Medical MRI, Assignment 1

Anders Olsen

March 2020

1 Abstract

Estimation of the tissue-dependent parameters Proton Density (PD) and T1 can be performed using Fast low-Angle Shot (FLASH) imaging with a selection of tip-angles (α). This report summarizes the work done in the first assignment of the course 22506 Medical Magnetic Resonance Imaging, where 3D FLASH images acquired at 4 different tip-angles were used to estimate PD and T1 voxel-wise using non-linear curve-fitting. The code and results were discussed with fellow students Marie Garnæs and Mikkel Ranum Boesen.

2 Methods

In a static magnetic field, a single, homogeneous object at equilibrium (say, a Hydrogen nucleus) has longitudinal magnetization M_0 and zero transverse magnetization. After inducing an excitation pulse with tip-angle α, the solution to the Bloch equation gives:

$$M_z(t) = M_z(0^-)\cos(\alpha)e^{-t/T_1} + M_0(1 - e^{-t/T1})$$

Here, T1 is the time constant for the longitudinal magnetization recovery and $M_z(0^-) = M_0$ at equilibrium. Just before the second excitation, the transverse magnetization has been killed by spoiling, and the longitudinal magnetization ($M_z(0^-)$) can be found by replacing t with the repetition time TR in the above equation. In the general case, M_z will not have recovered fully, leaving less magnetization to be converted to measurable transverse magnetization by the time of the second excitation. After several such excitation-recovery sequences, a steady-state is reached:

$$M_z(0^-) = M_0\frac{1 - e^{-TR/T1}}{1 - cos(\alpha)e^{-TR/T1}} \tag{1}$$

The measured signal is then proportional to the steady-state transverse magnetization at time $t = 0^+$:

$$M_{SS} = sin(\alpha)M_0\frac{1 - e^{-TR/T1}}{1 - cos(\alpha)e^{-TR/T1}} \tag{2}$$

Further, M_0 is proportional to the gyromagnetic ratio of (in this case) the Hydrogen atom, the strength of the B_0-field, temperature and the proton density (PD). This proportionality is noisy and unstable, though, and PD images are often normalized. This also means that M_0 is an estimate of PD. Also, we will assume that no T2-relaxation has occurred before readout, i.e. echo time $TE \approx 0$. This is a coarse assumption, but will significantly aid analysis as T2-estimation need not be performed.

2.1 Image acquisition

One subject was scanned 4 times at Hvidovre Hospital using a Philips 3-Tesla scanner and tip-angles $2, 8, 12, 22$. A sagittal slice of the resulting images is seen in Figure 1. All 4 images were acquired using $TR = 10ms$, $TE = 3.46ms$ and are shown with the same scaling. The Ernst angle for both grey matter (GM) and white matter (WM) is closer to $\alpha = 8$ than any of the other values. For cerebrospinal fluid (CSF), the highest values are supposed to be close to the lowest tip-angle. In this case, however, values within the lateral ventricle were the same for tip-angles $\alpha = 2$ and $\alpha = 8$.

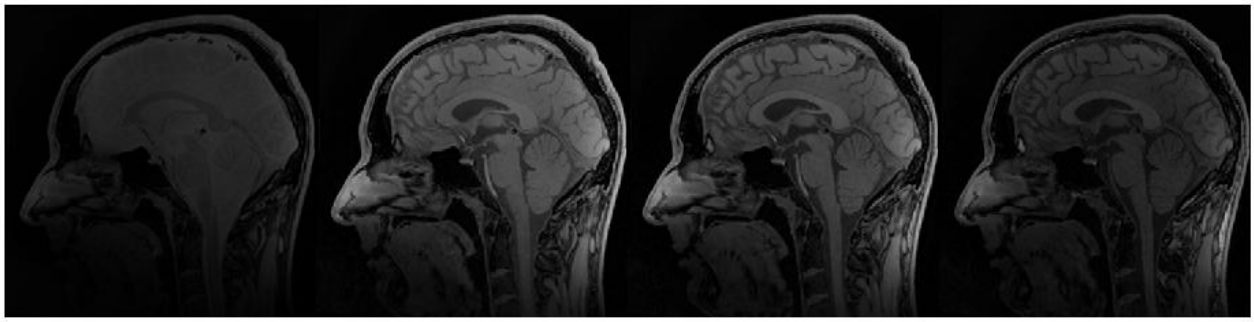

Figure 1: FLASH images with tip-angles (left-to-right) 4, 8, 12 and 22, respectively

2.2 Non-linear curve-fitting

Assuming all other parameters in Equation 2 to be constant, PD and T1 are the only varying tissue-dependent parameters, which means they can be estimated using non-linear approximation of overdetermined systems. In this case, we use the built-in MATLAB function *lsqcurvefit* voxel-wise. That is, for each voxel, the voxel-intensities from the 4 images are mapped to only two values, PD and T1.

To alleviate computational time, a simple masking procedure was implemented. For each slice in the horizontal plane, the pixels outside the head having a value below the threshold were masked out. In practice this was performed in the following way:

- For each row in the image, starting from the left and going right, all pixels in the row until the first encounter of a pixel value above the threshold were set to zero.
- The same procedure was carried out from the right going left.

Moreover, all slices below the cerebellum were masked out. This procedure reduced the number of voxels to be fitted by a factor of 4 with a threshold of 150. It also significantly shortened computation time, allowing for the full volume to be estimated in less than 3 hours and thereby alleviating the trial-and-error procedure necessary to select appropriate slices in all three directions.

For the resulting T1 image, the lateral ventricle was selected in the sagittal plane using the MATLAB functions *drawpolygon* and *createMask*. The average pixel value of the selected voxels was used to normalize the PD image, in order for CSF to have an average value of one. An illustration of the selection procedure is shown in Figure 2.

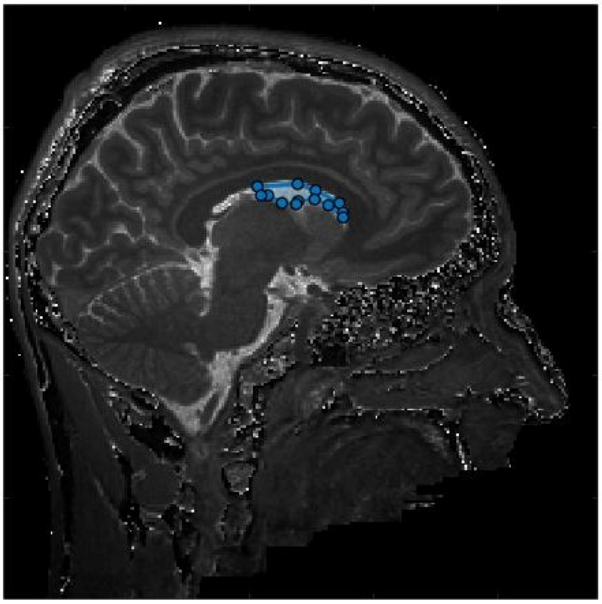

Figure 2: An illustration of the polygon-drawing procedure in the lateral ventricle on the estimated sagittal T1-slice

Finally, upon inspection of the resulting images, outliers were removed for the PD image if the voxel value was above 20, and for the T1 image if the value was above 6 seconds.

3 Results

Figure 3 shows the PD and T1-weighted images in the sagittal, coronal, and horizontal plane, respectively. Estimated PD and T1 values from the putamen (gray matter), corpus callosum (white matter) and lateral centricle (CSF) are provided in Table 1.

	PD	T1 [s]
Putamen	0.91	1.24
Corpus callosum	0.85	0.99
Lateral ventricle	1	3.95
Gray matter	0.75	1.45
White matter	0.65	0.79
CSF	1	4.16

Table 1: Top: Estimated PD and T1 values at representative tissues. Bottom: Standard values at 3T (PD values from Nishimura, T1 values from Lin et al., Measurements of T1 Relaxation times at 3.0T: Implications for clinical MRA, Proc. Intl. Soc. Mag. Reson. Med 9 (2001)).

4 Discussion

Upon first glance at the PD images in Figure 3(top), it can be very difficult to distinguish the different brain parts as the contrast in the image is very low. This could be solved by visualizing

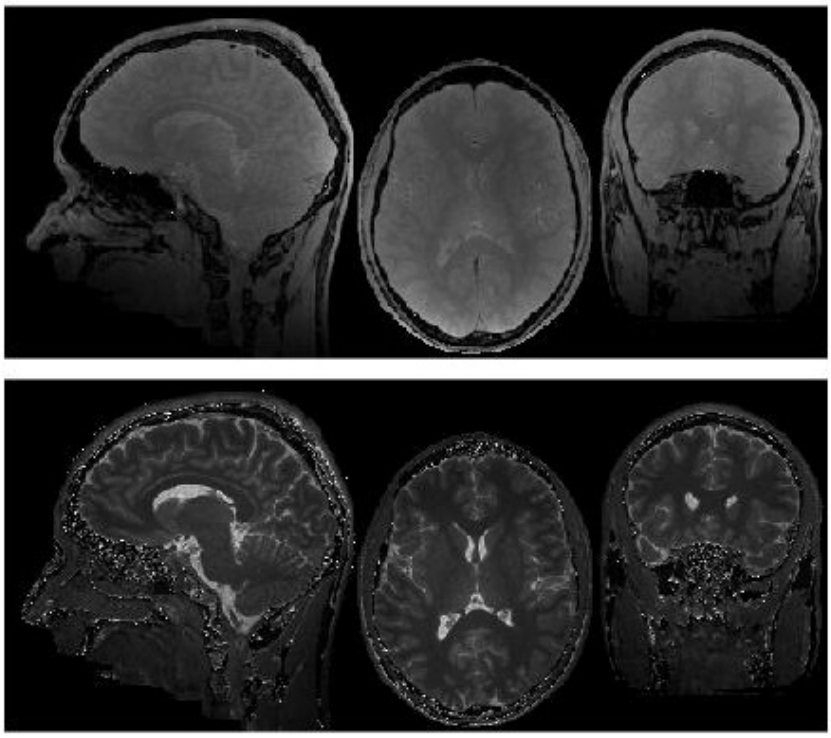

Figure 3: The PD (top) and T1 (bottom) images seen in the sagittal (left), horizontal (middle) and coronal (right) planes.

the image using a smaller pixel-window, however, it is also important to notice that the values shown in Table 1 show that the PD values for GM and WM differ from standard values and provide less contrast. We do get higher PD values for putamen than corpus callosum, but the difference is not nearly as high as in standard values.

Likewise, for the T1 images, it can be difficult to distinguish GM and WM, which is also reflected in the small difference between T1-values for putamen and corpus callosum.

Some error sources have been identified to explain this problem. We use a non-linear curve-fitting method that gives many outliers, which are especially visible in the T1-images near the skull. These outliers may arise from local inaccuracy of the signal equation (e.g. in bone) or the algorithm encountering a local minimum instead of the global. Furthermore, the selected scanning method (FLASH) generates images using a very short TR, including spoiling before each excitation. Spoiling removes all transverse magnetization by dephasing. Repeated spoiling shortly after one another may refocus some of the lost transverse magnetization and give way for unwanted signal. Finally, PD and T1 values in the putamen and corpus callosum can be very hard to estimate, as they are taken from a slice of the brain, and the selected areas may include a mixture of tissues. For example, the ventricle may be outlined too close to the edges, or putamen estimates may include poorly visible white matter and CSF.

In this assignment we have assumed that using very short $TR << T2$ makes the T2-relaxation time negligible. But is that a valid assumption? Including T2-relaxation, the steady-

state equation (Equation 2) has an extra term:

$$M_{SS} = sin(\alpha)M_0\frac{1-e^{-TR/T1}}{1-cos(\alpha)e^{-TR/T1}}e^{-TR/T2^*} \tag{3}$$

When plugging in standard values for T2, we obtain a decrease in signal by $\approx 10\%$ for GM and WM and negligible for CSF. This may explain that CSF is indistinguishable from GM and WM in the PD images. Note that in Equation 3, $T2^*$ is used and since $T2^* < T2$ the real effect may be smaller. The difference between $T2^*$ and $T2$ is independent of tissue type, however.

Near the occipital lobe, brighter voxel values are observed in the raw images as well as the PD images. This may be due to a receiver coil being situated near the posterior part of the skull. This does cause some field inhomogeneity, which in turn may distort PD and T1 estimates.

All in all, the computed images resemble real-world images to some degree, but it is clear that some extra estimation tools should be used to obtain images that can be used in the clinic.

5 MATLAB scripts

5.1 Main script

```
%%% Initialize
clear
directory = 'C:\Users\ander\Documents\10. Semester\Medical MRI\ExerciseData\ass1';
datadir = strcat(directory,'\Data\');
addpath(genpath(directory))
cd(directory)


%% load data and extract information
i1 = get_mri_data(datadir,{'X98279_WIP_sT1W_3D_FFE_31547_4_1'},1);
i2 = get_mri_data(datadir,{'X98279_WIP_sT1W_3D_FFE_31547_5_1'},1);
i3 = get_mri_data(datadir,{'X98279_WIP_sT1W_3D_FFE_31547_6_1'},1);
i4 = get_mri_data(datadir,{'X98279_WIP_sT1W_3D_FFE_31547_7_1'},1);
alpha = [i1.info.TipAngle,i2.info.TipAngle,i3.info.TipAngle,i4.info.TipAngle];
nVoxels = size(i1.data,1)*size(i1.data,2)*size(i1.data,3);
TR = i1.info.RepetitionTime;
numims = 4;

%% Decide which slices to use
slice = 90;
imagesc([i1.data(:,:,slice), i2.data(:,:,slice), ...
    i3.data(:,:,slice), i4.data(:,:,slice)]);colormap gray;axis image

data = cat(4,i1.data,i2.data,i3.data,i4.data);
dataM = fliplr(data(:,:,80,:));
montage ( dataM ,'Size',[1,4],'DisplayRange',[0 1600])

%% perform (simple) masking on selected slices
sag80 = i2.data(:,:,80);
cor80 = i2.data(80,:,:);
trans140 = i2.data(:,140,:);

threshold = 150;
masksag80 = masksimple(sag80,threshold);
maskcor80 = masksimple(cor80,threshold);
masktrans140 = masksimple(trans140,threshold);

figure;
subplot(1,2,1)
imagesc(sag80);colormap gray
subplot(1,2,2)
imagesc(masksag80);colormap gray
```

```
%% perform PD and T1 eval on the saggital slice

sag80 = data(:,:,90,:);

[PDsag,T1sag] = nonlincurvefit(sag80,masksag80,alpha,TR);

%%% Remove values above a threshold
PDsag2 = PDsag;
PDsag2(PDsag2>20)=0;
T1sag2 = T1sag;
T1sag2(T1sag2>6)=0;
figure;
subplot(1,2,1);imagesc(PDsag2);colormap gray; axis image
subplot(1,2,2);imagesc(T1sag2);colormap gray; axis image

%%%%%%% Normalizing PD using polygon mask on ventricle (saved)
load polycsf.mat %polygon for ventricle
ROIcsf = createMask(polycsf);
PDcsf = mean(PDsag(ROIcsf));
T1csf = mean(T1sag(ROIcsf));
PDsag3 = PDsag2./PDcsf;

%%%% find corpus callosum mask values
load polyWM.mat
ROIWM = createMask(polyWM);
PDWM = mean(PDsag3(ROIWM));
T1WM = mean(T1sag2(ROIWM));

%% Do the same on the other sides

cor80 = data(90,:,:,:);
trans140 = data(:,140,:,:);
[PDcor,T1cor] = nonlincurvefit(cor80,maskcor80,alpha,TR);
[PDtrans,T1trans] = nonlincurvefit(trans140,masktrans140,alpha,TR);

PDcor2 = PDcor./PDcsf;
PDcor2(PDcor2>20)=0;
T1cor2 = T1cor;
T1cor2(T1cor2>6)=0;

PDtrans2 = PDtrans./PDcsf;
PDtrans2(PDtrans2>20)=0;
T1trans2 = T1trans;
T1trans2(T1trans2>6)=0;
```

```
PDdata = [fliplr(PDsag3),flipud(PDcor2),PDtrans2];
figure,imagesc(PDdata,[0,3]);colormap gray;axis image;axis off
T1data = [fliplr(T1sag2),flipud(T1cor2),T1trans2];
figure,imagesc(T1data,[0,6]);colormap gray;axis image;axis off

figure,imagesc(flipud(T1cor2));colormap gray;axis image

%%% Find values within putamen
load polyGM.mat
ROIGM = createMask(polyGM);
PDGM = mean(PDcor2(ROIGM))
T1GM = mean(T1cor2(ROIGM))
%% Do on full volume

mask = maskfull(i2.data,150); %approved
PDfull = zeros(size(i2.data));
T1full = zeros(size(i2.data));

for slice = 1:160
    dat = squeeze(data(slice,:,:,:));
    mas = squeeze(mask(slice,:,:));
    [PDfull(slice,:,:),T1full(slice,:,:)] = nonlincurvefit(dat,mas,alpha,TR);
%changed disp in fun to 10000
disp(['Done with slice ',num2str(slice),' out of 160'])
end


PDfull2 = PDfull;
PDfull2(PDfull2>10)=0;
T1full2 = T1full;
T1full2(T1full2>6)=0;

slicei(PDfull2);colormap gray;
slicei(T1full2);colormap gray;

figure;imagesc(squeeze(PDfull2(:,:,80)));axis image;colormap gray

slice90 = flipud(squeeze(T1full2(90,:,:)));
ROIGM = createMask(polyGM);
T1GM = mean(slice90(ROIGM));
```

5.2 Slice-masking

```
function mask = masksimple(slice,threshold)
% Construct a simple mask by going from left to right and right to left in
% an image and output 0 if the voxel value is less than threshold,
% otherwise zero. For every line, the algorithm stops upon first encounter
```

```
% of a voxel value greater than threshold, to avoid masking out the inside
% of the head.
% To achieve good masking, select an image (i3) with high voxel values at
% the skin, and a high threshold to mask out everything outside the head.
slice = squeeze(slice);
topdown = size(slice,1);
leftright = size(slice,2);

mask = ones(size(slice));

for i=1:topdown

    j = 1;
    while j
        if j>leftright
            j = [];
        elseif slice(i,j)<threshold
            mask(i,j)=0;
            j = j+1;
        else
            j = [];
        end
    end

    k = 1;
    while k
        if k>leftright
            k = [];
        elseif slice(i,leftright+1-k)<threshold
            mask(i,leftright+1-k)=0;
            k = k+1;
        else
            k = [];
        end
    end


end
mask = logical(mask);
end
```

5.3 Full masking

```
function mask = maskfull(vol,threshold)


mask = ones(size(vol));
```

```
%remove from 160 up

for slice = 1:240

data = squeeze(vol(slice,:,:));
for i=1:240

    j = 1;
    while j
        if j>160
            j = [];
        elseif data(i,j)<threshold
            mask(slice,i,j)=0;
            j = j+1;
        else, mask(slice,i,j:j+12)=0;
            j = [];
%             mask(i,j-3:j-1)=1;
        end
    end

    k = 1;
    while k
        if k>160
            k = [];
        elseif data(i,161-k)<threshold
            mask(slice,i,161-k)=0;
            k = k+1;
        else, mask(slice,i,161-k-13:161-k)=0;
            k = [];
        end
    end


end
end
mask=mask(1:240,1:240,1:160);
mask(160:end,:,:) = 0;
mask = logical(mask);
end
```

5.4 Non-linear curvefitting

```
function [PD,T1] = nonlincurvefit(slices,mask,alpha,TR)
% This function finds PD and T1 values voxel-wise on an array of minimum 3
% slices, either coronal, saggital or transversal
```

```
% preprocessing
slices  = squeeze(slices);
szorig  = size(slices);
Nvoxels = szorig(1)*szorig(2);
slices  = reshape(slices,Nvoxels,szorig(3));

mask    = reshape(mask,Nvoxels,1);
Nmask   = sum(mask);

slices  = double(slices(mask,:)); %%% Use only voxels within mask
slices  = slices./max(slices(:)); %normalize to 0 and 1


%%% Curvefit
x = zeros(Nmask,2); %preallocation

fun = @(x,alpha)sind(alpha).*x(1).*(1-exp(-TR./x(2)))./(1-cosd(alpha).*exp(-TR./x(2)));
x0 = [0.1,0.1];
opts = optimset('Display','off');
lowbound = [0,0];

tic
for a = 1:Nmask
    x(a,:) = lsqcurvefit(fun,x0,alpha,slices(a,:),lowbound,[],opts);

    if ismember(a,[10000:10000:Nmask])
        disp(['Done with nr ',num2str(a),' out of ',num2str(Nmask)])
        toc
    end
end

%%% Output
PD = zeros(Nvoxels,1);
PD(mask) = x(:,1);
T1 = zeros(Nvoxels,1);
T1(mask) = x(:,2);

PD = reshape(PD,szorig(1),szorig(2));
T1 = reshape(T1,szorig(1),szorig(2));

end
```