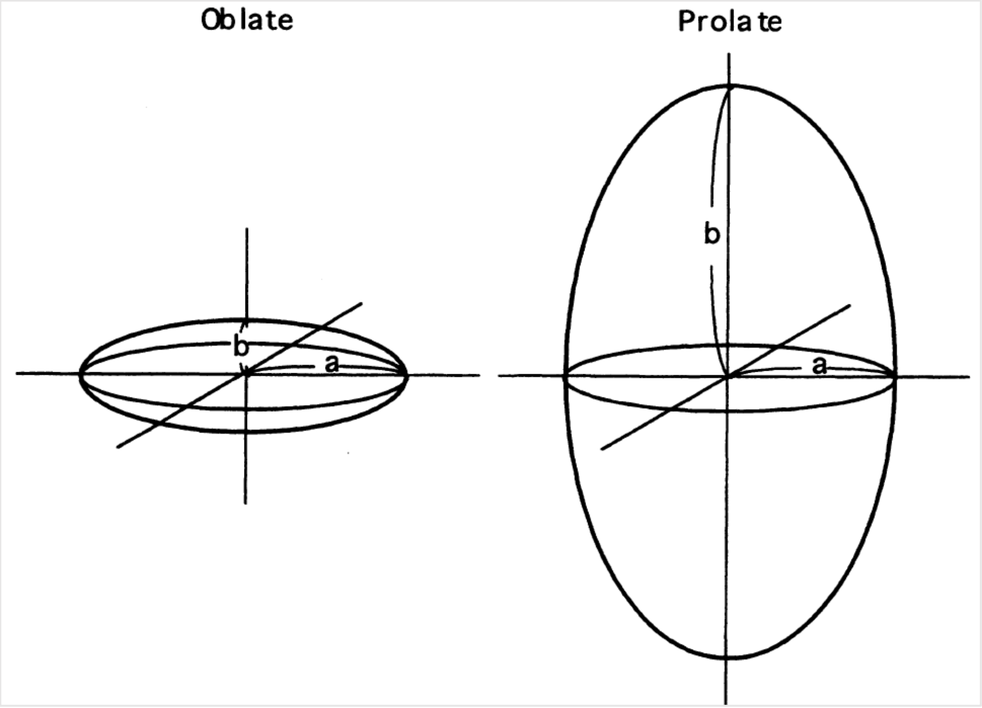
Spheroid Model: DLS Code Refactoring Notes

A group of randomly oriented, symmetric particles



- Optical characteristics: $\omega_0,\ \tau_{ext}\ \ (or\ C_{ext})$

$$F(\theta) = \begin{bmatrix} F_{11} & F_{12} & 0 & 0 \\ F_{12} & F_{22} & 0 & 0 \\ 0 & 0 & F_{33} & F_{34} \\ 0 & 0 & -F_{34} & F_{44} \end{bmatrix}, \quad f(\theta) = \frac{F(\theta)}{F_{11}(\theta)}$$

- Aspect ratio $\varepsilon = b/a$
- Size (radius) r $\quad V_{Spheroid} = V_{Sphere} = \frac{4}{3}\pi r^3$
- Size parameter $x = 2\pi r/\lambda$
- Kernel (LUT) $K_{ij}(\theta,\ r,\ n-ik,\ \varepsilon)$
- Fixed kernel $k_{ij}(\theta,\ r,\ n-ik) = \int_{\Delta\varepsilon} K_{ij}(\theta,\ r,\ n-ik,\ \varepsilon) D(\varepsilon)\, d\varepsilon \approx \mathbf{K}_{ij} \cdot \mathbf{D}\, \Delta\varepsilon$
- Log-spline or log-linear interpolation

$$f(x) \to \log(f(x)) \to \{i\} \to \log(f(y)) \to f(y) = \exp(\ldots)$$

What is DLS Code?

JOURNAL OF GEOPHYSICAL RESEARCH, VOL. 111, D11208, doi:10.1029/2005JD006619, 2006

Application of spheroid models to account for aerosol particle nonsphericity in remote sensing of desert dust

Oleg Dubovik,[1,2] Alexander Sinyuk,[1,3] Tatyana Lapyonok,[1,3] Brent N. Holben,[1] Michael Mishchenko,[4] Ping Yang,[5] Tom F. Eck,[1,6] Hester Volten,[7] Olga Muñoz,[8] Ben Veihelmann,[9] Wim J. van der Zande,[10] Jean-Francois Leon,[11] Michael Sorokin,[1,3] and Ilya Slutsker[1,3]

...

See Sec.2 for theory & numerical details

...

$$1.33 \le n \le 1.6,$$

$$0.0005 \le k \le 0.5, \qquad \leftarrow \textit{some absorption remains}$$

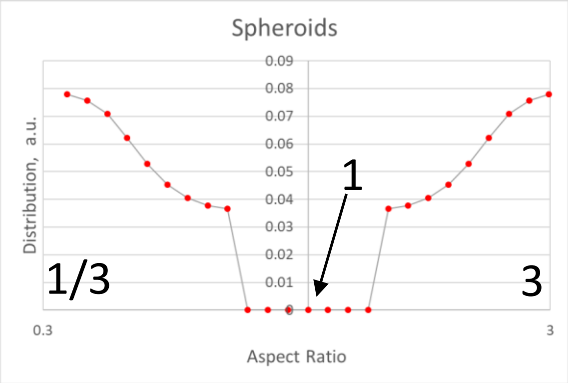
$$0.3 \le \varepsilon \le 3.0, \qquad (23)$$

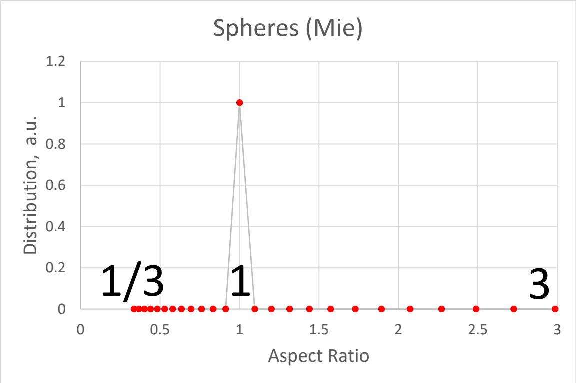
$$0.012 \le x (= 2\pi r/\lambda) \le 625, \quad \leftarrow \textit{wide range – different methods}$$

$$\Delta 1^\circ \text{ – resolution for } K_{ii'}(\Theta, \lambda, n, k, \varepsilon_p, r_k). \quad \leftarrow \textit{BIG Kernels, 5Gb in ASCII}$$

$$\lambda_{FIX} = 0.340\ (\mu m) \qquad \delta_{DLS} \sim 1\text{–}3\%\ (typical)$$

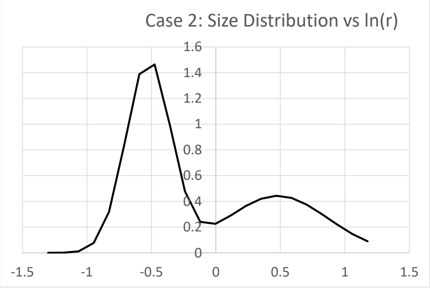
- Using BIG kernels *K*(...) and some aspect ratio, ε, distribution (note: Mie = spheres) ...





... one generalates fixed kernels *k*(...) (takes time – save as ASCII LUTs). For MAIAC, we did once in 2012.

- Using the fixed kernels *k*(...) , user's wavelength, refractive index, and particle size distribution (r_{user}) ...



... one generalates generates optical characteristics for MAIAC RT LUTs (on the fly)

One Slide Documentation: In → Out

```
#include "const_param_spheroids.h" // grids (e.g., size grid), constants (e.g., sizes of arrays, pi=3.14…), file names/paths

main():                 // Runs Cases 1 & 2;   in test_optichar.cpp;  for the rest: subroutine_name = file_name
  |
  +-optichar()          // for given input returns aerosol extinction, single scattering albedo, and normalized phase matrix
     |
     +-getix()  // for given x0 and X[] returns indices i1 & i2, so that X[i1] < x0 < X[i2]
     |
     +-interpolate_kernel_ext()
     |                 |
     |                 +-read_fixkernel_ext_bin()  // reads extinction and absorption fixed kernels
     |                 |
     |                 +-bilinear()                // performs bilinear interpolation (over refractive index)
     |
     +-interpolate_kernel_fij()
     |                 |
     |                 +-read_fixkernel_fij_bin()  // reads fixed kernels for phase matrix elements, fij
     |                 |
     |                 +-bilinear()
     |
     +-lognorm()  // calculates lognormal distribution
     |
     +-ddot()     // calculates dot product of 2 vectors
     |
     +-spline_grid()              // optional: interpolates from one grid into another using NAG cubic spline
     |       |
     |       +-e02baf() & e02bbf()  // optional: NAG cubic spline subroutines
     |
     +-simpson()                  // optional: numerical integration using Simpson's rule
```

$$D_V\left(x=\frac{r}{\lambda}\right)=\frac{C_V}{\sqrt{2\pi}\,\sigma}\exp\left(-\frac{\ln\left(x/x_m\right)^2}{2\sigma^2}\right)$$

$$f_{ij}=\int_{\Delta X} k_{ij}(\theta,\ x,\ n_0,k_0)D(x)\,d\ln(x)\approx \mathbf{K}_{ij}\cdot\mathbf{D}\,\Delta\ln(\mathrm{x})$$

In:

$\lambda \quad q_{Mie}$

$n \quad k>0$

$C_{vf} \quad C_{vc}$

$r_{mf} \quad r_{mc}$

$\sigma_{vf} \quad \sigma_{vc}$

$$x_0=\frac{1}{2}\int_0^{\pi} f_{11}(\theta)\sin(\theta)d\theta\approx 1$$

$$x_1=\frac{1}{2}\int_0^{\pi} f_{11}(\theta)\sin(\theta)\cos(\theta)d\theta$$

Out: $\omega_0,\ C_{ext},$

$$F(\theta)=\begin{bmatrix} F_{11} & f_{12} & 0 & 0\\ f_{12} & f_{22} & 0 & 0\\ 0 & 0 & f_{33} & f_{34}\\ 0 & 0 & -f_{34} & f_{44}\end{bmatrix},\quad f_{ij}=\frac{F_{ij}}{F_{11}}$$

$$\frac{\int\left(\ D_{VF}(x)+D_{VC}(x)\ \right)d\ln(x)}{C_{VF}+C_{VC}}\approx 1$$

See `readme.txt` for instructions & tests

/home/skorkin/spheroids/build - aeronet - Editor - WinSCP

Encoding · Color ·

```
g++ -Wall -W -c -O3 src/bilinear.cpp
g++ -Wall -W -c -O3 src/ddot.cpp
g++ -Wall -W -c -O3 src/getix.cpp
g++ -Wall -W -c -O3 src/interpolate_kernel_ext.cpp
g++ -Wall -W -c -O3 src/interpolate_kernel_fij.cpp
g++ -Wall -W -c -O3 src/lognorm.cpp
g++ -Wall -W -c -O3 src/optichar.cpp
g++ -Wall -W -c -O3 src/read_fixkernel_fij_bin.cpp
g++ -Wall -W -c -O3 src/read_fixkernel_ext_bin.cpp
g++ -Wall -W -c -O3 src/simpson.cpp
g++ -Wall -W -c -O3 src/spline_grid.cpp
g++ -Wall -W -c -O3 src/spline_nag.cpp
g++ -Wall -W -O3 src/test_optichar.cpp \
bilinear.o \
ddot.o \
getix.o \
interpolate_kernel_ext.o \
interpolate_kernel_fij.o \
lognorm.o \
optichar.o \
read_fixkernel_fij_bin.o \
read_fixkernel_ext_bin.o \
simpson.o \
spline_grid.o \
spline_nag.o \
-o run
rm *.o
```

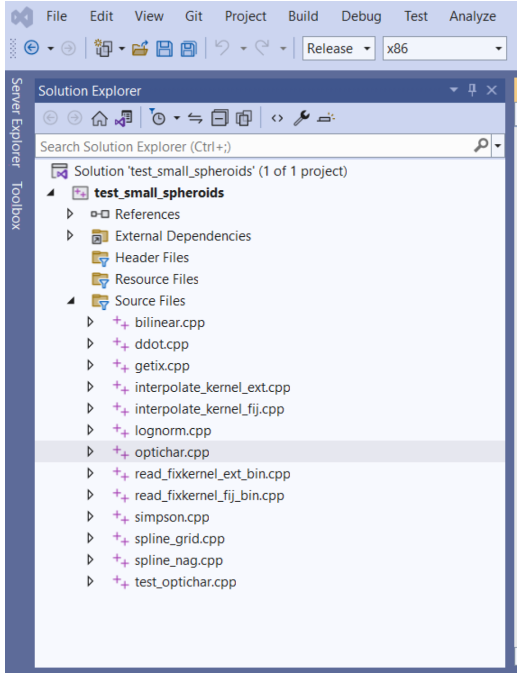
/home/skorkin/spheroids/readme.txt - aeronet - Editor - WinSCP

Encoding · Color ·

```
1. check path to kernels in const_param_spheroids.h (line 17)
   default is rootdir[path_len_max] = "./kernels_fix_bin/"
2. make sure 'build' is executable
   if not run chmod +x build
3. execute 'build' to build (ignore warnings)
4. execute 'run' to run
5. you should get this:

[skorkin@gs618-aerol1 spheroids]$ time ./run

Case 1 inp: wavel = 0.600
            mie_fraq = 0.700  refre = 1.60  refim = 0.00
            cvf = 0.50  radf = 0.30  sgmf = 0.40
            cvc = 1.00  radc = 3.00  sgmc = 0.90
Case 1 out:
  0.0    1.016008e+02   9.997294e-01   9.997294e-01   9.994589e-01   -0.000000e+00    0.000000e+00
  1.0    6.220902e+01   9.995592e-01   9.995460e-01   9.991066e-01   -1.154759e-04   -2.714780e-03
  2.0    3.925060e+01   9.993048e-01   9.992119e-01   9.985243e-01   -2.712795e-04   -6.797968e-03
  3.0    2.796145e+01   9.990311e-01   9.987746e-01   9.978275e-01   -3.672112e-04   -1.061744e-02
  4.0    2.172044e+01   9.987637e-01   9.982716e-01   9.970805e-01   -3.888452e-04   -1.376277e-02
  5.0    1.793710e+01   9.985168e-01   9.977399e-01   9.963354e-01   -3.531180e-04   -1.621641e-02
  6.0    1.546706e+01   9.982951e-01   9.972065e-01   9.956238e-01   -2.868358e-04   -1.815239e-02
  7.0    1.374882e+01   9.980966e-01   9.966855e-01   9.949587e-01   -2.135548e-04   -1.978601e-02
  8.0    1.248442e+01   9.979169e-01   9.961805e-01   9.943392e-01   -1.504886e-04   -2.130423e-02
  9.0    1.150547e+01   9.977509e-01   9.956875e-01   9.937566e-01   -1.080306e-04   -2.284536e-02
 10.0    1.071260e+01   9.975948e-01   9.951991e-01   9.931986e-01   -9.368139e-05   -2.449713e-02
 11.0    1.004511e+01   9.974453e-01   9.947061e-01   9.926523e-01   -1.149890e-04   -2.631430e-02
 12.0    9.464612e+00   9.972992e-01   9.941965e-01   9.921028e-01   -1.739252e-04   -2.832544e-02
 13.0    8.937836e+00   9.971424e-01   9.935101e-01   9.913892e-01   -2.834205e-04   -3.061301e-02
 14.0    8.474269e+00   9.970065e-01   9.930801e-01   9.909372e-01   -4.035975e-04   -3.293958e-02
 15.0    8.037483e+00   9.968554e-01   9.924524e-01   9.902967e-01   -5.706051e-04   -3.552128e-02
 16.0    7.628801e+00   9.966984e-01   9.917679e-01   9.896059e-01   -7.689842e-04   -3.826014e-02
 17.0    7.243392e+00   9.965340e-01   9.910261e-01   9.888636e-01   -9.946065e-04   -4.113392e-02
 18.0    6.877935e+00   9.963609e-01   9.902331e-01   9.880755e-01   -1.241838e-03   -4.412321e-02
 19.0    6.530111e+00   9.961782e-01   9.893970e-01   9.872494e-01   -1.506794e-03   -4.720896e-02
 20.0    6.198333e+00   9.959850e-01   9.885220e-01   9.863895e-01   -1.790618e-03   -5.037304e-02
 21.0    5.881550e+00   9.957804e-01   9.876013e-01   9.854891e-01   -2.097386e-03   -5.360001e-02
 22.0    5.579034e+00   9.955645e-01   9.866233e-01   9.845358e-01   -2.431636e-03   -5.687389e-02
 23.0    5.290212e+00   9.953367e-01   9.855760e-01   9.835179e-01   -2.796993e-03   -6.017743e-02
 24.0    5.014601e+00   9.950956e-01   9.844490e-01   9.824258e-01   -3.195151e-03   -6.349271e-02
 25.0    4.751790e+00   9.948406e-01   9.832355e-01   9.812527e-01   -3.626759e-03   -6.680126e-02
 26.0    4.501395e+00   9.945704e-01   9.819297e-01   9.799935e-01   -4.093385e-03   -7.008554e-02
 27.0    4.263041e+00   9.942846e-01   9.805282e-01   9.786452e-01   -4.596143e-03   -7.333017e-02
 28.0    4.036377e+00   9.939820e-01   9.790268e-01   9.772042e-01   -5.136973e-03   -7.652057e-02
 29.0    3.821034e+00   9.936619e-01   9.774226e-01   9.756676e-01   -5.716705e-03   -7.964252e-02
 30.0    3.616643e+00   9.933235e-01   9.757128e-01   9.740331e-01   -6.335532e-03   -8.268380e-02
 31.0    3.422803e+00   9.929662e-01   9.738952e-01   9.722991e-01   -6.994298e-03   -8.563487e-02
 32.0    3.239128e+00   9.925893e-01   9.719659e-01   9.704621e-01   -7.693968e-03   -8.848319e-02
```



Tests -- 2 cases: q*Mie+(1-q)*Srd, Cvf + Cvc

Parameter	Notation	Units	Case 1	Case 2	Comments
Wavelength	λ	µm	0.6	0.4	Red & blue
Refractive index: real part	n	-	1.6	1.3	n = [1.29 : 1.70]
Refractive index: imaginary part	k	-	0.001	0.1	k = [0.0005 : 0.5] > 0
Mean radius, fine fraction	rvf	µm	0.3	0.3	Same in both cases
Width parameter, fine fraction	σf	?	0.4	0.4	Same in both cases
Concentration, fine fraction	Cvf	a. u.	0.5	1.5	a.u. – arbitrary units
Mean radius, coarse fraction	rvc	µm	3.0	3.0	Same in both
Width parameter, coarse fraction	σc	?	0.9	0.9	Same in both
Concentration, coarse fraction	Cvc	a. u.	1.0	1.0	a.u. – arbitrary units
Mie fraction	q	r. u.	0.7	0.3	q = 1 means Mie only

$$d_V(r) = \frac{C_V}{\sqrt{2\pi}\,\sigma}\exp\left(-\frac{\ln\left(r/r_m\right)}{2\sigma^2}\right) \rightarrow d_V(x) = \frac{C_V}{\sqrt{2\pi}\,\sigma}\exp\left(-\frac{\ln\left(x/x_m\right)}{2\sigma^2}\right); \quad C_V = \int_X d_V(x)\,d\ln x$$

Numerical results (5 digits) & summary of changes

- **v.0: DLS** – original DLS package
- **v.1: cpp** – C version with kernels in *.bin files, flip sequence of interpolation – first over refractive index (bilinear), then r-spline
- **v.2: cpp** – same loglin interpolation for all fij (not reflected in the table); see slide “”
- **v.3: cpp** – lognormal distribution over size parameter (full range as in kernels, no r-spline); no artificial smoothing of f33 & f44 (next slide)

$$C_{ext}^{Kernels} \times \underbrace{1000 \left(\frac{\mu m^3}{\mu m^2} ? \right) \frac{\lambda_{User}}{\lambda_{Fix}}}_{scalef\ -\ ?}$$

$$x_0 = \frac{1}{2}\int_0^{\pi} f_{11}(\theta)\sin(\theta)\,d\theta$$

$$C_{Sca}^{K} F_{ij} \sim K_{ij}$$

$$F_{ij} \sim \frac{K_{ij}}{C_{Ext}^{K} - C_{Abs}^{K}} \frac{scalef}{scalef}$$

$$x_1 = \frac{1}{2}\int_0^{\pi} f_{11}(\theta)\sin(\theta)\cos(\theta)\,d\theta$$

Parameter	Case 1				Case 2			
	v.0: DLS	v.1: cpp	v.2: cpp	v.3: cpp	v.0: DLS	v.1: cpp	v.2: cpp	v.3: cpp
Extinction	5.1080	5.1080	5.1080	5.1098	9.8634	9.8634	9.8634	9.8643
S. S. Albedo	0.98922	0.98922	0.98922	0.98913	0.59104	0.59104	0.59104	0.59100
x0 = intg{F11}	0.99768	0.99720	0.99722	0.99749	1.0001	0.99902	0.99902	0.99965
intg{ F11/x0 }	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F11(0)	89.951	89.988	89.988	101.60	199.64	199.86	199.86	220.86
F11(90)	0.26762	0.26771	0.26771	0.26774	0.067541	0.067597	0.067597	0.067541
F11(180)	0.57250	0.57268	0.57268	0.57301	0.027849	0.027881	0.027881	0.027849
Aver. S. Cos	0.66682	0.66672	0.66672	0.66667	0.87397	0.87387	0.87387	0.87386

Graphical results: smoothing

```
//  Smooth f33 & f44 - as in the DLS code
//  f33:
    isca1 = 38;
    isca2 = 50;
    sca1 = sca_fix[isca1];
    f1 = log(f33[isca1]);
    sca2 = sca_fix[isca2];
    f2 = log(f33[isca2]);
    for (isca = isca1; isca < isca2+1; isca++)
        f33[isca] = exp( linear(sca_fix[isca], sca1, sca2, f1, f2) );
```

Hard-coded in DLS

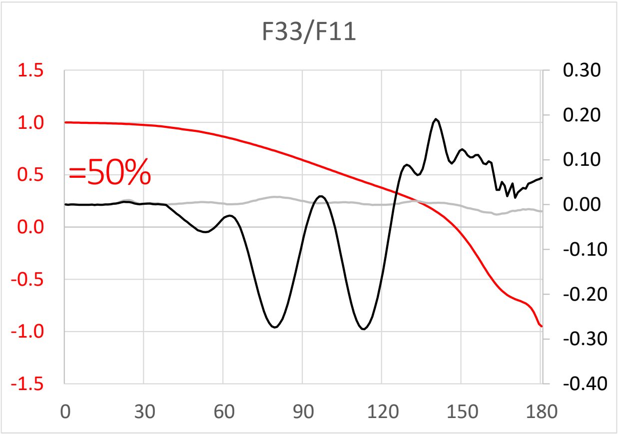
```
//  f44:
    isca1 = 48;
    isca2 = 60;
    sca1 = sca_fix[isca1];
    f1 = log(f44[isca1]);
    sca2 = sca_fix[isca2];
    f2 = log(f44[isca2]);
    for (isca = isca1; isca < isca2+1; isca++)
        f44[isca] = exp( linear(sca_fix[isca], sca1, sca2, f1, f2) );
```

F44 is not used IPOL and many other vRT codes

Case 1

Case 2

Smoothed

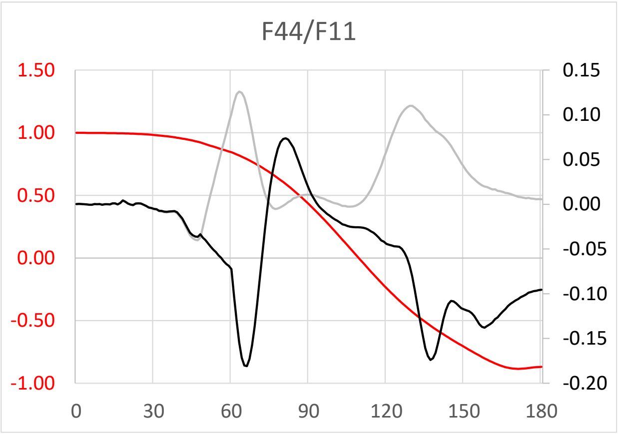


%

—v.0: DLS

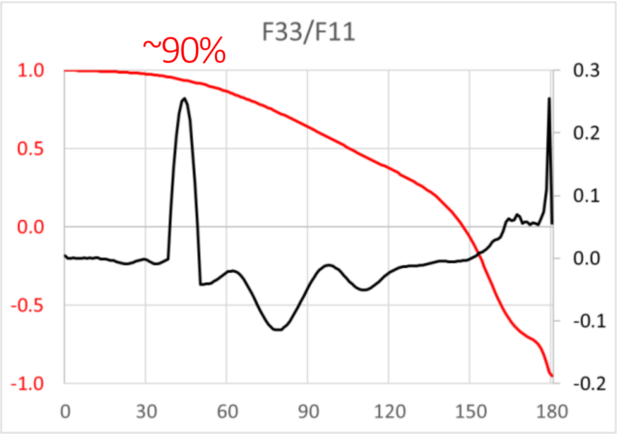
—v.0 vs v.1 in %

—v.0 vs v.2 in %

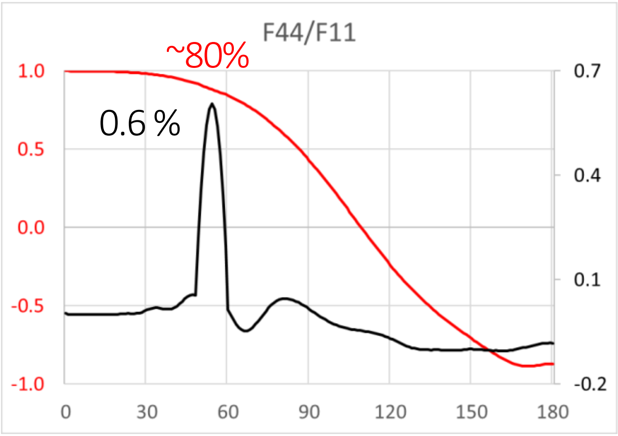


%

Not: v.3 vs DLS



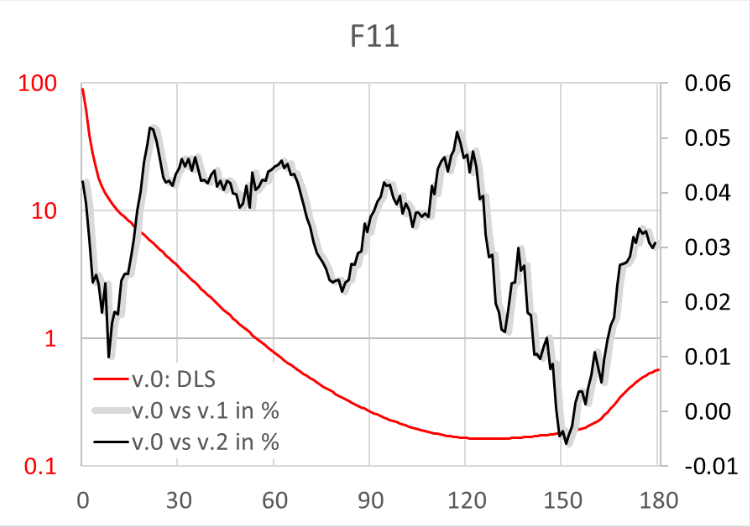
%

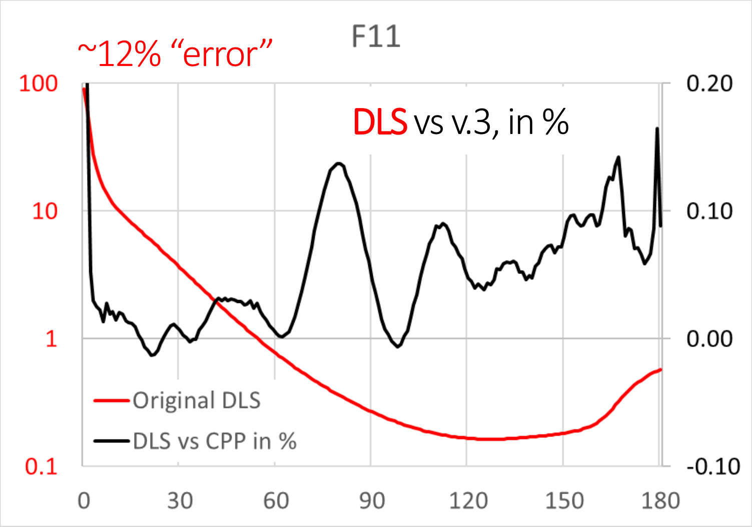


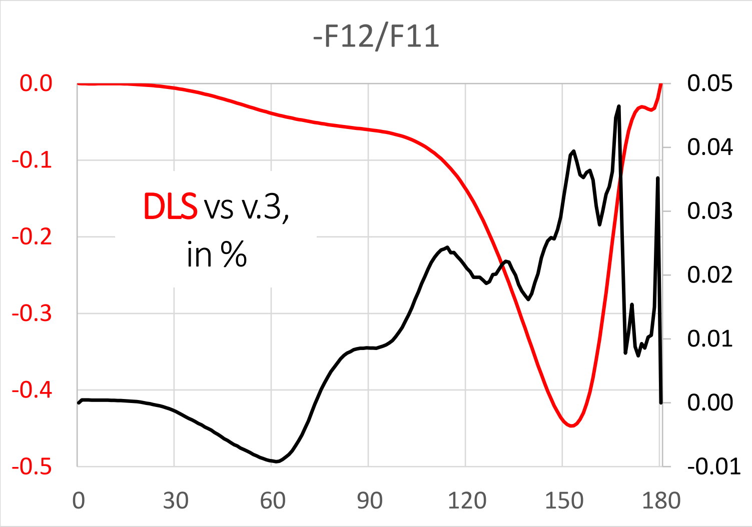
%

Graphical results: general

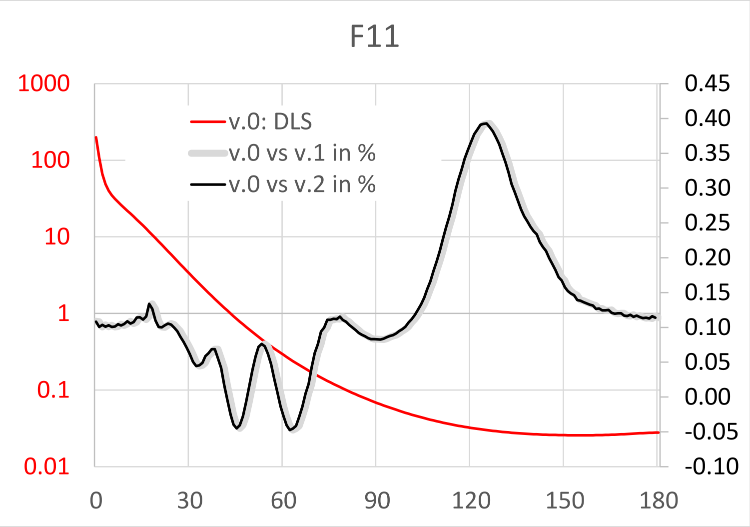
Case 1

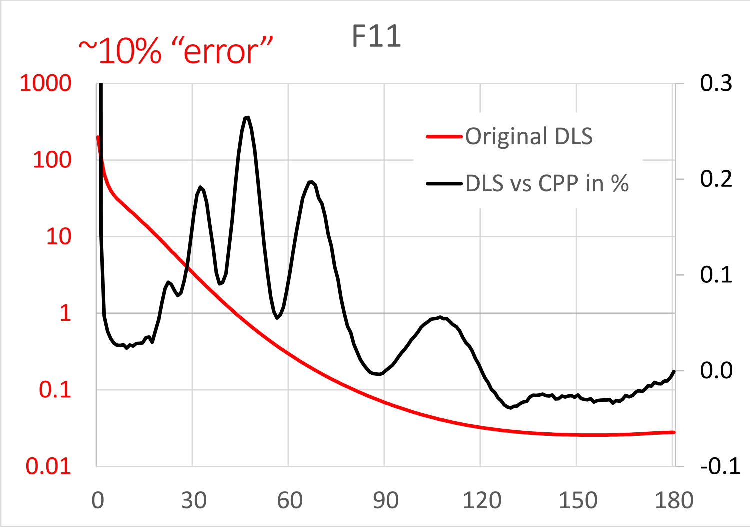


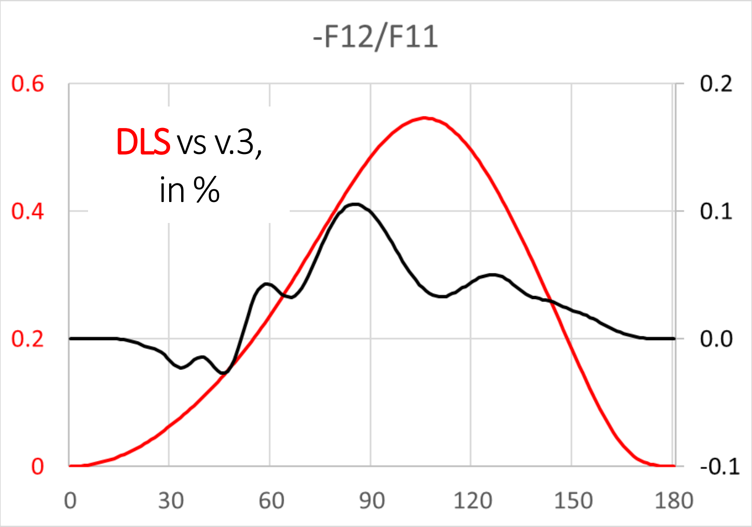




Case 2







PS: What if F & C fractions (or Mie & Srd… or both…) have different, e.g. n-ik, … or other parameters ?

n_1 k_1 0.3 0

```
code = optichar(wavel, refre, refim, mie_fraq, cvf, radf, sgmf, cvc, radc, sgmc,
                f11, f22, f33, f44, f12, f34, cext, ssa, conc_ratio);
```

F_1 ε_1 ω_1

Example:

Cvf / Cvc = 0.3 / 1

n_2 k_2 0 1

```
code = optichar(wavel, refre, refim, mie_fraq, cvf, radf, sgmf, cvc, radc, sgmc,
                f11, f22, f33, f44, f12, f34, cext, ssa, conc_ratio);
```

F_2 ε_2 ω_2

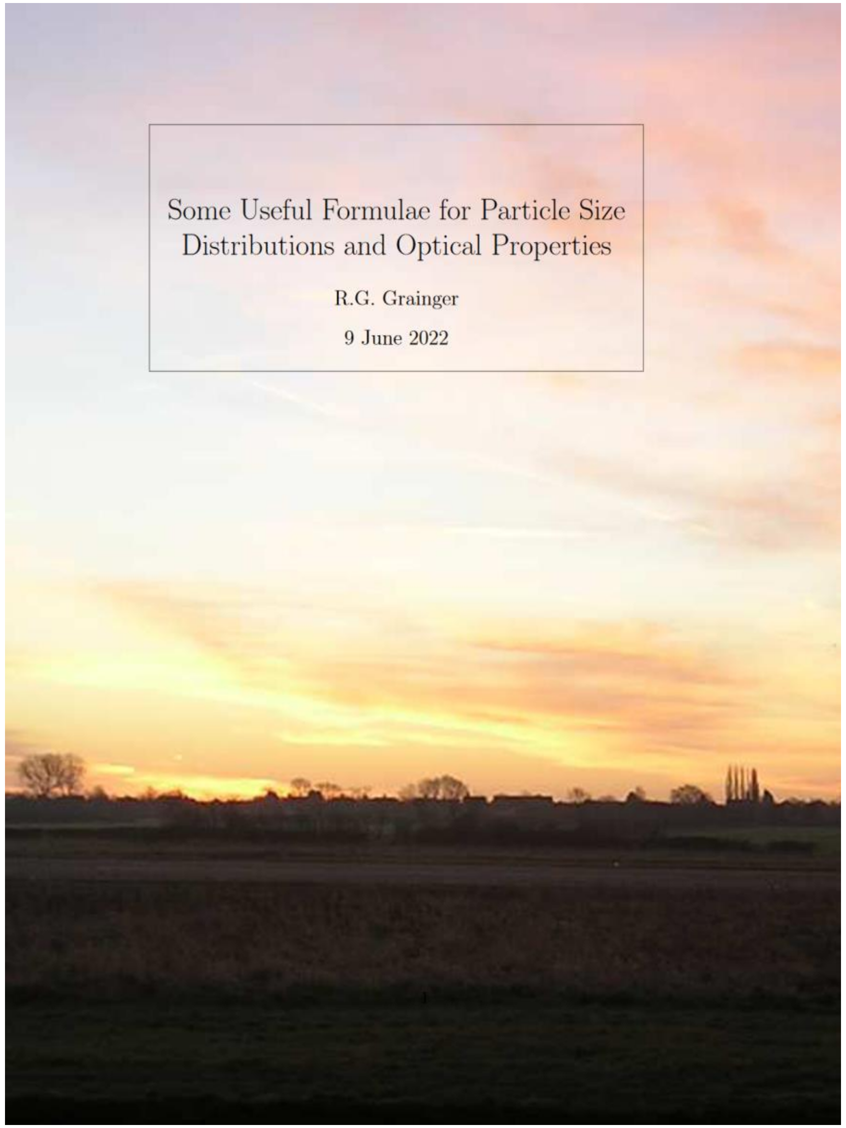
$$\varepsilon = \varepsilon_1 + \varepsilon_2 \qquad s = s_1 + s_2 \qquad F = \frac{s_1}{s} F_1 + \frac{s_2}{s} F_2$$

$$s_i = \omega_i \, \varepsilon_i \qquad \omega = s / \varepsilon$$

Q: How to predict unpredictable & account for unaccountable? Where will our research lead us tomorrow?

A: The new package must have maximum flexibility. Discussion to follow…

PS: Very good (concise!) summary on the topic



1 Describing Particle Size

Atmospheric particles come in a variety of compositions and sizes. There are three main classes: aerosols, water droplets and ice crystals. But that is just a start, a complete description of an ensemble of particles would encompass the composition and geometry of each particle. Such an approach is impracticable. For example atmospheric aerosols have concentrations as high as $\sim 10,000$ particles per cm^3. An alternate approach is to use a statistical description of the particle size distribution. This is assisted by the fact that small liquid drops adopt a spherical shape so that for a chemically homogeneous particle the problem becomes one of representing the number distribution of particle radii. The particle size distribution can be represented in tabular form but it is usual to adopt an analytic functional. The success of this approach hinges upon the selection of an appropriate size distribution function that approximates the actual distribution. There is no *a priori* reason for assuming this can be done.

1.1 Particle Size Distribution

A measured distribution of particle sizes can be described by a histogram of the number of particles per unit volume within defined size bins. By making the bin size tend to zero a continuous function is formed called the radius number density distribution $N(r)$ which represents the number of particles with radii between r and $r + dr$ per unit volume. The difficulty with this representation is that is is unmeasurable! This is because there are an infinite number of radius values so that the probability of the radius being any one specific value is zero. The problem is avoided by using the differential radius number density distribution $n(r)$ defined by

$$n(r) = \frac{dN(r)}{dr}. \tag{1}$$

The same equation can be written in integral form as

$$dN(r) = \int_r^{r+dr} n(r)\,dr. \tag{2}$$

The total number of particles per unit volume, N_0, is given by

$$N_0 = \int_0^\infty n(r)\,dr. \tag{3}$$

The total surface area of the distribution or surface area density is defined

$$a_V = 4\pi \int_0^\infty r^2 n(r)\,dr. \tag{4}$$

6.2 Back Scatter

■ to be done ■

6.3 Phase Function

The phase function represents the redistribution of the scattered energy.
For a collection of particles, the phase function is given by

$$P(\lambda,\theta) = \frac{1}{\beta^{sca}} \int_0^\infty \pi r^2 Q^{sca}(\lambda,r) P(\lambda,r,\theta) n(r)\,dr. \tag{126}$$

6.4 Single Scatter Albedo

The single scatter albedo is the ratio of the energy scattered from a particle to that intercepted by the particle. Hence

$$\omega(\lambda) = \frac{\beta^{sca}(\lambda)}{\beta^{ext}(\lambda)}. \tag{127}$$

6.5 Asymmetry Parameter

The asymmetry parameter is the average cosine of the scattering angle, weighted by the intensity of the scattered light as a function of angle. It has the value 1 for perfect forward scattering, 0 for isotropic scattering and -1 for perfect backscatter.

$$g = \frac{1}{\beta^{sca}} \int_0^\infty \pi r^2 Q^{sca}(\lambda,r) g(\lambda,r) n(r)\,dr \tag{128}$$

6.6 Integration

The integration of an optical properties over size is usually reduced from the interval $r = [0,\infty]$ to $r = [r_l, r_u]$ as $n(r) \to 0$ as $r \to 0$ and $r \to \infty$. Numerically an integral over particle size becomes

$$\int_{r_l}^{r_u} f(r) n(r)\,dr = \sum_{i=1}^{n} w_i f(r_i) \tag{129}$$

where w_i are the weights at discrete values of radius, r_i.
For a log normal size distribution the integrals are

$$\beta^{ext}(\lambda) = \frac{N_0}{\sigma}\sqrt{\frac{\pi}{2}} \int_{r_l}^{r_u} r Q^{ext}(\lambda,r) \exp\left[-\frac{1}{2}\left(\frac{\ln r - \ln r_m}{\sigma}\right)^2\right] dr \tag{130}$$

https://eodg.atm.ox.ac.uk/user/grainger/research/aerosols.pdf