A tutorial and pseudo-code on how to process SFDI data

Last update: 05.09.19

I. Introduction

This tutorial is designed to guide you through the basic stages of processing SFDI data, providing some pseudo-code examples in Matlab as to how these stages may be executed. SFDI may be implemented in many different ways, each having their own specific contingencies, caveats and/or additional data processing requirements. This tutorial will attempt to cover the most general case for SFDI systems: Images collected sequentially over multiple wavelengths at multiple spatial frequencies. Here, we only consider data sets where each spatial frequency is represented by a sinusoidal intensity pattern along one spatial dimension and projected at three evenly spaced phases (0, 120 and 240 degrees).

The data processing steps outlined here largely follow what is described in Cuccia, et.al JBO 2009 ("Quantitation and mapping of tissue optical properties using modulated imaging"). The steps and functions used in the pseudo code sections are not optimized in any way, but rather written in a way to illustrate the process in a simple, explicit way. While these functions should run correctly, it is strongly encouraged that you modify these codes to run it more efficiently.

This tutorial will continue to evolve and add additional topics and methods in the near future. *Since the last posting (24th June, 2019)*, a tutorial on how to implement a white Monte Carlo model has been added and both Diffusion Approximation and Monte Carlo models have been moved to its own section (III.), preceding the Calibration section. Additionally the Debugging and Diagnostics sections for Demodulation (II), Calibration (IV.) and Inverse Solver (V.) have been expanded.

If there are any questions regarding this tutorial or issues with attempting to implement the pseudo code functions, please feel free to contact me: rolf.saager@liu.se.

I am open to any additional questions or topics that can be added to this tutorial in the future, if there is interest (e.g. alternative inverse solver methods, such as Look-Up tables, or other emerging models for light transport)

II. Demodulation

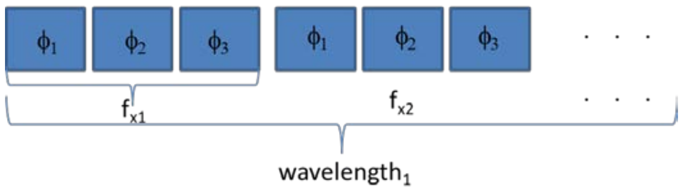
a. Description

SFDI is all about measuring how patterns of light decay and attenuate after interacting with turbid media like tissue. As the spatial frequency content of a pattern will interact differently between absorption and scattering, the shape of this decay as the spatial frequency increases will be unique for absorption and scattering values typically encountered in tissue. When patterns of sinusoidal light intensity are considered, this decay is represented by the AC amplitude of the sinusoid measured from tissue at multiple spatial frequencies.

There have been many approaches in SFDI systems to extract this AC using single snapshot, multi-frequency synthesis and others (most of these can be found in the reference literature list under the SFDI 101 section), but this tutorial will focus on the "classical" 3-phase demodulation approach.

b. Matlab pseudo code:

To start, let's assume that your data is stored as a series of images in the following hierarchy 1) Wavelength 2) Spatial frequency and 3) phase, namely what is shown below:



The approach we have taken is to set up a 3 level loop (loop over wavelengths, frequencies and phases), while this is not computationally efficient; it is simple and a bit easier to visualize:

```
for k=1:length(w) %loop over wavelengths (user defined or read in from meas. param file)
    for i=1:length(f) %loop over spatial frequencies (user defined or read in)
        for j=1:3 %loop over phases (user defined)
            % In this loop, we load the data, 3 images at a time (namely
            % for a given wavelength and frequency, we load the 0, 120, 240
            % phase images). we use a temp variable as these 3 phases will
            % be demodulated to extract the AC images one at a time. Key to
            % running this automatically is to have a consistant naming
            % scheme to identify wavelength, frequency and phase.  For our
            % group we had a simple numbering scheme and used the loop
            % index to id the wv, freq, and phase (for example the file:
            % image_2_3_1.png would be the image from the 2nd wavelength at
            % the 3 spatial frequency with a phase of 0 degrees).  PathName
            % is the folder location where the data is stored.
            % [FileName,PathName] = uigetfile is a popup window function
            % that can allow you to click your way to the correct folder,
            % rather than writing it all out manually.
            temp(:,:,j) = ...
               imread([PathName 'image_' num2str(k) '_' num2str(i) '_' num2str(j) '.png']);

        end
        %3-phase demodulation scheme: this implements eqn 20 from cuccia.
        %Note that this is now a 4-D array where the first 2 dimensions are
        %the x and y pixels, the 3rd is the wavelength and the 4th spatial
        %frequency.  We also divide each demodulated image by a variable
        %int_time(k). This is the exposure time used at each wavelength.
        %So in essence, the resulting demodulated image is an intensity per
        %sec and hence normalized the detected signals across all
        %wavelengths even though the integration times might vary
        AC(:,:,k,i)=sqrt(2)/3*sqrt( (temp(:,:,1)-temp(:,:,2)).^2 + ...
            (temp(:,:,2)-temp(:,:,3)).^2 + ...
            (temp(:,:,3)-temp(:,:,1)).^2 )./int_time(k);
    end

end
```

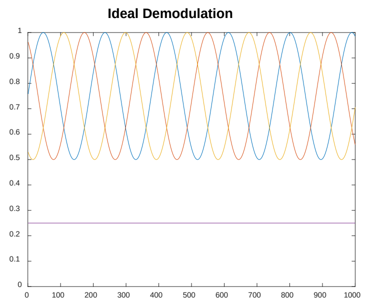
c. Debugging and Diagnostics

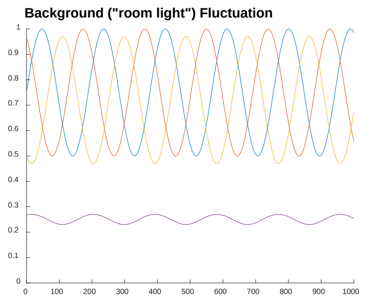
Although this is the first and most basic of processing steps for SFDI, there are a number of potential measurement issues that can be identified at this early stage. Given how computationally intensive the full data processing procedure is, it is often very useful to be able to identify any data integrity issues and also be able to determine where these issues may come from.

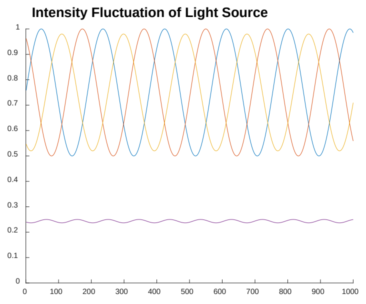
Successful demodulation should result in raw images of the tissue without any indication that a series of sinusoidal patterns were used to illuminate it. Residual modulation artifacts can remain in any of the images for a number of reasons. Three of the most common are

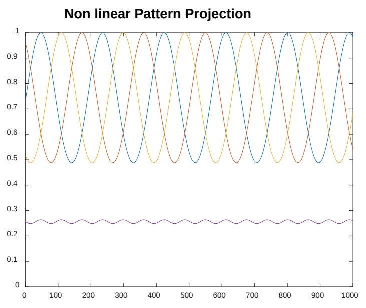
1. Changes in the ambient (room) light during the acquisition sequence
    i. While one distinct advantage of using the 3-phase demodulation approach is that it allows SFDI to be acquired under normal lighting conditions, so long as those room lights are constant over the 3 phases (acting as a constant DC offset). If the lighting condition changes (additional light turned on or off, someone or something moves over the measurement area, casting a shadow)
2. Fluctuations in the light source intensity
    i. As the 3 phases are imaged sequentially, it is critical for the output of that source be stable over that time. This typically can arise with pulsed sources where shot to shot variations of the source could occur or the synchronization between illumination and camera become unstable.
3. Nonlinear response to the projector output intensity
    i. This is a common issue with repurposed commercial projectors or running an SFDI system in video mode. Commercial projectors and video rendering are often optimized for the non-linear response of our vision and hence tend to apply a Gamma correction to images/videos input to it. This results in a distorted intensity pattern being projected. In this case, the patterns are no longer sinusoid and hence do not demodulate properly.

While there are many potential pitfalls to acquiring a clean SDFI dataset, each one of the issues identified above will not only show residual modulation, each one is distinct in the frequency of the modulation relative to the spatial frequency of the source (see the simple rendering below)





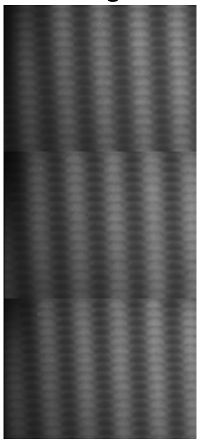




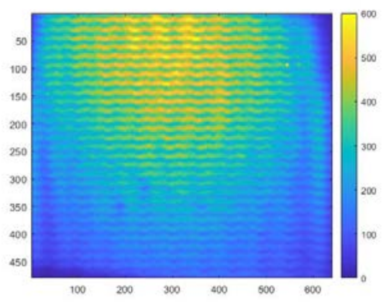
<u>Synchronization between projector and camera</u>: Another source of error may also arise when the camera acquisition time is on the same order as (or faster than) the frame rate of the projector. With digital mirco-mirror devices, DMDs, it is important to remember that the various levels of light intensity are produced by virtue of how much time a given mirror (pixel) is in the "on" vs "off" position. For example, there are 256 intensity levels in an 8-bit image and hence to render that level of detail, the projector will require (256)x(the minimum mirror switching speed). In practice DMDs generate patterns of increasing bit-depth through the sequential reconstruction of an image through increasing (or decreasing) bit-planes. Here, the resulting

imaging is the temporal sum of each lower bit-depth component. Should the camera acquisition time be less than the time required for this image reconstruction, the pattern captured by the camera will be incomplete. Additionally, if the camera acquisition is not an even multiple of the projector frame rate, the last remaining incomplete frame could also bias the pattern detected and potentially corrupt the demodulated image. An example of this is provided below:

Raw images



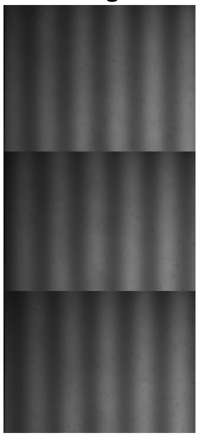
Demoduated Image



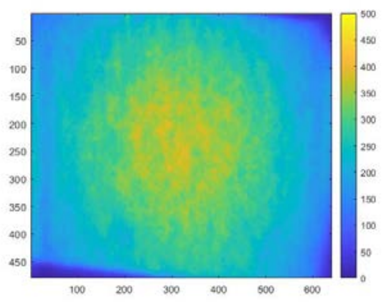
(@0.1 mm^{-1})

Exposure time = 4.322 ms (~1/4th Projection time)
Projection time = 16.66 ms

Raw images



Demodulated Image



(@ 0.1 mm^{-1})

Exposure time = 133.467 ms (8x Projection time)
Projection time = 16.66 ms

*Example data provided by Anahita Pilvar, Boston University (BOTLAB)

This issue can be managed by constraining camera exposure times to be even multiples of the projector frame rate. In this case even if the projector and camera are running asynchronously, each image acquisition will contain all bit-planes (complete projection images) even if the image acquisition starts in the middle of a given projection. Another way to address this is to ensure that the camera exposure time >> projection time. Here the assumption is that any image collected by the camera is the integrated sum of so many projected patterns that the contribution of a single partial frame will remain below the signal noise itself and hence pose negligible impact on the detected signal.

<u>Is this projector frame rate a fundamental limitation to the acquisition speed?</u> Not necessarily… One trick to "increase" the speed of the projector is to reduce the number of bits used to project the image (if you can access the bit-depth of the projector). Reducing the projections to 6-bits (64 levels of intensity), can increase the frame rate by a factor of 4, though given its non-standard bit depth, there are some additional tricks involved in generating (saving) truly 6-bit image format. Also it is worth noting that some groups have developed "very" fast acquisition methods by running projectors at 1-bit. Here the patterns are square waves and not sinusoidal and therefore make the interpretation and data processing in terms of SFDI a bit more complicated (not discussed here).

III. Models for Light Transport in the Spatial Frequency Domain

a. Description

As SFDI attempts to extract the absorption and reduced scattering coefficient values from *within volumes* of tissue, there is a need for a model to describe the light's complex interaction and behavior that can relate the light that is detected (i.e. Reflectance) from it in terms of the light that is delivered to the tissue. While light – tissue interaction (in terms of reflectance) can generally be described by the Radiative Transport Equation (RTE), this equation is difficult to work with in either a forward (calibration stage) or inverse (fitting for optical properties) solver context. So, without going into too much detail here about the RTE, we will focus on two common methods employed to model light transport as an analytical solution (Diffusion Approximation Model) or as a stochastic solution (Monte Carlo Model).

In this tutorial, I will briefly describe the two methods and provide pseudo-code for implementing each. These methods will be set up to match each other in terms of inputs and outputs so they can be seamlessly inserted for each other in the rest of the processing code outlined here.

As there will not be any detailed derivation or discussion behind the mechanics of these models, it is strongly recommended that individuals seek out other resources, workshops, and/or tutorials that explore these fundamental principles in more detail. There are many resources available, if you are curious to learn more about how to work with the RTE and how solutions like the diffusion approximation or Monte Carlo can be implemented. I recommend https://virtualphotonics.org, where you can not only find online lectures from previous workshops on the topics, but also online tools and resources to run these methods for yourself.

b. Diffusion Approximation model

Applying the "diffusion approximation" to the RTE greatly reduces the complexity of the equation such that it can be simplified to a first order differential equation and solved analytically. Skipping over all the interesting details and nuance, one significant principle behind this approach is to simplify the directionality of light scattering into two reduced forms: an isotropic scattering component and a forward directed component.

This simplification, however, requires many scattering events to occur (and hence also few absorption events) from the detect light to remain valid. While this is an approximation, it is *typically reasonable* to use in SFDI if you are 1) in the NIR (scattering>>absorption), 2) Use spatial frequencies between 0-0.25/mm (probing volumes of tissue > 1/(reduced scattering)) and 3) measuring homogenous tissue.

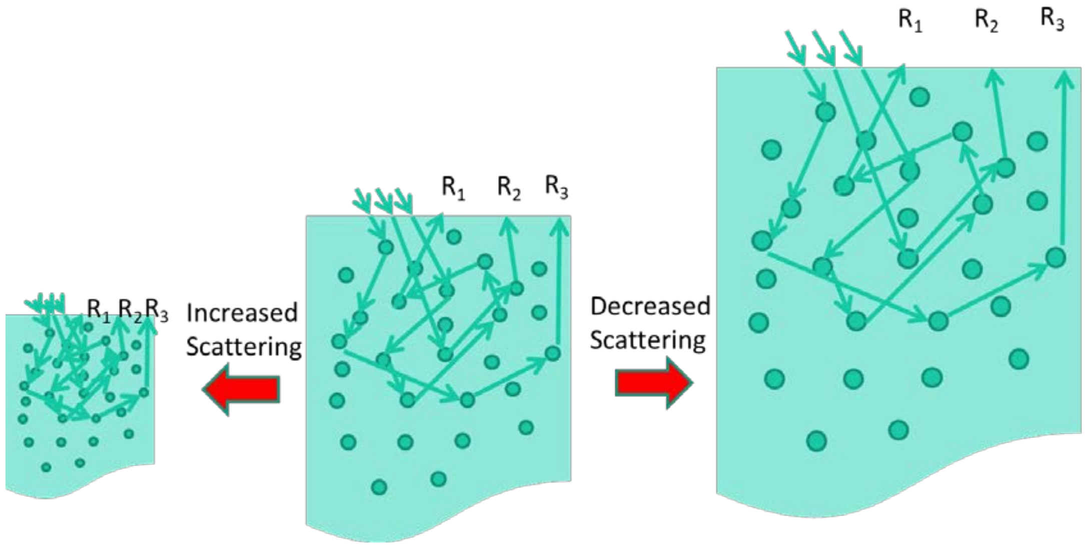
```
function  reflectance_fx= diff_model_for_SFDI(op,n, fx)
%diffusion model for light transport in tissue (taken from Cuccia 2009)
% op -- is a 2-element array containing [mua musp] (in units of mm^-1)
%          mua   -- the input absorption values at every wavelength (array)
%          musp  -- the input reduced scattering values at every wavelength
% n  -- the index of refraction of the sample measured
% fx -- the spatial frequencies used in the measurement This function will
%   then generate a matrix reflectance_fx(wv,fx) that represents the "true"
%   reflectances at each wavelength and spatial frequency

mua=op(:,1);
musp=op(:,2);
mutr=mua+musp; % reduced attenuation coefficient (a.k.a. "mu transport")
Reff=-1.440./n.^2+0.710./n+0.668+0.0636.*n; %surface loss (assumes air interface) eqn(8)
A=(1-Reff)./(2.*(1+Reff)); %eqn(8)

for w = 1:length(mua) %loop over wavelengths
    for f= 1:length(fx) %loop over spatial frequencies
        mueff_prime = (3*mua(w)*mutr(w)+(2*pi*fx(f))^2)^(0.5); % reduced "mu effective"
                                                                % eqn(5)
        reflectance_fx(w,f) = ...
    3*A(w).*(musp(w)/mutr(w))/(mueff_prime/mutr(w)+1)/(mueff_prime/mutr(w)+3*A(w));
        % derived from eqn(10)
    end
end
```

c. "Monte Carlo" model

The diffusion approximation allows us to derive an analytical function for tissue reflectance, where the absorption and scattering coefficients are inputs. The Monte Carlo approach simulates incremental light tissue interactions in a random medium defined by its absorption and scattering values, resulting in a statistical distribution of remitted light through stochastic methods. This approach allows for greater accuracy for modeling tissue reflectance when absorption approaches scattering values and when tissue volumes interrogated is on par with the reduced mean free path of light in tissue (1/(reduced scattering)). This advantage, though, comes at the price of computational power and processing time.

When the “Monte Carlo method” is typically evoked in the context of SFDI data processing, it does not refer to running a full MC simulation for every set of optical properties during the data processing, rather it refers to a specific implementation that uses a single, pre-calculated Monte Carlo simulation that is then manipulated to recreate the reflectance values for arbitrary sets of optical properties. This is referred to as a “White Monte Carlo” ([Kienle and Patterson 1996 Phys. Med. Biol. 41 2221], [Swartling et.al JOSA A 20(4) 2003]).

<u>What is a “white” Monte Carlo</u>? This is a simulation that models the distribution of remitted photons (in terms of space and time) relative to a pencil beam source at the surface of the tissue, where there is only scattering present, hence the term “white”. This simulation establishes the distribution of scattered light, given a specific density and anisotropy of scattering objects (and a specific index of refraction of the medium). Here the scattering coefficient can then change simply by changing the density of these objects in the simulation. This effect can be achieved by merely scaling the spatial extent of the resulting output after the simulation is completed, thereby eliminating the need to run multiple simulations (see figure below).



Once the pre-calculated simulation is scaled to match the desired scattering value, the resulting reflectance values from scattering-only simulation can be extracted for each spatial distance relative to the source pencil beam. The time domain information also stored in the simulation output could then be used to rescale the mean photon pathlength, <*l*>, traveled within the new tissue dimensions (<*l*>=<*t*>*c/n*) at each spatial location. The contribution from absorption is then added (weighted) to the simulated reflectance via Beer’s law.

$$e^{-\mu_a(\lambda)<t>c/n}$$

How do you set up this white Monte Carlo, in practice? What do you need? A white Monte Carlo is a precomputed output matrix that tracks the histogram of photons that exit tissue as a function of distance from a pencil beam (R(rho)) and as a function of time (R(t)). As the scaling of this simulation output is intended to cover the range of optical properties one might encounter in tissue, it is critical for it to be run over a “sufficient” spatial and temporal range with an “adequate” number of photons to ensure any reflectance output would be statistically robust.

What the best values and simulation parameters might be are a matter of debate, but previous incarnations have run with rho going out to ~100mm from the source (~0.1 mm intervals) and out to ~100 ns (at variable intervals from ~0.001-5 ns). For simplicity reduced scattering, musp, was set to 1/mm and absorption, mua, set to a negligible value ~1x10^{-6} or smaller (in discrete absorption weighting Monte Carlos, setting mua=0 could lead to exceptionally long run times as photons will only terminate when exiting tissue and for semi-infinite media…. that could take a really, really long time). In addition to the resulting output matrix of Reflectance as a function of distance (rho) and time (t), you will also need to track what the original intervals a distances/times each element of the matrix represents.

> *note for this pseudo code example, we assume a single white MC simulation for a given (assumed) index of refraction, n, (e.g. 1.4 for tissue) and anisotropy, g, (e.g. 0.8 for tissue) should other media be used, a separate white Monte Carlo should be generated (e.g. intralipid phantoms → n=1.33, g=0.7) or small biases will result…

```
function reflectance_fx= MC_model_for_SFDI(op,n,fx)
%white MC model for light transport in tissue (taken from Cuccia 2009)
% -> R_of_rho_and_t -- 2-D Reflectance matrix over distance (rho) and time (t)
% where rows→ time bins, columns→ rho bins
%   Additional parameters needed, saved with MC output in a *.mat file
%    rho       -- Array of source-detector separations (mm)
%    delta_rho -- Array of spacing between rho detector bins (mm)
%    t         -- Array of temporal detector bins (ns)
%    delta_t   -- Array of time intervals for detected bins (ns)

load('MC_simulation.mat')
v=300/n;  %speed through medium in mm/ns, (300mm/ns → in vacuum)

%calculate the Fresnel reflection at the interface of air and tissue
%at normal incidence, theta=0
fresn=((1-n)/(1+n))^2;

% for each pair of optical properties (i.e. each wavelength)
mua=op(:,1);
musp=op(:,2);

% traditional 'repmat' version
for i=1:length(mua) %loop over wavelengths
%First, generate a reflectance curve for a given absorption and reduced scattering
%coefficient in the spatial domain using a white Monte Carlo:
    % Weight for mua by applying a pathlength correction => sum[ R(rho,t)*dt ]
R_of_rho = sum(R_of_rho_and_t/fresn.*repmat(exp(-mua(i)*v*t*(1/musp(i))).* ...
delta_t*(1/musp(i)),length(rho),1), 2)*(1/musp(i))^-3; %reflectance per m^2 per s,
%(1/musp) represents the scaling factor, the numerator is the original value: 1/mm
% the last term, (1/musp)^-3 stems from scaling the simulation over area(^2) and time(^1)
% This is then summed over all t, i.e. steady state


%Then transform this reflectance curve from the spatial domain into the spatial frequency
domain, using a 1-D Hankel Transform of order zero (as outlined in Cuccia 2009, eqns 15 and
16):
f=length(fx); %number of spatial frequencies used in measurement
r=length(rho); % number of spatial points used in Monte Carlo simulation

R_of_fx(i,:) = sum( repmat(R_of rho.',[f,1]) .* ...
besselj(0,repmat(2*pi*fx,[1,r]).*repmat((rho.').(1/musp(i)),[f,1])) .* ...
             repmat(2*pi*(rho.').*(1/musp(i)),[f,1]) .* ...
             repmat((delta_rho.')*(1/musp(i)),[f,1]), 2)';
end
```

d. Debugging and Diagnostics

To test out the models generated here, a versatile online simulation GUI can be found at https://virtualphotonics.org/software . Here you can generate reflectance curves in both time and spatial domains as well as their Fourier counterparts and compare them to the responses you generate here. This online platform also utilized several models including the diffusion and Monte Carlo approaches invoked in the methods above.

IV. Calibration

a. Description

The motivation behind the calibration step is to convert the raw data collected by the instrument into "absolute" units, namely isolate the spectral characteristics of the tissue from any spectral characteristics of the instrument itself. Since SFDI deals with turbid media and images multiple patterns on to tissue, it is best to calibrate the data at every frequency independently and use a reference standard that exhibits both known absorption and scattering.

The calibration calculation is simple enough:
calibrated_reflectance=(tissue measurement/reference_measurement)*"true reference reflectance_model"

the rationale behind this is that every measurement you make is assumed to be the product of your sample reflectance and your instrument response (i.e. tissue_measurement="true tissue reflectance"*instrument_response and reference_measurement="true reference reflectance"*instrument_response) since both tissue and reference measurements used the same spatial frequencies and measured near the same time relative to each other, it is assumed that the instrument response is the same and hence cancel each other out leaving you with:

calibrated reflectance=("true tissue reflectance"/"true reference reflectance")*"true reference reflectance_model"

Here, we can model what the specific spatial frequency dependent reflectance, R(fx), of the reference standard if we know the absorption (mua) and scattering (musp) properties, index of refraction (n) (typically 1.33 for intralipid phantoms, 1.43 for Silicone (PDMS) phantoms) and the spatial frequencies (fx) used in the measurement. This can be executed through different models and methods, such as the diffusion approximation model or Monte Carlo model for light transport in tissue, described in the section above.

So long as we know the absorption, scattering and index of refraction of the reference standard in advance, we can plug these values into the model will then generate the appropriate reflectance values for the reference standard at each spatial frequency. The resulting output is the “true reference model” which in the calibration formula then cancels out the "true reference reflectance" leaving you with just the "true tissue reflectance".

b. Matlab pseudo code:

For the calibration, all you need as inputs are the demodulated tissue (AC_tissue) and reference data (AC_ref), spatial frequencies used (fx), and the absorption (mua), scattering (musp) index of refraction (n) for the reference standard (phantom)

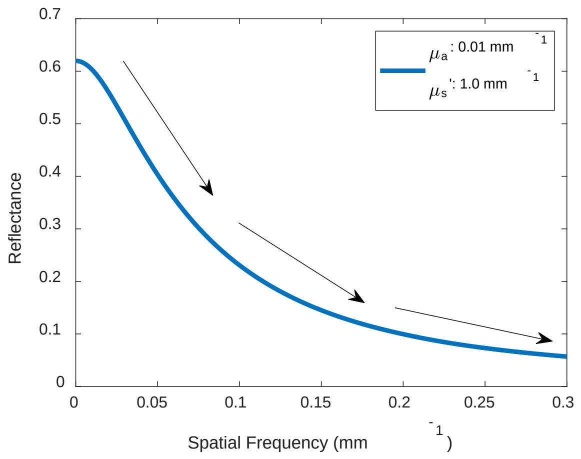
```
%calculate reflectance values of the reference phantom from known absorption and scattering
values
reference_model= diff_model_for_SFDI(op ,n, fx); %diffusion model
%or
%reference_model=MC_model_forSFDI(op,n,fx); %Monte Carlo

%correct for instrument response and reference reflectance to isolate tissue reflectance
for i=1:length(mua) %loop over wavelengths
    for k=1:length(fx) %loop over spatial frequencies

calibrated_reflectance(:,:,i,k)=AC_tissue(:,:,i,k)./AC_ref(:,:,i,k).*reference_model(i,k);
    end
end
```

c. Debugging and Diagnostics

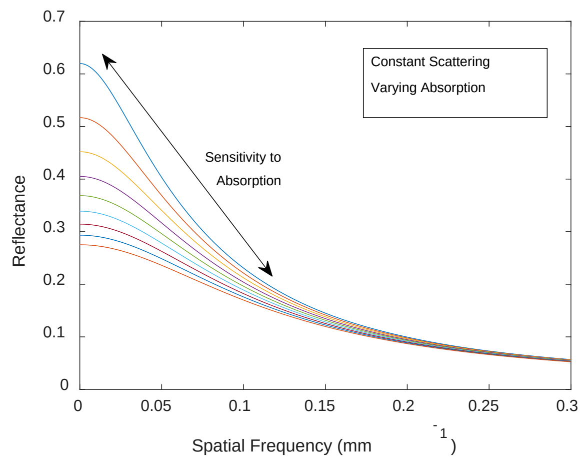
Calibrated reflectance is a bounded quantity: if values are higher than 1 or negative, there is a problem with the demodulation of the raw data (e.g. negative values), the reference values used for the calibration and/or the real vs recorded integration time values (e.g. values higher than 1).

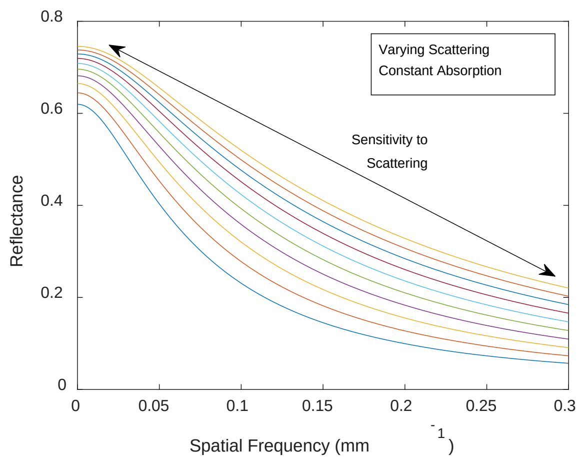
Signal to Noise of demodulated images (visual assessment of noise at higher spatial frequency (fx)): It is important to remember that scattering can greatly reduce the depth of modulation of high spatial frequency patterns. As a result, the demodulated signal at these higher spatial frequencies could represent a small fraction (i.e. 5-10%) of the total signal detected. Essentially, you are discarding 90-95% of your detected signal in post-processing, but still carrying all the noise of that signal with you into the demodulation calculation. For this reason, high spatial frequencies can be very susceptible to measurement (shot) noise and pose significant issues in the model-based fitting for optical properties stage that follows.

Expected behavior of R over increasing spatial frequency (fx): From the various models to represent light tissue interaction, there is a general expectation that the spatial frequency dependent component to (homogeneous) tissue reflectance should monotonically decrease as the spatial frequency increases. The calibrated reflectance should then reflect this behavior. If reflectance values increase, it may be due to loading data in the wrong order or other measurement related errors.



Additionally, it has been noted that the sensitivity to absorption and scattering differ at different spatial frequencies (particularly in the near infrared). Here absorption contrast is greatest at very low spatial frequencies whereas scattering contrast can persist through higher spatial frequencies. When looking at calibrated reflectance images, absorbing objects should "disappear" as spatial frequencies increase, while the scattering objects remain in the images. Have some expectation of what you are measuring can provide some sanity check on the data quality before submitting it to any inverse solving routine.





V. Inverse Solver

a. Description

In this stage, we are not looking to essentially "match" the measured, calibrated reflectance (as a function of spatial frequency) curve to a model of light transport in tissue. In your case, this is the same diffusion approximation model we used in the calibration step. The key differences here is that (1) this model will now be used to mimic the tissue response, where the model was used to mimic the calibration phantom in the previous step and (2) this model will be used in an "inverse solver" configuration rather than the "forward" model case used in the calibration.
A note on terminology: forward model is deterministic-->with known optical properties (mua, musp, n, etc) the reflectance can be determined, in the inverse case, one tries to use measured reflectance values to infer the optical properties. This is something that in many cases cannot be solved directly, so instead a minimization strategy is deployed. Here

1. Optical properties are guessed and inserted into the forward model
2. Reflectance curves are generated and compared to measured data
3. The error (residual) between these two curves are calculated
    a. if this error is above some previously specified threshold, new guesses are made and steps 1 and 2 are repeated
    b. if this error is below some threshold, the guess values are declared close enough and the minimization loop ends, outputting the final set of guesses as the inferred optical properties

This minimization occurs at every pixel and at every wavelength resulting in images of mua and musp for each LED.

b. Matlab pseudo code:

First we need to define another function that sets up the minimization metric for the inverse solver. There are many approached to doing this, but our group has historically used this as a general function:

```
function y = function_minus_exp(input_parameters,function_to_evaluate,values,varargin)
%This function establishes the minimization metric for comparing model and measured values
%why this particular function is unknown, it has been used for historical reasons
%original author unkown
%input_parameters     -- [mua musp]  (a.k.a "op" used in the model function)
%function_to_evaluate -- model function (e.g. diff_model_for_SFDI)
%values               -- measured calibrated data to compart to model
%varargin             -- additional parameters needed for model --> n, fx

temp = (function_to_evaluate(input_parameters,varargin{:})-values).^2;
y = sum(temp(:));
```

Now back to the inverse solver:

```
%simple inverse solver for diffusion model
%code is not optimized for speed, but to step through data one wavelength/pixel at a time
% wv        -- the wavelength values of the data
% n         -- the index of refraction of the sample measured
% fx        -- the spatial frequencies used in the measurement
% The function outputs a 4-D array that contains images of absorption and scattering
% at each individual wavelength: op_fit_maps(y_pixels,x_pixels,wavelengths,[abs scatt])

%initial guesses to seed the minimization process.
%currently blind guesses, but more intelligent guesses can speed up processing time
mua_guess(w) = 0.02; %units of mm^-1
musp_guess(w) = 1.0; %units of mm^-1

for w = 1:length(wv)  %loop over wavelengths
for i=1:length(calibrated_reflectance(:,1,1,1)) %loop over x dimension
for j=1:length(calibrated_reflectance(1,:,1,1)) %loop over y dimension
op_fit(i,j,w,:) = fminsearch(...
        @function_minus_exp, ... %minimization metric
        [mua_guess(w) musp_guess(w)], ... % initial guess for input into model
        optimset('TolFun',0.00001,'TolX',0.00001),... % options for minimization
        @diff_model_for_SFDI, ... %model (or @MC_model_for_SFDI)
        squeeze(calibrated_reflectance(i,j,w,:))',... %measured (transposed for min metric)
        n, fx); % remaining parameters for the model
end
end
end
```

c. Debugging and Diagnostics

As Sylvain Gioux has provided a nice, independently verified executable GUI for SFDI data processing, this is a great way to compare your own code to a known standard. Use it!

One common occurrence when running this inverse solver fitting routine is convergence. From the specified matlab functions used in the example above, the following warning can come up while processing all wavelengths and pixels in your data set:

```
'Exiting: Maximum number of function evaluations has been exceeded
    - increase MaxFunEvals option.
    Current function value: 0.005173'
```

This occurrence can pop up a few times even when there is nothing wrong with the data set at all (keep in mind that each instance is one event in a total of #wavelengths X #pixels… so often it might only represent a near-negligible fraction of data points contained within the total multi-spectral image set.) However these errors can increase with the noise in the dataset and begin to compromise the integrity of the total dataset. (Keep in mind that the AC component of higher spatial frequencies may only represent a small percentages of the total signal detected, yet though the demodulation process, it will carry most of the noise of the individual images). Assuming that adjacent pixel values in tissue do not vary greatly, smoothing of the image data may be used to great effect either at the demodulation stage and/or calibration stage and help manage the convergence error issues in the inverse solver stage.

Another error that can result in convergence errors and/or highly unrealistic optical properties (10^3-10^4) can result from the dimension of the measured data array that is submitted in the fminsearch.m function. Whether the measured data (R(fx) for a given wavelength and pixel) is input as a column or row can impact how the minimization metric (`@function_minus_exp)`evaluates the data with respect to the model.