Photon Migration Imaging

Toolbox 2001

http://www.nmr.mgh.harvard.edu/DOT/toolbox.htm

1 The Forward Problem

1.1 Geometry of Medium

This section indicates how to define the volume used for the forward problem.

You first need to describe the overall geometry, i.e. is it an 'Infinite', a 'Semi-Infinite' or a 'Slab' geometry.

```
pmi.Fwd.Boundary.Geometry = 'Infinite';
```

For 'Slab' geometry you also need to specify the slab thickness in units of centimeters.

```
pmi.Fwd.Boundary.Thickness = 7;
```

For 'Semi-Infinite' and 'Slab' geometries, it is assumed that one boundary is at $z = 0$ while the other boundary is at $z =$ *pmi.Fwd.Boundary.Thickness*. For a 'Semi-Infinite' geometry it is assumed that the boundary is at $z = 0$ and that the turbid medium is in the negative z half-space.

At the present we are only able to handle uniform computational volumes. In the future we may handle non-uniform voxel sizes and spatial distributions. At present we can only handle 'uniform'.

```
pmi.Fwd.CompVol.Type = 'uniform';
```

Next set the X, Y, Z locations of the voxel centers and the step length between each voxel in units of centimeters. The step lengths must be positive numbers and the progression of voxel centers must be increasing numbers.

```
pmi.Fwd.CompVol.X = [0,1,2,3,4,5,6,7];
pmi.Fwd.CompVol.Y = [0,1,2,3,4,5,6,7];

pmi.Fwd.CompVol.Z = [-2.1,-1.6,-1.1,-0.6,-0.1];

pmi.Fwd.CompVol.Xstep = 1;

pmi.Fwd.CompVol.Ystep = 1;

pmi.Fwd.CompVol.Zstep = 0.5;
```

1.2 Source and Detector Properties

This describes how to define the sources and detectors, including their positions and properties such as wavelength, modulation frequency, etc.

1.2.1 Source Modulation Frequency and Wavelengths

Set the Modulation Frequency for the system, for DC use 0, units are in Mega-Hertz. A [1 x *n*] vector of modulation frequencies can be defined.

```
pmi.Fwd.ModFreq = [0, 200];
```

Lambda is an [1 x *n*] vector where *n* is the number of wavelengths. These wavelengths are in units of nanometers.

```
pmi.Fwd.Lambda = [830, 780, 660]
```

1.2.2 Optode Positions and Amplitudes

To define source positions you can either fill in the source position (Pos Field) manually for non-uniform spacing of the sources or you can call SetOptode if the positions are distributed uniformly.

For example:

```
pmi.Fwd.Src = SetOptode(X, Y, Z, SourceAmplitudes)
pmi.Fwd.Src = SetOptode([1:2], [3:4], [5:6], …
        ones(nSrc,nLambda)*1e-3)
```

will generate the same as

```
pmi.Fwd.Src.Pos =           [1,3,5;
                             1,4,5;
                             2,3,5;
                             2,4,5;
                             1,3,6;
                             1,4,6;
                             2,3,6;
                             2,4,6];
pmi.Fwd.Src.Amplitude = ones(nSrc,nLambda)*1e-3;
pmi.Fwd.Src.Type = 'list';
```

Note: When needed the source amplitude matrix can be filled with different values and does not need to be uniform. The units on the source amplitude is the power entering the tissues in Watts. This includes coupling effects.

You can use the same function call for the detector structure:

```
pmi.Fwd.Det = SetOptode(X, Y, Z, Detector Amplitudes)
pmi.Fwd.Det = SetOptode([1:2], [3:4], [5:6],…
        ones(nDet, nLambda)*1e-3)
```

Note: The detector amplitude is the aperture area [cm^2] times the coupling coefficient for the detector, including the detector quantum efficiency.

1.2.3 The Measurement List

The measurement list identifies the source-detector pairs used in the calculation. Typically for simulations one uses 'all' pairs, but for noisy data the 'Nearest' and 'SNR' methods of preparing the list can be useful.

```
pmi.Fwd.MeasList = genMeasList('all',pmi.Fwd);
```

```
pmi.Fwd.MeasList = genMeasList('Nearest', pmi.Fwd, rho_separation);
```

'Nearest' selects source-detector pairs from *pmi.Fwd.Src* and *pmi.Fwd.Det* with a separation less than or equal to *rho_separation*

```
pmi.Fwd.MeasList = genMeasList('SNR', pmi.Fwd, snr);
```

'SNR' selects source detector pairs from *pmi.Fwd* with a signal to noise ratio greater than or equal to *snr*

pmi.Fwd.MeasList is a matrix with dimensions nMeas x 4 where nMeas is the total number of measurements and the 4 columns are used to identify in order the source index, detector index, modulation frequency index, and lambda index for each measurement.

You can specify pmi.Fwd.MeasList directly without a call to genMeasList( ). For example

```
pmi.Fwd.MeasList =      [1,1,1,1;
                         2,1,1,1;
                         1,2,1,1;
                         2,2,1,1];
```

will specify measurements between sources 1 and 2 and detectors 1 and 2 at the 1st modulation frequency and 1st Lambda. These indices point to the corresponding elements in pmi.Fwd.Src, pmi.Fwd.Det, pmi.Fwd.ModFreq, and pmi.Fwd.Lambda.

1.3 Optical Properties of the Medium

Index of Refraction Vector with the same length as the number of wavelengths

```
pmi.Fwd.idxRefr = [1.37, 1.37, 1.37];
```

Background scattering coefficient, μ_s, in units of [cm^{-1}] for each wavelength

```
pmi.Fwd.Mu_s = [100, 100, 100];
```

Background average cosine of the scattering angle.

```
pmi.Fwd.g = [0.9, 0.9, 0.9];
```

Calculate the corresponding μ_s from the given g and μ_s

```
pmi.Fwd.Mu_sp = pmi.Fwd.Mu_s .* (1 – pmi.Fwd.g);
```

Calculate the velocity in the medium [*cm/sec*]

```
pmi.Fwd.v = 2.99e10 ./ pmi.Fwd.idxRefr;
```

1.3.1 Chromophores

Setup the chromophores used in the forward model. You don't need to use this, if you want you can input the absorption directly, but this allows you to exploit the built in absorption spectra for Oxy and de-Oxy hemoglobin, water, cytochrome oxidase and lipipmi. The units for all Chromophores are Moles per liter except for H2O which is given as a volume fraction.

Name of the first chromophore in the background, ['Hb','HbO','H2O','aa3','lipid']

```
pmi.Fwd.Cphore(1).Name  = 'HbO';
```

Concentration of the First chromophore.

```
pmi.Fwd.Cphore(1).Conc = 85e-6;
```

Name of the second chromophore in the background, ['Hb','HbO','H2O','aa3','lipid']

```
pmi.Fwd.Cphore(2).Name = 'Hb';
```

Concentration of the Second chromophore

```
pmi.Fwd.Cphore(2).Conc = 30e-6;
```

Now we can calculate the absorption using the following function. Note that if you input the absorption directly you don’t need to use this function.

```
pmi.Fwd.Mu_a = GetMua(pmi.Fwd.Lambda, pmi.Fwd.Cphore);
```

1.3.2 Spatial Variation in the Optical Properties of the Medium

The following describes how to treat spatial variations of the optical properties within the medium.

For only one object with constant optical properties that are different than the background optical properties described above, you would do the following. If you have more than one object in the medium then you would need to do the following for each *pmi.Object{number}* of the object. As used for the background description you can use the chromophore structure to define the absorption parameters or define the absorption directly:

```
pmi.Object{1}.Cphore(1).Name = ‘HbO’;
pmi.Object{1}.Cphore(1).Conc = 50e-6;
pmi.Object{1}.Cphore(2).Name = ‘Hb’;
pmi.Object{1}.Cphore(2).Conc = 10e-6;
pmi.Object{1}.Mu_a = GetMua(pmi.Fwd.Lambda.pmi.Object{1}.Cphore);
```

Now describe where the object is located and whether it is a sphere or just a block.

```
pmi.Object{1}.Type = ‘Sphere’;
pmi.Object{1}.Pos = [x_center, y_center, z_center];
pmi.Object{1}.Radius = radius;
```

Or for a Block:

```
pmi.Object{1}.Type = ‘Block’;
pmi.Object{1}.Pos = [x_center, y_center, z_center];
pmi.Object{1}.Dims = [x_Dimension, y_Dimension, z_Dimension];
```

You also need to define the scattering and *g* of the object for each wavelength.

```
pmi.Object{1}.Mu_s = 100 * ones(1,nLambda);

pmi.Object{1}.g = 0.9 * ones(1,nLambda);

pmi.Object{1}.Mu_sp = pmi.Object{1}.Mu_s.* (1 – pmi.Object{1}.g)
```

1.4 Solution Methods for the Forward Problem

1.4.1 Uniform Optical Properties

In the case of media with spatially uniform optical properties, the method is:

pmi.Fwd.Method.Type = 'Helmholtz Homogeneous';

This finds the analytic solution to the Helmholtz equation for the given geometry.

1.4.2 Linear Methods for Non-Uniform Optical Properties

The First Born approximation is specified by:

pmi.Fwd.Method.Type = 'Born';

The Rytov approximation is specified by:

pmi.Fwd.Method.Type = 'Rytov';

1.4.3 Non-Linear Methods for Non-Uniform Optical Properties

The N^{th}-Born approximation is specified by, where *n* is the order of the approximation:

pmi.Fwd.Method.Type = 'BornN';

pmi.Fwd.Method.Born_order = n;

The Full Born solution to the diffusion equation is specified by (Note: This is equivalent to the *'BornN'* solution solved to all orders):

pmi.Fwd.Method.Type = 'FullBorn';

The Extended Born solution and the radius of influence (in centimeters) is indicated by

pmi.Fwd.Method.Type = 'ExtBorn';

```
pmi.Fwd.Method.ExtBorn_Radius = Radius;
```

The exact solution for a spherical object in a homogeneous medium is specified by (only works for an 'Infinite' medium):

```
pmi.Fwd.Method.Type = 'Spherical';
```

1.5 Matrix Generation

Many solution methods for the forward problem require a matrix to be generated. Before calculating the forward matrix it is necessary to specify the quantities to be considered. Setting these flags to '1' means that you have a perturbation of this kind to consider and a '0' means to ignore this perturbation.

To consider an absorption perturbation:

```
pmi.Fwd.Method.ObjVec_mua = 1;
```

To consider a perturbation in the reduced scattering coefficient:

```
pmi.Fwd.Method.ObjVec_musp = 1;
```

To consider a perturbation in the source and detector amplitudes:

```
pmi.Fwd.Method.ObjVec_sd = 1;
```

Once all these fields are filled, the forward matrix is calculated by:

```
pmi.Fwd = genFwdMat(pmi.Fwd);
```

This fills in the following field:

```
pmi.Fwd.P(idxLambda).A
pmi.Fwd.P(idxLambda).PhiInc
```

1.6 Generating the Simulated Data

Now we can create the simulated measurement data. This produces the detected photon fluence given the above specifications.

```
pmi = genMeasData(pmi);
```

This fills in the following fields:

```
pmi.Fwd.P(idxLambda).PhiInc
pmi.Fwd.P(idxLambda).PhiScat
pmi.Fwd.P(idxLambda).PhiTotal
```

1.7 Adding Noise

These parameters are used to create the noise and add it to the fluence of the forward model.

Consider the effects of Shot Noise (Yes = 1, No = 0)

pmi.Noise.ShotSNRflag = 1;

The Shot noise is determined by the measurement Bandwidth (e.g. 20 Hz below) and the previously specified source and detector amplitudes as well as the characteristics of the medium.

pmi.Noise.ShotSNR = 20;

Consider the effects of Electronic Noise (Yes = 1, No = 0)

pmi.NoiseElectronicSNRflag = 1;

The electric noise is the noise equivalent power of your detector (NEP) times the square root of measurement bandwidth:

pmi.Noise.ElectronicSNR = 1e-12;

Generate a specific instance of the noise:

pmi = genNoise(pmi);

This function copies

pmi.Fwd.P(idxLambda).PhiTotal

into

pmi.Inv.P(idxLambda).PhiTotal

calculates the variance

pmi.Noise.P(idxLambda).TotalVar

and calculates the measurement noise using a normal distribution of width

sqrt(pmi.Noise.P(idxLambda).TotalVar)

which is then added to pmi.Inv.P(idxLambda).PhiTotal to create the noisy measurement data stored in

pmi.Inv.P(idxLambda).PhiTotalN

2 The Inverse Problem

It is necessary to specify the inverse problem parameters. A number of the parameters are defined using the same structure as in the forward problem.

2.1 Generate Measurement List

This must be the same as the forward problem.

```
pmi.Inv.MeasList = pmi.Fwd.MeasList
```

2.2 Computational Volume

The computational volume for the inverse problem should differ from the forward problem so as not to perform the INVERSE CRIME (as called by Simon Arridge).

```
pmi.Inv.Boundary.Geometry = 'Semi-Infinite';
pmi.Inv.CompVol = 'uniform';
pmi.Inv.CompVol.X = [0,1,2,3,4,5,6,7];
pmi.Inv.CompVol.Y = [0,1,2,3,4,5,6,7];
pmi.Inv.CompVol.Z = [-0.1,-0.6,-1.1,-1.6,-2.1];
pmi.Inv.CompVol.Xstep = 1;
pmi.Inv.CompVol.Ystep = 1;
pmi.Inv.CompVol.Zstep = -0.5;
```

2.3 Inverse Model Method

The inverse method does not need to be the same as the forward method. A popular linear method is:

```
pmi.Inv.Method.Type = 'Rytov';
```

2.4 Calculate the Matrix for the Inverse Problem

The reconstruction needs the forward matrix.

To reconstruct an absorption perturbation:

```
pmi.Inv.Method.ObjVec_mua = 1;
```

To reconstruct a perturbation in the reduced scattering coefficient:

```
pmi.Inv.Method.ObjVec_musp = 1;
```

To reconstruct a perturbation in the source and detector amplitudes:

```
pmi.Inv.Method.ObjVec_sd = 1;
```

Once all these fields are filled, the matrix is calculated by:

```
pmi.Inv = genFwdMat(pmi.Inv);
```

2.5 Reconstruction Parameters

Setup the appropriate Reconstruction parameters by filling in the field that corresponds to the chosen inversion method. Generating the reconstruction (explained in the next section) will result in some fields be adjusted depending on the reconstruction algorithm. The parameters which are thus altered in a way that depends on the algorithm are described within the corresponding algorithm.

2.5.1 SIRT

The algorithm flag is set to

```
pmi.Recon.ReconAlg = 'SIRT';
```

where the number of iterations is given by *nSIRT*:

```
pmi.Recon.SIRT_nIter = nSIRT
```

2.5.2 ART

The algorithm flag is set to

```
pmi.Recon.ReconAlg = 'ART';
```

where the number of iterations is given by *nART*:

```
pmi.Recon.ART_nIter = nART;
```

2.5.3 Truncated SVD

The algorithm flag is set to

```
pmi.Recon.ReconAlg = 'TSVD';
```

where the number of singular values used in the inverse is given by *nTrunc*:

```
pmi.Recon.TSVD_nSV = nTrunc;
```

Generating the reconstruction will then fill in the fields

```
pmi.Recon.W(idxLambda).Uecon
pmi.Recon.W(idxLambda).Secon
pmi.Recon.W(idxLambda).Vecon
```

pmi.Inv.P(idxLambda).Ainv

corresponding to the singular value decomposition (A = U S V^T) of the system matrix and the pseudo-inverse given by the truncated SVD.

The user has the option of using the MATLAB SVDS rather than the default SVD. This is determined by the flag

pmi.Recon.TSVD_FullSVS = 1;

where the 1 indicates the use of SVD. A 0 indicates the use of SVDS. The SVDS may be faster when *nTrunc* is small.

The user can tell the reconstruction routine to calculate the SVD or use the pre-calculated SVD parameters. Normally you should leave this set to '1' unless you are repeating a reconstruction that does not change the system matrix in which case you use '0'.

pmi.Recon.TSVD_CalcSVD = 1;

The user can specify that the SVD be obtained for the system matrix squared by setting the flag

pmi.Recon.TSVD_Lsq = 0;

A value of '0' will take the SVD of A. A value of '1' takes the SVD of AA^T (i.e. in the measurement space). A value of '2' takes the SVD of A^TA (i.e. in the voxel space).

2.5.4 Tikhonov

The algorithm flag is set to

pmi.Recon.ReconAlg = 'TIK';

The regularization parameter is specified relative to the maximum value within the matrix AA^T.

pmi.Recon.TIK_alpha = 1e-3;

Generating the reconstruction will then fill in the field

pmi.Inv.P(idxLambda).Ainv

corresponding to the pseudo-inverse given by

$$A^T * \mathrm{inv}(\ A*A^T + \mathit{TIK\_alpha} * \max(A*A^T) * \mathrm{eye}(\mathrm{size}(A*A^T))\ )$$

2.6 Reconstruction

Now we get to reconstruct the image.

```
pmi = genRecon( pmi );
```

This fills in the appropriate parameters in pmi.Recon.

The absorption perturbation image is stored in

```
pmi.Recon.Mua
```

The reduced scattering perturbation image is stored in

```
pmi.Recon.Musp
```

The reconstructed Source and Detector amplitude perturbations

```
pmi.Recon.SD
```

2.7 Image Viewing

To view the image use

```
showImage( pmi )
```

This takes the images stored in pmi.Recon.Mua and pmi.Recon.Musp and displays them in separate figures.