```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reflectance as a function of spatial frequency according to Diffusion approximation
% Inputs:
% [mua, musp] - Nx2 absorption and scattering parameters, dimensionally 1/mm
% f = spatial frequencies vector
% n = refractive index
% % Output:
% y_at_fx = Reflectance relative to spatial frequency at particular optical properties
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [y_at_fx] = SFD([mua, musp],f,n)
nfreq = length(f);
nwv = length(mua(:,1));
y_at_fx = zeros(length(f),length(mua(:,1)));
mutr=mua+musp;% reduced att. coefficient
c=300;         % light velocity (mm/ns)
v=c/n;
Reff=-1.440./n.^2+0.710./n+0.668+0.0636.*n;
ETA=(1-Reff)./2./(1+Reff);
mua = repmat(mua',nfreq,1);
musp = repmat(musp',nfreq,1);
mutr = repmat(mutr',nfreq,1);
fx = repmat(f',1,nwv);
mueff = (3*mua.*mutr+(2*pi*fx).^2).^0.5;
y_at_fx =
(3*ETA*musp./mutr)./((ones(size(mueff))+mueff./mutr).*(3*ETA*ones(size(mueff))+mueff./mut

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Nonlinear Version of Diffusion model
% INPUT:
% fx = frequencies
% wv = wavelengths
% y_at_fx = reflectance data
% n = refractive index
% c_  = data structure
%
% OUTPUT:
% yhat = fit data
% beta = fit parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [yhat,beta] = minSFD(fx,y_at_fx,n,c_)
% Coefficients
nfreq = length(fx);
Reff=-1.440./n.^2+0.710./n+0.668+0.0636.*n;
ETA=(1-Reff)./2./(1+Reff);
%Fitting Options
sc = 25; %Rescale
fitOpt.options = optimset('TolFun',0.000001,'TolX',0.000001,'display','off');
fitOpt.betaInt = [c_.mua_guess*sc c_.musp_guess];    %musp_guess should be 100x that of mua_guess
fitOpt.betaMin = [zeros(size(fitOpt.betaInt))];
fitOpt.betaMax = [0.333*sc 2.5];

%%%%%%%%%% OPTIMIZATION %%%%%%%%%%
% Least Squares Fitting
beta =
lsqcurvefit(@Nest,fitOpt.betaInt,fx,y_at_fx,fitOpt.betaMin,fitOpt.betaMax,fitOpt.optio
ns);
yhat = Nest(beta,fx);
beta(1) = beta(1)/sc;


 %%%%%%%%%% OTHER FUNCTION %%%%%%%%%%
 function yhat = Nest(beta,fx)
 %Initializers
 mua = beta(1)/sc;
 musp = beta(2);
 f = fx;
 mutr=mua+musp;
 mueff = (3*mua.*mutr+(2*pi*f).^2).^0.5;
 yhat =
(3*ETA*musp./mutr)./((ones(size(mueff))+mueff./mutr).*(3*ETA*ones(size(mueff))+mueff./mut
r));
 yhat = yhat';
 end %

 function resid = MinNest(beta)

 %Initializers
 mua = beta(1)/sc;
 musp = beta(2);
 f = fx;
 mutr=mua+musp;

 %Vectorize
 mueff = (3*mua.*mutr+(2*pi*f).^2).^0.5;
 yhat =
(3*ETA*musp./mutr)./((ones(size(mueff))+mueff./mutr).*(3*ETA*ones(size(mueff))+mueff./mut
r));
 yhat = yhat';
 resid=sum(sum((yhat-y).^2));
 end
end
```