```
function [beta,yhat,Res] = NLMIN(x,y,C,fitS)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Nonlinear minimization
% INPUT:
% x - wavelength
% y - reflectance
% C - extinction coefficient of absorption spectra [/m/M], base10
% dimension (# chromophores, length(x))
% fitS is a data structure with the following fields:
% .objFunc - handle to function of form R(mua,musp)
% .options - optimset parameters
% .betaKey - fit parameters
% .betaMin - min bound
% .betaMax - max bound
% .betaInt - initial guesses
%
% OUTPUT:
% beta - parameters corresponding to fitS.betaKey
% yhat - modeled reflectance
% Res - goodness of fit 1-RSS/TSS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% FORWARD MODEL
objFunc = fitS.objFunc;

% SPECTRAL PARAMETERS
wv = x(:); % wavelength
HbO2 = C(1,:); % extinction spectra of oxygenated hemoglobin
Hb = C(2,:); % extinction spectra of deoxygenated hemoglobin
if(size(C,1) >2)
N = C(3,:); % additional chromophore extinction spectra
end

% OPTIMIZATION
[beta,resnorm,resid,exitflag,output,lambda,J]=lsqnonlin(@Nest,fitS.betaInt,fitOp
t.betaMin, fitS.betaMax, fitS.options);
yhat = Nest(beta,x);


% R-squared
Res = 1-(sum((yhat(:)-y(:)).^2)/sum((y(:)-mean(y(:))).^2));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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
fitS.options = optimset('TolFun',0.000001,'TolX',0.000001,'display','off');
fitS.betaInt = [c_.mua_guess*sc c_.musp_guess];    %musp_guess should be 100x than mua_guess
fitS.betaMin = [zeros(size(fitS.betaInt))];
fitS.betaMax = [0.333*sc 2.5];


%%%%%%%%%% OPTIMIZATION %%%%%%%%%%
% Least Squares Fitting
Beta = lsqcurvefit(@Nest,fitS.betaInt,fx,y_at_fx,fitS.betaMin,fitS.betaMax,fitS.optio ns);


%%%%% NESTED FUNCTION %%%%%%
function yhat = Nest(Beta,x)

% Optical parameters
musp = beta(1)*(wv).^-beta(2);
if(size(C,1) >2)
mua = log(10)*((beta(3))*(beta(4)*HbO2+(1-beta(4))*Hb)+beta(5)*N);
mua = abs(mua(:));
else
mua = log(10)*((beta(3))*(beta(4)*HbO2+(1-beta(4))*Hb));
mua = abs(mua(:));
end
else

% Optical Properties
wv = x(:,1);
wv = wv(:);
musp = beta(:,1);
mua = beta(:,2);
end

% Evaluate objective function
yhat = objFunc(musp,mua);
resid = yhat(:)-y(:);
end
```