```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Preliminary Look-up Table Generation based on Scattering Theory     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Baseline directory
base = '';      %Load your phantom data
wv = array of wavelengths;
load([base])

%%FITTING OPTIONS
%Forward model
fitOpt.objFunc = @SFD;

%Wavelengths
fitOpt.wvbd = []; %Range of wavelengths e.g.: 400:715nm
wvbd_idx = find(wavelengths>=fitOpt.wvbd(1)):find(wavelengths>=fitOpt.wvbd(2));
%Chromophores
CHROM_fit = [];
COEFF= []'; % extinction spectra measured or from literature, /m/M
%Optimization parameters
fitOpt.options= optimset('MaxIter', 5000000, 'LargeScale', 'off', 'Display', 'off',...
'FunValCheck','off','TolFun', 1e-10, 'TolX', 1.e-8,'JacobMult','on');

%%CALIBRATION
% Scattering coefficient extraction
y0 = 8.261E+1;
a = -1.288E-1;
b = 6.093E-5;
muSp20IL = y0+a*wv(:)+b*wv(:).^2;
muSp20IL = muSp20IL(:);
% Dilute to input percent
r_musp =
(squeeze(repmat(muSp20IL,1,length(#_of_phantoms))).*squeeze(repmat(#_of_phantoms,length(wv(:)),1)
*10^-2)./0.2)*1000;

% Scattering Simulation Mesh

function M1 = M_sim_Mesh(na_particle, na_medium, cosTheta, wv, diameters)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Look-up table mesh of certain particles diameters, wavelengths & phase functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M1.na_particle = na_particle;         %Refractive index of particle
M1.na_medium = na_medium;             %Refractive index of medium
M1.cosTheta = cosTheta;               %Average cosine of backscattered angle
M1.wv = wv(:);                        %Array of wavelengths
M1.diameters = dm;                    %Array of diameters
m = na_particle/n_medium = 0.0i;
M1.diameters = dm;
for ii = 1:length(wv)
    for jj = 1: length(dm)
        wvi = wv(ii);
        di = dm(jj);
        x = 2*pi*di/(wvi/na_medium);  %Size dimensional
        A = (pi*di^2)/4;              %Cross-sectional area of particle

function M2 = M(m, x);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Scattering efficiencies
%input:
% m = refractive index
% x = wave number as a product of the particle radius
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if x==0
M2 = [real(m) imag(m) 0 0 0 0 0 0 1];
elseif x>0
if x >=0.02 && x <=8
namax = round(2+x+4*x^(1/3));
elseif x > 8 && x < 4200
namax = round(x+4.05*x^(1/3)+2);
else
```

```
namax = round(x+4*x^(1/3)+2);
end
n1 = namax-1;
n = (1:namax);
cn = 2*n+1;
c1n = n.*(n+2)./(n+1);
c2n = cn./n./(n+1);
x2 = x*x;

function M3 = a_d(m, x);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Scattering series
% input:
% m = refractive index
% x = wave number as a product of the particle radius
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if x >=0.02 && x <=8
namax = round(2+x+4*x^(1/3));
elseif x > 8 && x < 4200
namax = round(x+4.05*x^(1/3)+2);
else
namax = round(x+4*x^(1/3)+2);
end
n =(1:namax);
nu = (n+0.5);
z = m.*x;
m2 = m.*m;
sqx = sqrt(0.5*pi./x); sqz= sqrt(0.5*pi./z);
bx = besselj(nu, x).*sqx;
bz = besselj(nu, z).*sqz;
yx = bessely(nu, x).*sqx;
hx = bx+i*yx;
b1x = [sin(x)/x, bx(1:namax-1)];
b1z = [sin(z)/z, bz(1:namax-1)];
y1x = [-cos(x)/x, yx(1:namax-1)];
h1x = b1x+i*y1x;
ax = x.*b1x-n.*bx;
az = z.*b1z-n.*bz;
ahx = x.*h1x-n.*hx;
an = (m2.*bz.*ax-bx.*az)./(m2.*bz.*ahx-hx.*az);
bn = (bz.*ax-bx.*az)./(bz.*ahx-hx.*az);
cn = (bx.*ahx-hx.*ax)./(bz.*ahx-hx.*az);
dn = m.*(bx.*ahx-hx.*ax)./(m2.*bz.*ahx-hx.*az);
M3 = [an; bn; cn; dn];
end; end

M3 = a_d(m,x);
anp =(real(f(1,:)));
anpp =(imag(f(1,:)));
bnp =(real(f(2,:)));
bnpp =(imag(f(2,:)));
g1(1:4,namax) = [0; 0; 0; 0];
g1(1,1:n1) = anp(2:namax);
g1(2,1:n1) = anpp(2:namax);
g1(3,1:n1) = bnp(2:namax);
g1(4,1:n1) = bnpp(2:namax);
dn = cn.*(anp+bnp);
q = sum(dn);
qext = 2*q/x2;
en = cn.*(anp.*anp+anpp.*anpp+bnp.*bnp+bnpp.*bnpp);
q = sum(en);
qsca = 2*q/x2;
qabs = qext-qsca;
fn = (f(1,:)-f(2,:)).*cn;
gn = (-1).^n;
M3(3,:) = fn.*gn;
q = sum(f(3,:));
qb = q*q'/x2;
asy1 = c1n.*(anp.*g1(1,:)+anpp.*g1(2,:)+bnp.*g1(3,:)+bnpp.*g1(4,:));
asy2 = c2n.*(anp.*bnp+anpp.*bnpp);
asy = 4/x2*sum(asy1+asy2)/qsca;
```

```
qratio = qb/qsca;
M2 = [real(m) imag(m) x qext qsca qabs qb asy qratio];
end;
        M1.Qs(ii, jj) = M2(5);                 %Forward Efficiency
        M1.Qb(ii, jj) = M2(7);                 %Backward Efficiency
        M1.g(ii, jj) = M2(8);                  %Anisotropy

function M4 =  S1_2(m, x, cosTheta)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Functions based on complex refraction
%input:
% m = refractive index
% x = wave number as a product of the particle radius
% cosTheta = average backscattering angle
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if x >=0.02 && x <=8
nmax=round(2+x+4*x^(1/3));
elseif x > 8 && x < 4200
nmax=round(x+4.05*x^(1/3)+2);
else
nmax = round(x+4*x^(1/3)+2);
end
M3 = a_d(m,x);
an = M3(1,:);
bn = M3(2,:);

function M5 = pt_(cosTheta, namax)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Angular components
%input:
%cosTheta = average backscattering angle
%namax = max angle
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

p(1) = 1;
t(1) = cosTheta;
p(2) = 3*cosTheta;
t(2) = 3*cos(2*acos(cosTheta));
for n1 = 3:namax
    p1 = (2*n1-1)./(n1-1).*p(n1-1).*cosTheta;
    p2 = n1./(n1-1).*p(n1-2);
    p(n1) = p1-p2;
    t1 = n1*u.*p(n1);
    t2 = (n1+1).*p(n1-1);
    t(n1) = t1-t2;
end
M5 = [p; t];
end
end

M5 = pt_(cosTheta, namax);
pin = pt(1,:);
tin = pt(2,:);
n = 1:namax;
n2 = (2*n+1)./(n.*(n+1));
pin = n2.*pin;
tin = n2.*tin;
S1 = (an*pin'+bn*tin');
S2 = (an*tin'+bn*pin');
M4 = [S1_; S2_];
        M1.S1_(ii,jj) = M4(1);
        M1.S2_(ii,jj) = M4(2);
end
end
end

function [WV, MUS_P, MUSB_P11, MUSB_PHG, G_TOT] =
gen_(dmin, d_max, d_mean, d_num, n_med, cosTheta, M1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Simulated particle model
%input
```

```
%min diameter
%max diameter
%mean diameter
%range of particle diameters from min to max
%refractive index
%average backscattered angle
%Matrix results of scattering simulations ‘Sim.res’
%output
%wavelength
%scattering coefficient
%scattering coefficient at cross-section
%scattering coefficient based on Henyey-Greentein phase function
%diffusion tensor/anisotropy result
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Original values
dd_orig = M1.dd; %new array of diameters
Qb_orig = M1.Qb;
Qs_orig = M1.Qs;
S1_orig = M1.S1;
S2_orig = M1.S2;
g_orig = M1.g;
wv_orig = M1.wv;

%Update size
Nd = d_num;    %discretization
for dmaxi = 1:length(d_max)
for dmeani = 1:length(d_mean)
dmax = d_max(dmaxi); %nucleii’s diameter

%Originals
dd = dd_orig;
Qb = Qb_orig;
Qs = Qs_orig;
S1 = S1_orig;
S2 = S2_orig;
g = g_orig;
wv = wv_orig;
ind = find(dd >= dmin):find(dd >= dmax);

%Matrices update
dd = dd(ind); % new array of diameters
Qb = Qb(:,ind);
Qs = Qs(:,ind);
 S1 = S1(:,ind);
 S2 = S2(:,ind);
 g = g(:,ind);

 %Size parameters in nm
 dbin = gradient(dd);
 dbin = repmat(dbin,1,length(wv))';
 d = repmat(dd,1,length(wv))';
 w = repmat(wv,1,length(dd));
 x = 2*pi*(d/2)./(w/n_med); %size of particle, dimensionless
 k = 2*pi./w;              %wavenumber, nm^-1
 A = pi*(d.^2)/4;          %area of particle with diameter d, in nm^2
 V = pi*(d.^3)/6;          %particle volume, in nm^3

 %Optical parameters
 Cs = Qs.*A;               % nm^2

%Differential cross-section at unpolarized cos<theta>
dCs = (w.^2)/(8*pi^2).*(abs(S1).^2+abs(S2).^2);
Cb = Qb.*A;                % back-scattering cross-section

%Intensity component of phase matrix
P11 = ((4*pi)./(k.*k.*Cs)).*(0.5*abs(S1).*abs(S1)+0.5*abs(S2).*abs(S2));
PHG = (1/4*pi).*(1-g.*g)./((1+g.*g-2*g*u).^3/2);

%Exponential distribution
d_mu = d_mean(dmeani);                             %in nm
```

```
eta0 = (1/d_mu)*exp(-d/d_mu);                      %exponential distribution
A = 1./sum(eta0.*dbin,2); A = A(1)/size(w,1); %normalization constant
eta = A*eta0;
%Volume fraction
Vf = 0.01;                                       %tissue volume fraction

%Normalize to convert eta into effective particle density and to match the
Vnorm = sum(eta.*V.*dbin,2)/Vf; Vnorm = Vnorm(1); %normalization, in nm^3
densityperbin = eta/Vnorm;                         %density type per bin

%Correcting for correlated scattering
eta_eff = densityperbin.*((1-densityperbin).^4)./((1+2*densityperbin).^2);

%Scattering coefficient
musi = (eta_eff.*dbin).*Cs; %in nm
muspi = (eta_eff.*dbin).*Cs.*(1-g);

%Backscattering coefficient
musbiP11 = (eta_eff.*dbin).*Cs.*P11; %in nm

%Optical parameters of particle distribution in mm
clear mus musbP11 musp
mus = sum(musi,2)*(10^7);
musp = sum(muspi,2)*(10^7);
musbP11 = sum(musbiP11,2)*(10^7);
gtot = 1 - musp./mus;           %Diffusion tensor or anisotropy

% Henyey-Greenstein phase function
PHG = (1/4*pi).*(1-gtot.*gtot)./((1+gtot.*gtot-2*gtot*u).^3/2);
PHG = repmat(PHG,1,size(Cs,2));

%Calculate backscattering based on diffusion tensor
musbiPHG = (eta_eff.*dbin).*Cs.*PHG;
musbPHG = sum(musbiPHG,2)*(10^7); % per mm

%Solution arrays
WV = w(:,1);
MUS_P(dmaxi,dmeani,:) = musp;
MUSB_P11(dmaxi,dmeani,:) = musbP11;
MUSB_PHG(dmaxi,dmeani,:) = musbPHG;
G_TOT(dmaxi,dmeani,:) = gtot;
Sim.res = [WV, MUS_P, S1, S2, Qs, Qb, MUSB_P11, MUSB_PHG, G_TOT];
save([dir 'Sim.mat'], 'Sim.res')
end; end

r_mua = zeros(size(r_musp));

%Calibration
aREF = repmat(fitOpt.objFunc(r_musp(:),r_mua(:)),[1 size(r_musp,1) size(r_musp,2)]);
aREF = permute(aREF,[2 3 1]);
PH.abs_data = (Literature_reference).*(aREF);

%PHANTOM INFO
figure();
n1 = ;                  %# of samples of phantom 1
n1 = ;                  %# of samples of phantom 2
for iHbO2 = 1:n1        %varying hemoglobin concentrations
for iIL = 1:n2          %varying Intralipid concentrations
X = squeeze(wv(wvbd_idx));
Y = squeeze(PH.abs_data(iHBO2,iIL,wvbd_idx));


%GUESSWORK
%Optimize initial guess (g1, g2, g3) for A,b, using three points
g1 = ; %guess 1
g2 = ; %guess 2
g3 = ; %guess 3
x1 = log(wv(wvbd_idx)>=g1);
x2 = log(wv(wvbd_idx)>=g2);
x3 = log(wv(wvbd_idx)>=g3);
y1 = log(abs(Y(wv(wvbd_idx)>=g1)));
```

```
y2 = log(abs(Y(wv(wvbd_idx)>=g2)));
y3 = log(abs(Y(wv(wvbd_idx)>=g3)));

%Initial guess for slope of the scatterers
m = (2*y3-y2-y1)/(2*x3-x2-x1);
fitOpt.betaKey = {'A', 'b', 'HbO2', 'Hb'};
fitOpt.betaMin = [ ];
fitOpt.betaMax = [ ];
fitOpt.betaInt = [ ];

%FIT
[IMG(iHbO2,iIL,:),YHAT(iHbO2,iIL,:),RSQ(iHbO2,iIL)] = nonlinOptim(X,Y,COEFF,fitOpt);
mas_musp(iHbO2,iIL,:) = IMG(iHbO2,iIL,1)*wv(wvbd_idx).^-IMG(iHbO2,iIL,2);
end
end


%% RUN SEPARATE SIMULATIONS
load('Sim');  %Simulation results for particle size distribution
muspTABLE = M1.musp;
clear musp;
nph = ;         %#of liquid phantoms
A_ind = find(strcmp(fitOpt.betaKey,'A'));
b_ind = find(strcmp(fitOpt.betaKey,'b'));
for iHbO2 = 1:n1
for nph = 1:n2
A = IMG(iHbO2, nph, A_ind);
b = IMG(iHbO2, nph, b_ind);

function [r_avg, N] = minMUSP(A,b,M1,wv,r_musp)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mapping of simulations to experimental measurement using minimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for ri = 1:length(r_mu)
musp_M1 = M1(ri,:);
musp_exp = A*wv.^-b;
wv0 = find(wv >= 685);
musp_exp_norm = musp_exp/musp_exp(wv0);
musp_M1_norm = musp_M1/musp_M1(wv0);

%Least squares
sum = 0;
for wi = 1:length(wv)
sum = sum + ((musp_exp_norm(wi) - musp_M1_norm(wi))/musp_M1_norm(wi)).^2;
end
chi(ri) = sqrt((1/length(wv))*sum);
end

%Minimization
[C,min_index] = min(chi);
r_avg = r_mu(min_index);
N = musp_exp(wv0)/M1(min_index,wv0); % relative to reference N0 (5% IL)
end %

[r_avg(iHbO2, nph),N(iHbO2, nph)] = minMUSP(A,b,M1,wv,ru_musp);
end; end
```