#pragma TextEncoding = "UTF-8"

#pragma rtGlobals=3 // Use modern global access method and strict wave access.

///////////////////////////////////////////////////////////////////////////////////////////////

// Development of a Universal Correction Algorithm for Filter-Based Absorption Photometers//

// Igor Pro code for the algorithms (A,B,C) //

///////////////////////////////////////////////////////////////////////////////////////////////

// Code by: Hanyang Li, li.4860@osu.edu //

//////////////////////////////////////////////////////////////////////////////////////////////

//load the data

LoadWave /A/Q/O/J/D/W/K=0 "C:Users:Hanyang:Desktop:FIREX\_CLAP.csv"

//=====================================list of variables used in the code===================//

//reference absorption coefficient (e.g., from a Photoacoustic Extinctiometer): babs\_red\_ref, babs\_green\_ref, babs\_blue\_ref

//scattering coefficient (e.g., from a Nephelometer): bscat\_red, bscat\_green, bscat\_blue

//attenuation coefficient measured by the Filter-Based Absorption Photometer: batn\_red, batn\_green, batn\_blue

//transmission measured by the Filter-Based Absorption Photometer: Tr\_red, Tr\_green, Tr\_blue

//========================================================================================//

function Algo\_Li\_et\_al\_2019()

if (exists("babs\_red\_ref")==1 && exists("babs\_green\_ref")==1 && exists("babs\_blue\_ref")==1) // reference absorption coefficient exists

if (exists("bscat\_red")==1 && exists("bscat\_green")==1 && exists("bscat\_blue")==1) // scattering coefficient exists

print "Implement Algorithm A"

Algo\_A()

else

// scattering coefficient does not exist

print "The scenario is not considered in the manuscript. We recommend using the photoacoustic Babs."

endif

else // reference absorption coefficient does not exist

if (exists("bscat\_red")==1 && exists("bscat\_green")==1 && exists("bscat\_blue")==1) // scattering coefficient exists

print "Implement Algorithm B"

Algo\_B()

else

// scattering coefficient does not exist

print "Implement Algorithm C"

Algo\_C()

endif

endif

end

function Algo\_A()

wave babs\_red\_ref,babs\_green\_ref,babs\_blue\_ref,bscat\_red,bscat\_green,bscat\_blue,batn\_red,batn\_green,batn\_blue,Tr\_red,Tr\_green,Tr\_blue

variable npnts=numpnts(batn\_red)

//initialize AAE, using batn

AAE\_power\_fit(batn\_red,batn\_green,batn\_blue)

wave AAE\_three\_lambda

make /N=(npnts)/O AAE\_temp=AAE\_three\_lambda //"\_temp" means that the term is a temporary variable, which is updated in the following loop

//initialize SSA, using batn and bscat\_ref

make /N=(npnts)/O SSA\_red\_temp=bscat\_red/(bscat\_red+batn\_red)

make /N=(npnts)/O SSA\_green\_temp=bscat\_green/(bscat\_green+batn\_green)

make /N=(npnts)/O SSA\_blue\_temp=bscat\_blue/(bscat\_blue+batn\_blue)

//initialize set of coefficients

make/N=8/O W\_coef\_temp\_red = {0,0,0,0,0,0,0,0}

make/N=8/O W\_coef\_temp\_green = {0,0,0,0,0,0,0,0}

make/N=8/O W\_coef\_temp\_blue = {0,0,0,0,0,0,0,0}

//compute "g" term in Eq. (9), where g(Tr,SSA,AAE)=babs\_ref/batn

make /N=(npnts)/O g\_term\_red\_temp=babs\_red\_ref/batn\_red

make /N=(npnts)/O g\_term\_green\_temp=babs\_green\_ref/batn\_green

make /N=(npnts)/O g\_term\_blue\_temp=babs\_blue\_ref/batn\_blue

//compute ln\_Tr

make /N=(npnts)/O ln\_Tr\_red=ln(Tr\_red)

make /N=(npnts)/O ln\_Tr\_green=ln(Tr\_green)

make /N=(npnts)/O ln\_Tr\_blue=ln(Tr\_blue)

variable loop\_number=0

print "////////////iterative process////////////" //show major results in the command window

do

loop\_number+=1

print "Loop No.", loop\_number

//temporarily save the outputs from the previous iteration

duplicate /O AAE\_temp,AAE\_previous\_cycle

//implement "Algorithm A" on each wavelength

make/D/N=8/O W\_coef=W\_coef\_temp\_red

FuncFit/Q Algorithm\_function W\_coef g\_term\_red\_temp /X={ln\_Tr\_red,SSA\_red\_temp,AAE\_temp} /w=batn\_red

W\_coef\_temp\_red=W\_coef

make/D/N=8/O W\_coef =W\_coef\_temp\_green

FuncFit/Q Algorithm\_function W\_coef g\_term\_green\_temp /X={ln\_Tr\_green,SSA\_green\_temp,AAE\_temp} /w=batn\_green

W\_coef\_temp\_green=W\_coef

make/D/N=8/O W\_coef =W\_coef\_temp\_blue

FuncFit/Q Algorithm\_function W\_coef g\_term\_blue\_temp /X={ln\_Tr\_blue,SSA\_blue\_temp,AAE\_temp} /w=batn\_blue

W\_coef\_temp\_blue=W\_coef

if (loop\_number==1) //output coefficients for future uses

Concatenate /O/NP=2 {W\_coef\_temp\_red,W\_coef\_temp\_green,W\_coef\_temp\_blue}, W\_coef\_initial\_guess\_all\_wvl

endif

//Calculate Babs for each wavelength using Eq. (8) in the manuscript

make /N=(npnts)/O b\_abs\_red\_temp=batn\_red\*(W\_coef\_temp\_red[0]+W\_coef\_temp\_red[1]\*ln\_Tr\_red+W\_coef\_temp\_red[2]\*SSA\_red\_temp+W\_coef\_temp\_red[3]\*AAE\_temp+W\_coef\_temp\_red[4]\*ln\_Tr\_red\*SSA\_red\_temp+W\_coef\_temp\_red[5]\*SSA\_red\_temp\*AAE\_temp+W\_coef\_temp\_red[6]\*ln\_Tr\_red\*AAE\_temp+W\_coef\_temp\_red[7]\*ln\_Tr\_red\*AAE\_temp\*SSA\_red\_temp)

make /N=(npnts)/O b\_abs\_green\_temp=batn\_green\*(W\_coef\_temp\_green[0]+W\_coef\_temp\_green[1]\*ln\_Tr\_green+W\_coef\_temp\_green[2]\*SSA\_green\_temp+W\_coef\_temp\_green[3]\*AAE\_temp+W\_coef\_temp\_green[4]\*ln\_Tr\_green\*SSA\_green\_temp+W\_coef\_temp\_green[5]\*SSA\_green\_temp\*AAE\_temp+W\_coef\_temp\_green[6]\*ln\_Tr\_green\*AAE\_temp+W\_coef\_temp\_green[7]\*ln\_Tr\_green\*AAE\_temp\*SSA\_green\_temp)

make /N=(npnts)/O b\_abs\_blue\_temp=batn\_blue\*(W\_coef\_temp\_blue[0]+W\_coef\_temp\_blue[1]\*ln\_Tr\_blue+W\_coef\_temp\_blue[2]\*SSA\_blue\_temp+W\_coef\_temp\_blue[3]\*AAE\_temp+W\_coef\_temp\_blue[4]\*ln\_Tr\_blue\*SSA\_blue\_temp+W\_coef\_temp\_blue[5]\*SSA\_blue\_temp\*AAE\_temp+W\_coef\_temp\_blue[6]\*ln\_Tr\_blue\*AAE\_temp+W\_coef\_temp\_blue[7]\*ln\_Tr\_blue\*AAE\_temp\*SSA\_blue\_temp)

//create new g\_term, AAE and SSA

g\_term\_red\_temp=b\_abs\_red\_temp/batn\_red

g\_term\_green\_temp=b\_abs\_green\_temp/batn\_green

g\_term\_blue\_temp=b\_abs\_blue\_temp/batn\_blue

AAE\_power\_fit(b\_abs\_red\_temp,b\_abs\_green\_temp,b\_abs\_blue\_temp)

AAE\_temp=AAE\_three\_lambda

SSA\_red\_temp=bscat\_red/(bscat\_red+b\_abs\_red\_temp)

SSA\_green\_temp=bscat\_green/(bscat\_green+b\_abs\_green\_temp)

SSA\_blue\_temp=bscat\_blue/(bscat\_blue+b\_abs\_blue\_temp)

//check if convergence is reached (tolerance is set to be 1.5%)

CurveFit/Q/ODR=2 line AAE\_temp /x=AAE\_previous\_cycle /D

while (W\_coef[1]>1.015||W\_coef[1]<0.985)

//Output final results

duplicate/O AAE\_temp,AAE\_final

duplicate/O SSA\_red\_temp,SSA\_red\_final

duplicate/O SSA\_green\_temp,SSA\_green\_final

duplicate/O SSA\_blue\_temp,SSA\_blue\_final

duplicate/O b\_abs\_red\_temp,b\_abs\_red\_final

duplicate/O b\_abs\_green\_temp,b\_abs\_green\_final

duplicate/O b\_abs\_blue\_temp,b\_abs\_blue\_final

KillWaves g\_term\_red\_temp,g\_term\_green\_temp,g\_term\_blue\_temp,SSA\_red\_temp,SSA\_green\_temp,SSA\_blue\_temp,b\_abs\_red\_temp,b\_abs\_green\_temp,b\_abs\_blue\_temp,AAE\_previous\_cycle

edit W\_coef\_initial\_guess\_all\_wvl

end

function Algo\_B()

wave bscat\_red,bscat\_green,bscat\_blue,batn\_red,batn\_green,batn\_blue,Tr\_red,Tr\_green,Tr\_blue

variable npnts=numpnts(batn\_red)

//initialize AAE, using batn

AAE\_power\_fit(batn\_red,batn\_green,batn\_blue)

wave AAE\_three\_lambda

make /N=(npnts)/O AAE\_temp=AAE\_three\_lambda //"\_temp" means that the term is a temporary variable, which is updated in the following loop

//initialize SSA, using batn and bscat\_ref

make /N=(npnts)/O SSA\_red\_temp=bscat\_red/(bscat\_red+batn\_red)

make /N=(npnts)/O SSA\_green\_temp=bscat\_green/(bscat\_green+batn\_green)

make /N=(npnts)/O SSA\_blue\_temp=bscat\_blue/(bscat\_blue+batn\_blue)

//compute ln\_Tr

make /N=(npnts)/O ln\_Tr\_red=ln(Tr\_red)

make /N=(npnts)/O ln\_Tr\_green=ln(Tr\_green)

make /N=(npnts)/O ln\_Tr\_blue=ln(Tr\_blue)

//Initialize set of coefficients. FIREX\_CLAP coefficients are used in the current code.

//Recommended initial guesses for the other combinations of

//aerosol type and filter-based absorption photometer can be found in the manuscript.

Make/D/N=8/O W\_coef\_temp\_red= {0.27, -0.16, -0.18, -0.05, 0.18, 0.08, 0.01, 0.03}

Make/D/N=8/O W\_coef\_temp\_green= {0.30, -0.28, -0.18, -0.07, 0.25, 0.10, 0.12, -0.17}

Make/D/N=8/O W\_coef\_temp\_blue= {0.32, -0.38, -0.19, -0.08, 0.33, 0.12, 0.23, -0.31}

//Calculate temporary Babs for each wavelength using Eq. (8) in the manuscript

make /n=(npnts)/o b\_abs\_red\_temp=batn\_red\*(W\_coef\_temp\_red[0]+W\_coef\_temp\_red[1]\*ln\_Tr\_red+W\_coef\_temp\_red[2]\*SSA\_red\_temp+W\_coef\_temp\_red[3]\*AAE\_temp+W\_coef\_temp\_red[4]\*ln\_Tr\_red\*SSA\_red\_temp+W\_coef\_temp\_red[5]\*SSA\_red\_temp\*AAE\_temp+W\_coef\_temp\_red[6]\*ln\_Tr\_red\*AAE\_temp+W\_coef\_temp\_red[7]\*ln\_Tr\_red\*AAE\_temp\*SSA\_red\_temp)

make /n=(npnts)/o b\_abs\_green\_temp=batn\_green\*(W\_coef\_temp\_green[0]+W\_coef\_temp\_green[1]\*ln\_Tr\_green+W\_coef\_temp\_green[2]\*SSA\_green\_temp+W\_coef\_temp\_green[3]\*AAE\_temp+W\_coef\_temp\_green[4]\*ln\_Tr\_green\*SSA\_green\_temp+W\_coef\_temp\_green[5]\*SSA\_green\_temp\*AAE\_temp+W\_coef\_temp\_green[6]\*ln\_Tr\_green\*AAE\_temp+W\_coef\_temp\_green[7]\*ln\_Tr\_green\*AAE\_temp\*SSA\_green\_temp)

make /n=(npnts)/o b\_abs\_blue\_temp=batn\_blue\*(W\_coef\_temp\_blue[0]+W\_coef\_temp\_blue[1]\*ln\_Tr\_blue+W\_coef\_temp\_blue[2]\*SSA\_blue\_temp+W\_coef\_temp\_blue[3]\*AAE\_temp+W\_coef\_temp\_blue[4]\*ln\_Tr\_blue\*SSA\_blue\_temp+W\_coef\_temp\_blue[5]\*SSA\_blue\_temp\*AAE\_temp+W\_coef\_temp\_blue[6]\*ln\_Tr\_blue\*AAE\_temp+W\_coef\_temp\_blue[7]\*ln\_Tr\_blue\*AAE\_temp\*SSA\_blue\_temp)

//compute "g" term in Eq. (9), where g(Tr,SSA,AAE)=babs\_ref/batn

make /N=(npnts)/O g\_term\_red\_temp=b\_abs\_red\_temp/batn\_red

make /N=(npnts)/O g\_term\_green\_temp=b\_abs\_green\_temp/batn\_green

make /N=(npnts)/O g\_term\_blue\_temp=b\_abs\_blue\_temp/batn\_blue

variable loop\_number=0

print "////////////iterative process////////////" //show major results in the command window

do

loop\_number+=1

print "Loop No.", loop\_number

//temporarily save the outputs from the previous iteration

duplicate /O AAE\_temp,AAE\_previous\_cycle

//implement "Algorithm A" on each wavelength

make/D/N=8/O W\_coef=W\_coef\_temp\_red

FuncFit/Q Algorithm\_function W\_coef g\_term\_red\_temp /X={ln\_Tr\_red,SSA\_red\_temp,AAE\_temp} /w=batn\_red

W\_coef\_temp\_red=W\_coef

make/D/N=8/O W\_coef =W\_coef\_temp\_green

FuncFit/Q Algorithm\_function W\_coef g\_term\_green\_temp /X={ln\_Tr\_green,SSA\_green\_temp,AAE\_temp} /w=batn\_green

W\_coef\_temp\_green=W\_coef

make/D/N=8/O W\_coef =W\_coef\_temp\_blue

FuncFit/Q Algorithm\_function W\_coef g\_term\_blue\_temp /X={ln\_Tr\_blue,SSA\_blue\_temp,AAE\_temp} /w=batn\_blue

W\_coef\_temp\_blue=W\_coef

//Update Babs for each wavelength using Eq. (8) in the manuscript

b\_abs\_red\_temp=batn\_red\*(W\_coef\_temp\_red[0]+W\_coef\_temp\_red[1]\*ln\_Tr\_red+W\_coef\_temp\_red[2]\*SSA\_red\_temp+W\_coef\_temp\_red[3]\*AAE\_temp+W\_coef\_temp\_red[4]\*ln\_Tr\_red\*SSA\_red\_temp+W\_coef\_temp\_red[5]\*SSA\_red\_temp\*AAE\_temp+W\_coef\_temp\_red[6]\*ln\_Tr\_red\*AAE\_temp+W\_coef\_temp\_red[7]\*ln\_Tr\_red\*AAE\_temp\*SSA\_red\_temp)

b\_abs\_green\_temp=batn\_green\*(W\_coef\_temp\_green[0]+W\_coef\_temp\_green[1]\*ln\_Tr\_green+W\_coef\_temp\_green[2]\*SSA\_green\_temp+W\_coef\_temp\_green[3]\*AAE\_temp+W\_coef\_temp\_green[4]\*ln\_Tr\_green\*SSA\_green\_temp+W\_coef\_temp\_green[5]\*SSA\_green\_temp\*AAE\_temp+W\_coef\_temp\_green[6]\*ln\_Tr\_green\*AAE\_temp+W\_coef\_temp\_green[7]\*ln\_Tr\_green\*AAE\_temp\*SSA\_green\_temp)

b\_abs\_blue\_temp=batn\_blue\*(W\_coef\_temp\_blue[0]+W\_coef\_temp\_blue[1]\*ln\_Tr\_blue+W\_coef\_temp\_blue[2]\*SSA\_blue\_temp+W\_coef\_temp\_blue[3]\*AAE\_temp+W\_coef\_temp\_blue[4]\*ln\_Tr\_blue\*SSA\_blue\_temp+W\_coef\_temp\_blue[5]\*SSA\_blue\_temp\*AAE\_temp+W\_coef\_temp\_blue[6]\*ln\_Tr\_blue\*AAE\_temp+W\_coef\_temp\_blue[7]\*ln\_Tr\_blue\*AAE\_temp\*SSA\_blue\_temp)

//create new g\_term, AAE and SSA

g\_term\_red\_temp=b\_abs\_red\_temp/batn\_red

g\_term\_green\_temp=b\_abs\_green\_temp/batn\_green

g\_term\_blue\_temp=b\_abs\_blue\_temp/batn\_blue

AAE\_power\_fit(b\_abs\_red\_temp,b\_abs\_green\_temp,b\_abs\_blue\_temp)

AAE\_temp=AAE\_three\_lambda

SSA\_red\_temp=bscat\_red/(bscat\_red+b\_abs\_red\_temp)

SSA\_green\_temp=bscat\_green/(bscat\_green+b\_abs\_green\_temp)

SSA\_blue\_temp=bscat\_blue/(bscat\_blue+b\_abs\_blue\_temp)

//check if convergence is reached (tolerance is set to be 1.5%)

CurveFit/Q/ODR=2 line AAE\_temp /x=AAE\_previous\_cycle /D

while (W\_coef[1]>1.015||W\_coef[1]<0.985)

//Output the final results

duplicate/O AAE\_temp,AAE\_final

duplicate/O SSA\_red\_temp,SSA\_red\_final

duplicate/O SSA\_green\_temp,SSA\_green\_final

duplicate/O SSA\_blue\_temp,SSA\_blue\_final

duplicate/O b\_abs\_red\_temp,b\_abs\_red\_final

duplicate/O b\_abs\_green\_temp,b\_abs\_green\_final

duplicate/O b\_abs\_blue\_temp,b\_abs\_blue\_final

KillWaves g\_term\_red\_temp,g\_term\_green\_temp,g\_term\_blue\_temp,SSA\_red\_temp,SSA\_green\_temp,SSA\_blue\_temp,b\_abs\_red\_temp,b\_abs\_green\_temp,b\_abs\_blue\_temp,AAE\_previous\_cycle

end

function Algo\_C()

wave batn\_red,batn\_green,batn\_blue,Tr\_red,Tr\_green,Tr\_blue

variable npnts=numpnts(batn\_red)

//initialize AAE, using batn

AAE\_power\_fit(batn\_red,batn\_green,batn\_blue)

wave AAE\_three\_lambda

make /N=(npnts)/O AAE\_temp=AAE\_three\_lambda //"\_temp" means that the term is a temporary variable, which is updated in the following loop

//initialize SSA, using AAE-SSA relationship

variable a\_652=1.19,b\_652=1.49,c\_652=1.84

variable a\_528=1.16,b\_528=1.44,c\_528=1.68

variable a\_467=1.13,b\_467=1.41,c\_467=1.59

make /n=(npnts)/o SSA\_red\_temp=((AAE\_temp-a\_652)/b\_652)^(1/c\_652)

make /n=(npnts)/o SSA\_green\_temp=((AAE\_temp-a\_528)/b\_528)^(1/c\_528)

make /n=(npnts)/o SSA\_blue\_temp=((AAE\_temp-a\_467)/b\_467)^(1/c\_467)

//compute bscat using SSA and Batn

make /n=(npnts)/o bscat\_red=SSA\_red\_temp/(1-SSA\_red\_temp)\*batn\_red

make /n=(npnts)/o bscat\_green=SSA\_green\_temp/(1-SSA\_green\_temp)\*batn\_green

make /n=(npnts)/o bscat\_blue=SSA\_blue\_temp/(1-SSA\_blue\_temp)\*batn\_blue

//compute ln\_Tr

make /N=(npnts)/O ln\_Tr\_red=ln(Tr\_red)

make /N=(npnts)/O ln\_Tr\_green=ln(Tr\_green)

make /N=(npnts)/O ln\_Tr\_blue=ln(Tr\_blue)

//Initialize a set of coefficients. FIREX\_CLAP coefficients are used in the current code.

//Recommended initial guesses for the other combinations of

//aerosol type and filter-based absorption photometer can be found in the manuscript.

Make/D/N=8/O W\_coef\_temp\_red= {0.27, -0.16, -0.18, -0.05, 0.18, 0.08, 0.01, 0.03}

Make/D/N=8/O W\_coef\_temp\_green= {0.30, -0.28, -0.18, -0.07, 0.25, 0.10, 0.12, -0.17}

Make/D/N=8/O W\_coef\_temp\_blue= {0.32, -0.38, -0.19, -0.08, 0.33, 0.12, 0.23, -0.31}

//Calculate temporary Babs for each wavelength using Eq. (8) in the manuscript

make /n=(npnts)/o b\_abs\_red\_temp=batn\_red\*(W\_coef\_temp\_red[0]+W\_coef\_temp\_red[1]\*ln\_Tr\_red+W\_coef\_temp\_red[2]\*SSA\_red\_temp+W\_coef\_temp\_red[3]\*AAE\_temp+W\_coef\_temp\_red[4]\*ln\_Tr\_red\*SSA\_red\_temp+W\_coef\_temp\_red[5]\*SSA\_red\_temp\*AAE\_temp+W\_coef\_temp\_red[6]\*ln\_Tr\_red\*AAE\_temp+W\_coef\_temp\_red[7]\*ln\_Tr\_red\*AAE\_temp\*SSA\_red\_temp)

make /n=(npnts)/o b\_abs\_green\_temp=batn\_green\*(W\_coef\_temp\_green[0]+W\_coef\_temp\_green[1]\*ln\_Tr\_green+W\_coef\_temp\_green[2]\*SSA\_green\_temp+W\_coef\_temp\_green[3]\*AAE\_temp+W\_coef\_temp\_green[4]\*ln\_Tr\_green\*SSA\_green\_temp+W\_coef\_temp\_green[5]\*SSA\_green\_temp\*AAE\_temp+W\_coef\_temp\_green[6]\*ln\_Tr\_green\*AAE\_temp+W\_coef\_temp\_green[7]\*ln\_Tr\_green\*AAE\_temp\*SSA\_green\_temp)

make /n=(npnts)/o b\_abs\_blue\_temp=batn\_blue\*(W\_coef\_temp\_blue[0]+W\_coef\_temp\_blue[1]\*ln\_Tr\_blue+W\_coef\_temp\_blue[2]\*SSA\_blue\_temp+W\_coef\_temp\_blue[3]\*AAE\_temp+W\_coef\_temp\_blue[4]\*ln\_Tr\_blue\*SSA\_blue\_temp+W\_coef\_temp\_blue[5]\*SSA\_blue\_temp\*AAE\_temp+W\_coef\_temp\_blue[6]\*ln\_Tr\_blue\*AAE\_temp+W\_coef\_temp\_blue[7]\*ln\_Tr\_blue\*AAE\_temp\*SSA\_blue\_temp)

//compute "g" term in Eq. (9), where g(Tr,SSA,AAE)=babs\_ref/batn

make /N=(npnts)/O g\_term\_red\_temp=b\_abs\_red\_temp/batn\_red

make /N=(npnts)/O g\_term\_green\_temp=b\_abs\_green\_temp/batn\_green

make /N=(npnts)/O g\_term\_blue\_temp=b\_abs\_blue\_temp/batn\_blue

variable loop\_number=0

print "////////////iterative process////////////" //show major results in the command window

do

loop\_number+=1

print "Loop No.", loop\_number

//temporarily save the outputs from the previous iteration

duplicate /O AAE\_temp,AAE\_previous\_cycle

//implement "Algorithm A" on each wavelength

make/D/N=8/O W\_coef=W\_coef\_temp\_red

FuncFit/Q Algorithm\_function W\_coef g\_term\_red\_temp /X={ln\_Tr\_red,SSA\_red\_temp,AAE\_temp} /w=batn\_red

W\_coef\_temp\_red=W\_coef

make/D/N=8/O W\_coef =W\_coef\_temp\_green

FuncFit/Q Algorithm\_function W\_coef g\_term\_green\_temp /X={ln\_Tr\_green,SSA\_green\_temp,AAE\_temp} /w=batn\_green

W\_coef\_temp\_green=W\_coef

make/D/N=8/O W\_coef =W\_coef\_temp\_blue

FuncFit/Q Algorithm\_function W\_coef g\_term\_blue\_temp /X={ln\_Tr\_blue,SSA\_blue\_temp,AAE\_temp} /w=batn\_blue

W\_coef\_temp\_blue=W\_coef

//Update Babs for each wavelength using Eq. (8) in the manuscript

b\_abs\_red\_temp=batn\_red\*(W\_coef\_temp\_red[0]+W\_coef\_temp\_red[1]\*ln\_Tr\_red+W\_coef\_temp\_red[2]\*SSA\_red\_temp+W\_coef\_temp\_red[3]\*AAE\_temp+W\_coef\_temp\_red[4]\*ln\_Tr\_red\*SSA\_red\_temp+W\_coef\_temp\_red[5]\*SSA\_red\_temp\*AAE\_temp+W\_coef\_temp\_red[6]\*ln\_Tr\_red\*AAE\_temp+W\_coef\_temp\_red[7]\*ln\_Tr\_red\*AAE\_temp\*SSA\_red\_temp)

b\_abs\_green\_temp=batn\_green\*(W\_coef\_temp\_green[0]+W\_coef\_temp\_green[1]\*ln\_Tr\_green+W\_coef\_temp\_green[2]\*SSA\_green\_temp+W\_coef\_temp\_green[3]\*AAE\_temp+W\_coef\_temp\_green[4]\*ln\_Tr\_green\*SSA\_green\_temp+W\_coef\_temp\_green[5]\*SSA\_green\_temp\*AAE\_temp+W\_coef\_temp\_green[6]\*ln\_Tr\_green\*AAE\_temp+W\_coef\_temp\_green[7]\*ln\_Tr\_green\*AAE\_temp\*SSA\_green\_temp)

b\_abs\_blue\_temp=batn\_blue\*(W\_coef\_temp\_blue[0]+W\_coef\_temp\_blue[1]\*ln\_Tr\_blue+W\_coef\_temp\_blue[2]\*SSA\_blue\_temp+W\_coef\_temp\_blue[3]\*AAE\_temp+W\_coef\_temp\_blue[4]\*ln\_Tr\_blue\*SSA\_blue\_temp+W\_coef\_temp\_blue[5]\*SSA\_blue\_temp\*AAE\_temp+W\_coef\_temp\_blue[6]\*ln\_Tr\_blue\*AAE\_temp+W\_coef\_temp\_blue[7]\*ln\_Tr\_blue\*AAE\_temp\*SSA\_blue\_temp)

//create new g\_term, AAE and SSA

g\_term\_red\_temp=b\_abs\_red\_temp/batn\_red

g\_term\_green\_temp=b\_abs\_green\_temp/batn\_green

g\_term\_blue\_temp=b\_abs\_blue\_temp/batn\_blue

AAE\_power\_fit(b\_abs\_red\_temp,b\_abs\_green\_temp,b\_abs\_blue\_temp)

AAE\_temp=AAE\_three\_lambda

SSA\_red\_temp=bscat\_red/(bscat\_red+b\_abs\_red\_temp)

SSA\_green\_temp=bscat\_green/(bscat\_green+b\_abs\_green\_temp)

SSA\_blue\_temp=bscat\_blue/(bscat\_blue+b\_abs\_blue\_temp)

//check if convergence is reached (tolerance is set to be 1.5%)

CurveFit/Q/ODR=2 line AAE\_temp /x=AAE\_previous\_cycle /D

while (W\_coef[1]>1.015||W\_coef[1]<0.985)

//Output the final results

duplicate/O AAE\_temp,AAE\_final

duplicate/O SSA\_red\_temp,SSA\_red\_final

duplicate/O SSA\_green\_temp,SSA\_green\_final

duplicate/O SSA\_blue\_temp,SSA\_blue\_final

duplicate/O b\_abs\_red\_temp,b\_abs\_red\_final

duplicate/O b\_abs\_green\_temp,b\_abs\_green\_final

duplicate/O b\_abs\_blue\_temp,b\_abs\_blue\_final

KillWaves g\_term\_red\_temp,g\_term\_green\_temp,g\_term\_blue\_temp,SSA\_red\_temp,SSA\_green\_temp,SSA\_blue\_temp,b\_abs\_red\_temp,b\_abs\_green\_temp,b\_abs\_blue\_temp,AAE\_previous\_cycle

end

function AAE\_power\_fit(babs\_wvl\_red,babs\_wvl\_green,babs\_wvl\_blue)

wave babs\_wvl\_red,babs\_wvl\_green,babs\_wvl\_blue

//the operated wavelengths are specified by the manufacture

make /O/N=3 wavelength\_instrument={652,528,467}

variable npnts=numpnts(babs\_wvl\_red)

//compute AAE\_two\_lambda, which are used later when initializing AAE\_three\_lambda

make /N=(npnts)/O AAE\_R\_G=-ln(babs\_wvl\_red/babs\_wvl\_green)/ln(wavelength\_instrument[0]/wavelength\_instrument[1])

make /N=(npnts)/O AAE\_R\_B=-ln(babs\_wvl\_red/babs\_wvl\_blue)/ln(wavelength\_instrument[0]/wavelength\_instrument[2])

make /N=(npnts)/O AAE\_G\_B=-ln(babs\_wvl\_green/babs\_wvl\_blue)/ln(wavelength\_instrument[1]/wavelength\_instrument[2])

make /N=(npnts)/O AAE\_three\_lambda=nan

//initialize coefficients in babs-AAE function

variable temp\_AAE,temp\_AAE\_constant,temp\_sigma,temp\_sigma\_constant

variable m

for (m=0;m<(npnts);m+=1)

if (numtype(babs\_wvl\_red[m])==0&&numtype(babs\_wvl\_green[m])==0&&numtype(babs\_wvl\_blue[m])==0)

make /O /N=3 one\_babs\_at\_three\_lambda={babs\_wvl\_red[m],babs\_wvl\_green[m],babs\_wvl\_blue[m]}

////use AAE\_G\_B as initial input

make/D/N=2/O W\_coef = {AAE\_G\_B[m],babs\_wvl\_green[m]/ (528^(-AAE\_G\_B[m])) }

FuncFit/Q AAE\_fit\_one\_measurement W\_coef one\_babs\_at\_three\_lambda /X=wavelength\_instrument /D

wave W\_coef,w\_sigma

temp\_AAE=W\_coef[0];temp\_AAE\_constant=W\_coef[1];temp\_sigma=w\_sigma[0];temp\_sigma\_constant=w\_sigma[1]

////use AAE\_R\_G as initial input

make/D/N=2/O W\_coef = {AAE\_R\_G[m],babs\_wvl\_red[m]/ (652^(-AAE\_R\_G[m])) }

FuncFit/Q AAE\_fit\_one\_measurement W\_coef one\_babs\_at\_three\_lambda /X=wavelength\_instrument /D

//check if the current inital guess (AAE\_R\_G) is better than the previous one (AAE\_G\_B): ? w\_sigma is smaller

if (w\_sigma[0]<temp\_sigma) //Yes, the current w\_sigma is smaller

temp\_AAE=W\_coef[0];temp\_AAE\_constant=W\_coef[1];temp\_sigma=w\_sigma[0];temp\_sigma\_constant=w\_sigma[1]

else

//do not replace the previous results

endif

////use AAE\_R\_B as initial input

make/D/N=2/O W\_coef = {AAE\_R\_B[m],babs\_wvl\_blue[m]/ (467^(-AAE\_R\_B[m])) }

FuncFit/Q AAE\_fit\_one\_measurement W\_coef one\_babs\_at\_three\_lambda /X=wavelength\_instrument /D

//check if the current initial guess (AAE\_R\_B) is better than the previous one (AAE\_G\_B or AAE\_R\_G): ? w\_sigma is smaller

if (w\_sigma[0]<temp\_sigma) //Yes, the current w\_sigma is smaller

temp\_AAE=W\_coef[0];temp\_AAE\_constant=W\_coef[1];temp\_sigma=w\_sigma[0];temp\_sigma\_constant=w\_sigma[1]

else

//do not replace the previous results

endif

//Output the results

AAE\_three\_lambda[m]=temp\_AAE

temp\_AAE=nan;temp\_AAE\_constant=nan;temp\_AAE=nan;temp\_sigma=nan

endif

endfor

end

Function AAE\_fit\_one\_measurement(w,wavelength) : FitFunc

Wave w

Variable wavelength

//CurveFitDialog/ These comments were created by the Curve Fitting dialog. Altering them will

//CurveFitDialog/ make the function less convenient to work with in the Curve Fitting dialog.

//CurveFitDialog/ Equation:

//CurveFitDialog/ f(wavelength) = AAE\_constant\*wavelength^(-AAE\_cal)

//CurveFitDialog/ End of Equation

//CurveFitDialog/ Independent Variables 1

//CurveFitDialog/ wavelength

//CurveFitDialog/ Coefficients 2

//CurveFitDialog/ w[0] = AAE\_cal

//CurveFitDialog/ w[1] = AAE\_constant

return w[1]\*wavelength^(-w[0])

End

Function Algorithm\_function(w,ln\_tr\_in\_function,SSA\_in\_function, AAE\_in\_function) : FitFunc

Wave w

Variable ln\_tr\_in\_function

Variable SSA\_in\_function

Variable AAE\_in\_function

//CurveFitDialog/ These comments were created by the Curve Fitting dialog. Altering them will

//CurveFitDialog/ make the function less convenient to work with in the Curve Fitting dialog.

//CurveFitDialog/ Equation:

//CurveFitDialog/ f(ln\_tr\_in\_function,SSA\_in\_function,aae\_in\_function) = c0+c1\*ln\_tr\_in\_function+c2\*SSA\_in\_function+c3\*aae\_in\_function+c4\*ln\_tr\_in\_function\*SSA\_in\_function+c5\*SSA\_in\_function\*aae\_in\_function+c6\*ln\_tr\_in\_function\*aae\_in\_function+c7\*ln\_tr\_in\_function\*aae\_in\_function\*SSA\_in\_function

//CurveFitDialog/ End of Equation

//CurveFitDialog/ Independent Variables 3

//CurveFitDialog/ SSA\_in\_function

//CurveFitDialog/ aae\_in\_function

//CurveFitDialog/ ln\_tr\_in\_function

//CurveFitDialog/ Coefficients 8

//CurveFitDialog/ w[0] = c0

//CurveFitDialog/ w[1] = c1

//CurveFitDialog/ w[2] = c2

//CurveFitDialog/ w[3] = c3

//CurveFitDialog/ w[4] = c4

//CurveFitDialog/ w[5] = c5

//CurveFitDialog/ w[6] = C6

//CurveFitDialog/ w[7] = C7

return w[0]+w[1]\*ln\_tr\_in\_function+w[2]\*SSA\_in\_function+w[3]\*aae\_in\_function+w[4]\*ln\_tr\_in\_function\*SSA\_in\_function+w[5]\*SSA\_in\_function\*aae\_in\_function+w[6]\*ln\_tr\_in\_function\*aae\_in\_function+W[7]\*ln\_tr\_in\_function\*aae\_in\_function\*SSA\_in\_function

End

Function Decimal\_control(value, numSigDigits)

variable value, numSigDigits

string str

sprintf str, "%.\*g\r", numSigDigits, value

return str2num(str)

End