

REGULARIZATION
FOR
MRI DIFFUSION INVERSE PROBLEM

REGULARIZATION
FOR
MRI DIFFUSION INVERSE PROBLEM

By
TAHANI ALMABRUK, B.Sc.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree
Master of Science

McMaster University
©Copyright by Tahani Almabruk, June,17, 2008

MASTER OF SCIENCE (2008)
COMPUTING AND SOFTWARE

McMaster University
Hamilton, Ontario

TITLE: Regularization for MRI Diffusion Inverse Problem.

AUTHOR: Tahani Almabruk, B.Sc.

SUPERVISORS: Dr. Christopher Kumar Anand.

NUMBER OF PAGES: xii, 81

Abstract

In this thesis, we introduce a novel method of reconstructing fibre directions from diffusion images. By modelling the Principal Diffusion Direction PDD (the fibre direction) directly, we are able to apply regularization to the fibre direction explicitly, which was not possible before.

Diffusion Tensor Imaging (DTI) is a technique which extracts information from multiple Magnetic Resonance Images about the amount and orientation of diffusion within the body. It is commonly used for brain connectivity studies, providing information about the white matter structure.

Many methods have been represented in the literature for estimating diffusion tensors with and without regularization. Previous methods of regularization applied to the source images or diffusion tensors. The process of extracting PDDs therefore required two or three numerical procedures, in which regularization (including filtering) is applied in earlier steps before the PDD is extracted. Such methods require and/or impose smoothness on all components of the signal, which is inherently less efficient than using regularizing terms that penalize non-smoothness in the principal diffusion direction directly.

Our model can be interpreted as a restriction of the diffusion tensor model, in which the principal eigenvalue of the diffusion tensor is a model variable and not a derived quantity.

We test the model using a numerical phantom designed to test many fibre orientations in parallel, and process a set of thigh muscle diffusion-weighted images.

Acknowledgements

I would like to present a great thanks to my supervisor Dr. Christopher Anand, for his help and guidance during my study at McMaster university. He is really an ideal example of the supervisor who provides his students with a complete support and encouragements.

Thanks to my family who gave me the opportunity to travel and study here and for their standing by me during this important future step.

Thanks a lot to all my friends specially Mr. Agrera and Huda Khatab.

Notation

MRI: Magnetic Resonance Imaging

B_0 : The external magnetic field

ω_0 : Larmor Frequency

RF: Radio Frequency

B_1 : Radio Frequency Magnetic field

M_z : Magnetization

AC: Alternating Current

T_1 : Longitudinal Relaxation Time

T_2 : Transverse Relaxation Time

g: Gradient Magnetic Field

ADC: Apparent Diffusion Coefficient

DT: Diffusion Tensor

PGSE: Pulsed Gradient Spin Echo

WM: White Matter

GM: Gray Matter

PDD: Principle Diffusion Direction.

Pixel: The minimum unit of information that can be distinguished in a 2D image.

Voxel: The minimum unit of information that can be distinguished in a 3D volume (position, color, and density).

Contents

Abstract	iii
Acknowledgements	v
Notation	vii
List of Figures	xi
1 Fundamental Basics of MRI	1
1.1 External Static Magnetic Field (B_0)	1
1.2 Radio Frequency (B_1)	3
1.3 Relaxation	4
1.3.1 Longitudinal Relaxation Time (T_1)	4
1.3.2 Transverse Relaxation Time (T_2)	4
1.4 Linear Magnetic Field	5
1.5 Spatial Encoding	5
1.5.1 Slice Selection	6
1.5.2 Phase Encoding	7
1.5.3 Frequency Encoding	7
2 Diffusion Tensor Imaging	9
2.1 Free Diffusion	9
2.2 Restricted Diffusion	10
2.3 Isotropy and Anisotropy	11
2.4 Diffusion Weighted-Images and Diffusion Tensor	11
2.5 Diffusion Tensor Eigensystem	15
2.6 Fibre Tracking	16
3 Diffusion Tensor Estimation and Regularization	21
3.1 Diffusion Tensor Estimation	21
3.1.1 Least Squares Estimation Method	21

3.1.2	Our Method	22
3.2	Diffusion Tensor Regularization	23
3.2.1	Tikhonov Regularization	23
4	Numerical and Experimental Results	27
4.1	Experiment Environment	27
4.2	Numerical Results	28
4.2.1	Solvers	28
4.2.2	Initial Guess X_0	29
4.2.3	Choice of Gradient Encodings	30
4.2.4	Noise	30
4.2.5	Regularized Fiber Direction Images	30
4.3	Experimental Results	31
5	Conclusions and Future Work	45
5.1	Conclusions	45
5.2	Future Work	45
	Appendix	47
	Bibliography	47

List of Figures

1.1	Single spin as a small magnet with its poles.	1
1.2	Spin random movement in the absence of an external magnetic field. Spin magnetic moments cancel each other and $M_z = 0$	2
1.3	Spin alignment in the presence of external magnetic field.	3
1.4	Spins are subjected to the external magnetic field and they precess around B_0 at a certain frequency ω_0	3
1.5	Choosing a certain slice via slice selection process.	6
1.6	Identifying each row within the selected slice via phase encoding process.	7
1.7	Identifying each column within the selected slice via frequency encoding process	8
2.1	Restricted diffusion.	10
2.2	The Stejskal-Tanner sequence.	13
2.3	Graphical representation of fibre direction.	16
2.4	Weighted-images with corresponding gradients magnetic field.	17
2.5	Tensor eigensystem.	18
2.6	Simple interpretation for the Seed Point Method.	18
2.7	Reconstructed fibre direction images using MRI-DTI.	19
3.1	Graphical representation of the neighbour-pixels for a certain pixel.	26
4.1	Graphical representation of fibre direction results from using using two initial guesses $[0\ 0\ 0\ 0]$ and $[0\ 0.1\ 0.1\ 0]$: (a) by Tomlab solvers, (b) Fminsearch solver	33
4.2	Graphical representation of fibre direction results from Fminsearch with an initial guess $[0\ 0\ 0\ 0]$	34
4.3	Created shapes each of which surround a certain pixel (gray color) by different neighbour-pixels (black color).	34

4.4	Graphical representation of regularized fibre direction (a) centred, four neighbour pixel experiment, (b) best-of-nine neighbourhood experiment.	35
4.5	Graphical representation of the calculated differences between the regularized fibre direction and the correct model which results from (a) centred, four neighbour pixel experiment, (b) best-of-nine neighbourhood experiment . Small arrows indicate small errors, without variation across the image or correlation with the ring structure.	36
4.6	Diffusion weighted images obtained by adding more noise (20 instead of .01).	37
4.7	Graphical representation shows the calculated difference between the correct model and the results of (a) centred, four neighbour pixel experiment (long arrows specially around edges indicate big errors) and (b) best-of-nine neighbourhood experiment (error reductions can be seen specially around the edges (1), (2), (3), and (4)).	38
4.8	T_2 weighted image through the thigh of a healthy volunteer. Here an EPI diffusion encoding sequence was used with $b = 0$.	39
4.9	The diffusion weighted images with the used gradient at each case.	40
4.10	Reconstructed image (anisotropic component) without regularization.	41
4.11	Reconstructed images when $\alpha=1000$, (a) 4 neighbour-pixels regularization method (b) 8 neighbour-pixels regularization method.	42
4.12	Reconstructed images (b) 8 neighbour-pixels regularization method ($\alpha=1000$) (c) 8 neighbour-pixels regularization method $\alpha=10000$.	43
4.13	Reconstructed images F_z -components	44

Chapter 1

Fundamental Basics of MRI

Magnetic Resonance Imaging (MRI) depends on the existence of water molecules in the body. Hydrogen forms a large part of the water molecule (H_2O). Hydrogen atoms contain, as all atoms, protons and neutrons. If the number of at least one of either protons or neutrons is odd then a property called Spin can be considered. You can think about the spin as a small magnet, and the magnet has its interactions with any external magnetic field (B_0) [29,34].

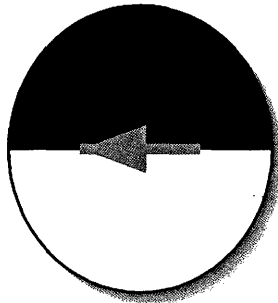


Figure 1.1: Single spin as a small magnet with its poles.

1.1 External Static Magnetic Field (B_0)

When no external magnetic field is applied to the body, no magnetization can be found. The reason returns to the fact that spins will be oriented randomly and they will hit each other. This kind of process results in magnetic moment cancellation (magnetization = 0) Fig. 1.2. However, applying an external magnetic field in the Z-direction (conventionally) forces spins to align parallel

or anti-parallel to the direction of the external magnetic field, Fig. 1.3. If the spin has low energy, then it will align parallel to the B_0 direction (preferred way). In contrast, if it has high energy then its alignment will be anti-parallel [3]. Parallel or anti-parallel does not mean that spins will exactly line up with the direction of B_0 but precess around B_0 Fig. 1.4. The rate of this precession around B_0 is defined as the Larmor Frequency ω_0 [29].

$$\omega_0 = \gamma B_0 \tag{1.1}$$

where γ is the gyromagnetic rate which is unique for each type of atom. Eq.(1.1) states that the strength of B_0 has strong effect on the Larmor Frequency of the spins.

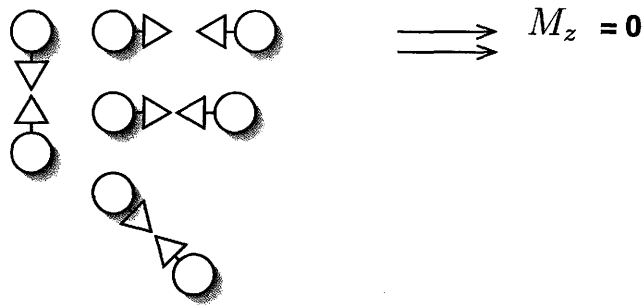


Figure 1.2: Spin random movement in the absence of an external magnetic field. Spin magnetic moments cancel each other and $M_z = 0$.

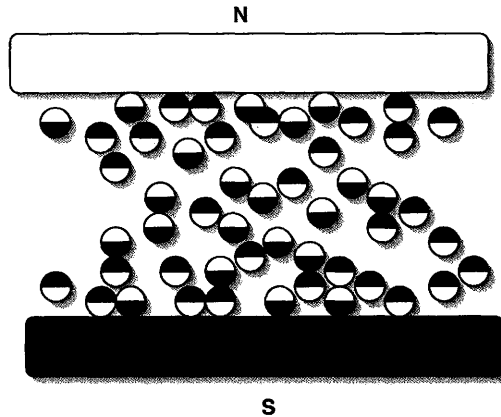


Figure 1.3: Spin alignment in the presence of external magnetic field.

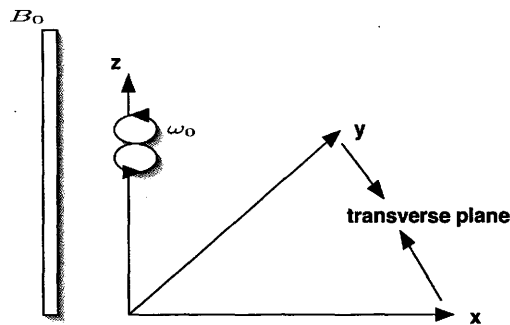


Figure 1.4: Spins are subjected to the external magnetic field and they precess around B_0 at a certain frequency ω_0 .

1.2 Radio Frequency (B_1)

As mentioned before, turning the external magnetic field B_0 on forces the spins to align with one of two possible ways (parallel or anti-parallel). Consequently, longitudinal magnetization is produced (M_z). This magnetization decays to the equilibrium state due to no change occurring to the spin energy levels. This case continues and no measurable signal occurs unless applying Radio Frequency energy (RF) parallel to the transverse plane. RF magnet (B_1) is a small amount compared to the external magnetic field (B_0). The effect of applying 90° RF (by convention) to the equilibrium system, is that a lot of

spins absorb energy from RF and then jump from the low-energy level (parallel to B_0) to the high-energy level (anti-parallel). This process is known as spin excitation, which means that the magnetization (M_z) is flipped by 90° RF into the transverse plane (XY-direction) and spins begin their precession (in phase) at the Larmor Frequency (ω_0) in the transverse plane around (B_0) [1,29,34].

1.3 Relaxation

The spin excitation process indicates a difference between the two energy levels. If we keep applying RF energy at the Larmor Frequency in the transverse plane for an unlimited amount of time, spins in the low-energy level absorb enough energy to arrive at the equalization state. No difference can be seen between the two energy levels. This phenomenon is known as saturation, where the magnetization in the transverse plane decreases in time until reaching zero. Unacceptable cases occur when no measured signal can be observed. We end up with the fact that spins are required to release their absorbed energy and return to the original values. This process is known as Relaxation. Two independent relaxation times are recognized T_1 and T_2 [1,9].

1.3.1 Longitudinal Relaxation Time (T_1)

As spins absorb energy from Radio Frequency (RF), they rotate the longitudinal magnetization (M_z) into the transverse plane. As a result M_z disappears and spins begin their precession at the Larmor Frequency in the transverse plane around B_0 . When the RF is turned off, spins transfer their absorbed energy to their surrounding area. As time goes on, absorbed longitudinal magnetization M_z returns. The time required for spins to return to the equilibrium is called longitudinal relaxation time, T_1 relaxation time or spin-lattice relaxation time (spins give up their energy to the surroundings) [29].

1.3.2 Transverse Relaxation Time (T_2)

Transverse magnetization is exhibited when the excited spins precess in phase at the same Larmor Frequency in the transverse plane. After the elapsed time, spins lose their absorbed energy which is transferred to a nearby spin. This causes a small difference in the spin's Larmor Frequency. Consequently, spins begin to precess out of phase (dephase) and a loss of the transverse magnetization is observed. The time required for this process is called transverse relaxation, T_2 relaxation time or spin-spin relaxation time [6,29].

1.4 Linear Magnetic Field

Linearly varying magnetic fields are added to the system to introduce spatial encoding. This additional magnet changes the static external magnetic field (B_0) into an inhomogeneous field. In the presence of only the static magnetic field (B_0), all spins within the body encounter the same magnetic influence and then all spins precess at the same Larmor Frequency Eq.(1.1). As a result, one RF pulse would excite the whole body making it difficult to distinguish between the measured signals according to their origins within the body. Placing a magnetic coil parallel to the static magnetic field (B_0) would be sufficient to generate a gradient magnetic field (g). Gradients are represented as wedges. Their strength at one side is stronger than the other, which means for example that the total magnetic field ($B_0 + g$) on the patient's head exceeds that on the patient's feet [29,34]. The Larmor equation has to be expanded [1, 26] as

$$\omega_i = \gamma(B_0 + gr_i) \quad (1.2)$$

where

- ω_i is the Larmor Frequency of the spin located at r_i .
- g is the gradient amplitude and direction.

Let us, for example, take the gradient applied in Z-direction g_z , then the Larmor equation is

$$\omega_i = \gamma(B_0 + g_z z_i) \quad (1.3)$$

which means that the Larmor Frequency varies from one spin to another based on their position along the Z-direction (the gradient applied direction).

1.5 Spatial Encoding

In this section, we introduce a brief review on spatial encoding. This process can be divided into three main steps [3,6,29]:

1. Slice selection.
2. Phase encoding.
3. Frequency encoding.

1.5.1 Slice Selection

Switching the gradient causes variation in the strength of the applied magnetic field. The spins of each slice within the body precess at some Larmor Frequency that differs from the Larmor Frequency of another slice. Consequently to investigate a certain slice in the body we have to adjust the frequency of the RF pulse to be approximately equal to the Larmor Frequency of the spins within the desired slice. The given example in section 1.4 selects a slice perpendicular to the Z-dimension because the gradient is applied in the Z-direction [3].

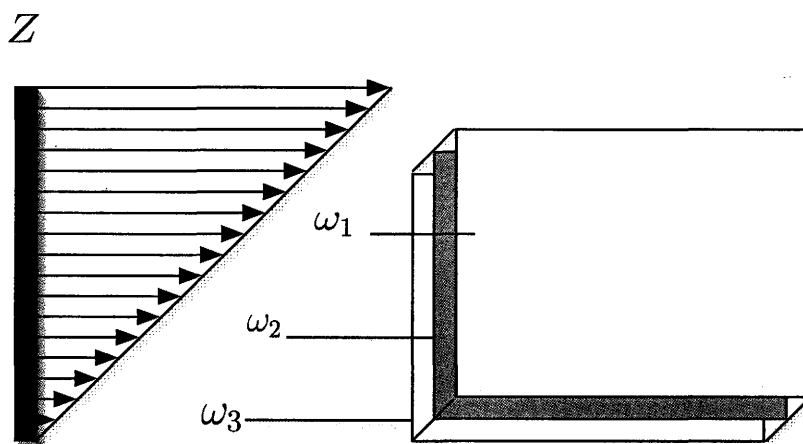


Figure 1.5: Choosing a certain slice via slice selection process.

1.5.2 Phase Encoding

The second step in spatial encoding is phase encoding, which is used to encode one dimension within an image. This task can be done by applying a phase-encoding gradient after excitation in the Y-direction. The strength of the phase encoding gradient at one end is stronger than the other end, and it is intermediate between the two ends. The total magnetic field ($B_0 + g_y y$) is then different for each spin because of the spin's position. Consequently, Larmor Frequency varies from one spin to another. When the phase-encoding gradient is turned off, Larmor Frequencies go back to the uniform case, but the phase change remains. Spins located at the same line (row) within the investigated slice have the same phase, while those located in different lines within the slice have different phases [3,29]. So the slice can be identified by the phase variation, as it shown in Fig. 1.6.

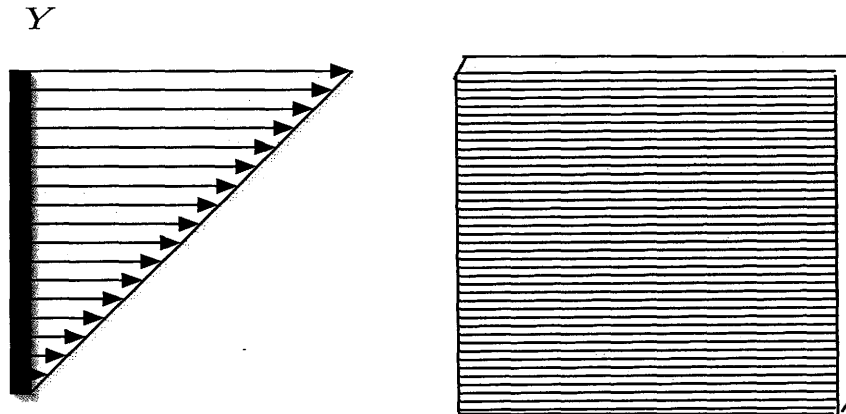


Figure 1.6: Identifying each row within the selected slice via phase encoding process.

1.5.3 Frequency Encoding

During signal measurement, the frequency encoding gradient is applied in the X-direction, which is used to encode the second dimension within an image. Applying this gradient results in variation in the signal frequencies along its direction, as it is shown in Fig. 1.7. Consequently, each column in the slice can be identified by its unique frequency [9,34].

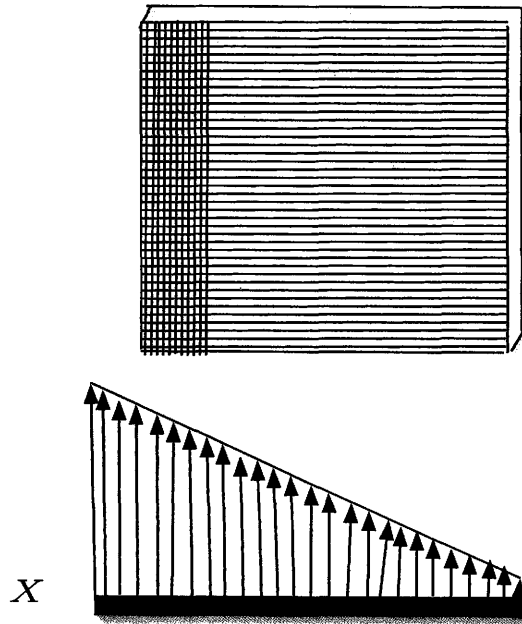


Figure 1.7: Identifying each column within the selected slice via frequency encoding process

Chapter 2

Diffusion Tensor Imaging

“Diffusion is one of several transport processes that occur in nature.”[10]. Fluid molecules such as water, move randomly via their thermal energy from one part of space to another. This type of movement is known as Brownian motion and is measured by a constant called the Self-Diffusion Coefficient (D).

2.1 Free Diffusion

Free diffusion can be described by Fick’s First Law and understood by the following simple example [5]. If we let a drop of ink fall in a certain volume of water, we will see the random spread of the ink molecules through the entire volume of water. Ink molecules move from the high concentration region to the low concentration region, which explains the meaning of the minus sign in the Fick’s First Law, (2.1).

$$J = -D\nabla C \tag{2.1}$$

where

- J is the molecular flux.
- D is the diffusion coefficient.
- ∇C is the molecular concentration gradient.

Consider the equation of the conservation of mass [1,7], which is defined by

$$\partial C / \partial t = -\nabla \cdot J \tag{2.2}$$

And then combine the two previous equations as follows:

$$\partial C / \partial t = -\nabla \cdot J = \nabla \cdot (D\nabla C) \tag{2.3}$$

The result of the combination is known as the diffusion equation.

If we consider the spin displacement concept, which is defined as:

$$R = r - r_0 \quad (2.4)$$

where

- r_0 is the position of the spin at time 0, and
- r is the position of the spin at time τ .

Then the process of diffusion in the equilibrium systems (isotropic system) can be viewed in terms of spin displacement by Einstein's equation,

$$D' = 1/6\tau \langle R^T R \rangle . \quad (2.5)$$

Where $\langle \dots \rangle$ is the average over the spin ensemble. For the non-equilibrium systems (anisotropic system), the Einstein equation easily generalizes to the diffusion tensor equation,

$$D = 1/6\tau \langle RR^T \rangle . \quad (2.6)$$

2.2 Restricted Diffusion

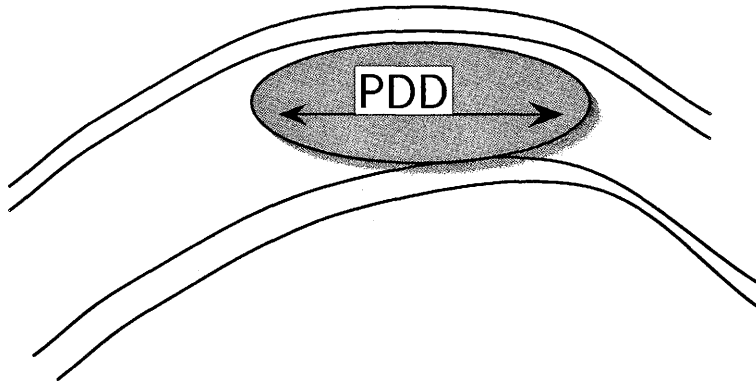


Figure 2.1: Restricted diffusion.

The movement of water molecules is not always free where it might be restricted by boundaries. In this case water molecules will be reflected after encountering these boundaries. The best example of the restricted diffusion is the molecular diffusion of water within biological tissues, where water molecules encounter physical barriers such as membranes [11,12]. Water molecules move through the cell membranes at a slower rate than they diffuse along the membrane. It is impossible to measure the diffusivity within the cell membrane, because the volume is not significant, but measurements can be made on a group of cells which are contained by the voxel.

2.3 Isotropy and Anisotropy

Measuring diffusion within tissues depends on whether the diffusion is affected by the direction of the applied magnetic gradient field or not. In some parts of the body, such as brain gray matter (GM), the applied magnetic gradient field has no affect on the measured apparent diffusion coefficient (ADC). Experimentally the diffusivity in this case is equal in all directions (isotropic diffusion), a single scalar is sufficient for describing the ADC. On the other hand there are some parts of the body such as brain white matter (WM) that are affected by the applied magnetic gradient field. The diffusivity is unequal through the directions (anisotropic diffusion) so a single scalar is not sufficient for the measurement. Because of this, the (3×3) Symmetric Tensor is used to measure the diffusivity for each voxel [2,7].

$$D = \begin{pmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{pmatrix}$$

2.4 Diffusion Weighted-Images and Diffusion Tensor

As mentioned before, measuring diffusion in the body in the anisotropic case can be done by using a tensor for each voxel, because water molecules diffuse randomly in several directions. The measured diffusion here is called Diffusion Tensor (DT). DT measurement requires at least 6 diffusion weighted images, in addition to one non-weighted image. The method that is used to obtain these weighted images is called Pulsed Gradient Spin Echo (PGSE). We would like to

explain some details on the strategy of PGSE, which is based on the Stejskal-Tanner sequence [33]. Fig. 2.2 shows the PGSE sequence, which begins by applying a 90^0 RF pulse. Spins are flipped to the transverse plane and start their rotation in the plane with phase 0. This is followed by a pulsed gradient of duration δ , which introduces a difference in the rate of precession as a function of the position. After this pulse the phase is

$$\Theta_1 = \gamma g \delta r \quad (2.7)$$

where

- r is the spin position during the first gradient pulse,
- γ is the proton gyromagnetic ratio,
- g is the applied field gradient, and
- δ is the gradient duration.

The rotation of the spins rapidly goes out of phase. A 180^0 RF pulse is used to refocus the magnetization, causing the spins to rotate back into phase Θ_2 different from Θ_1 since the spins have moved to new positions due to the diffusion.

$$\Theta_2 = \gamma g \delta r' \quad (2.8)$$

where r' is the spin position during the second gradient pulse. The difference between Θ_1 and Θ_2 defines a concept known as Phase Difference or Molecular Spin Displacement

$$\delta(\Theta) = \gamma g \delta (r - r') \quad (2.9)$$

This equation leads to two different cases:

1. $r = r'$ means the spins remain at the same positions during the first and second pulsed gradient (no diffusion). This indicates that there is no signal loss (no signal attenuation).
2. $r \neq r'$ indicates that the spins have moved to different positions during the second pulsed gradient due to the Brownian motion during time Δ , therefore signal loss (signal attenuation) is an expected result.

The signal attenuation is the ratio of the diffusion weighted signal to the unweighted signal.

$$\frac{S}{S_0} = e^{-bD} \quad (2.10)$$

where

$$b = \gamma^2 g^2 \delta^2 (\Delta - \delta/3)$$

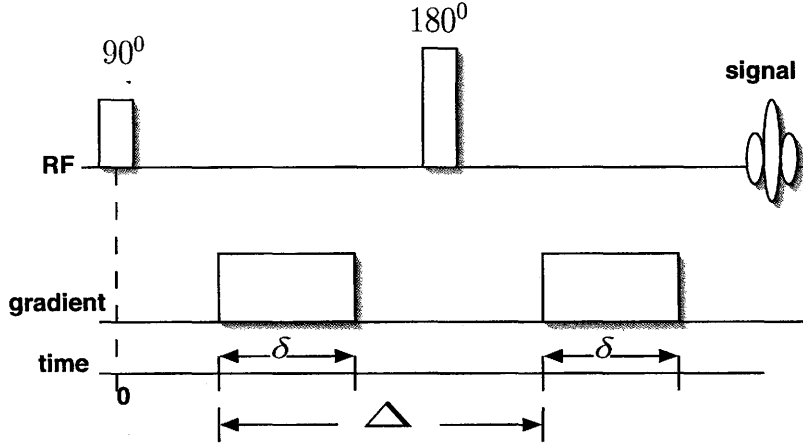


Figure 2.2: The Stejskal-Tanner sequence.

- b is a diffusion weight.
- S_0 is the signal intensity in the absence of a diffusion gradient (non-weighted image).
- S is the signal intensity in the presence of a diffusion gradient (weighted image).

For the case of anisotropic diffusion, Eq.(2.10) has to be written in a more general form.

$$S_k = S_0 e^{-\gamma^2 \delta^2 (\Delta - \delta/3) g_k^T D g_k} \quad (2.11)$$

where

$$k = 1, 2, \dots, n,$$

is the image number.

In our problem, we form the model just in the X-direction and Y-direction, and we consider the fibre to be in a certain pattern such as the circle shown in Fig. 2.3. We applied different gradients and the obtained diffusion weighted images are depicted in Fig. 2.4. When the gradient magnetic field is applied parallel to the direction of the fibre, the diffusivity is large. In contrast, if the gradient field is applied perpendicular to the fibre direction, then the diffusivity is small.

The above fact we just mentioned clearly explains the difference between the weighted images. For example, in the first image (S_1) in Fig. 2.4, the applied gradient is

$$g_1 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{pmatrix}$$

which states that, the gradient is applied in the X-direction. The diffusivity is large in the fibre part that is parallel to the X- axis, and small in the fibre part that is perpendicular to the X-axis. Then no change in the measured signals in the perpendicular fibre part, this explains why the fibre is hidden in the perpendicular direction in S_1 in Fig. 2.4. Note the third image (S_3) is totally white, due to the direction of the applied gradient.

$$g_3 = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

The gradient is applied in the Z-direction, where we do not have a fibre in our model. Also in the fourth image (S_4), our gradient is turned on in two directions.

$$g_4 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$$

X-direction \longrightarrow and Y-direction \uparrow . The sum vector is: $\longrightarrow \uparrow = \nearrow$ Then the anti-parallel part of the fibre disappears in image S_4 of Fig. 2.4.

Note that there are restrictions on the gradients sufficient to distinguish all fibres or tensors. Being linearly independent is not sufficient, as we will show in a 2D example, with two fibre directions not distinguishable by two linearly independent gradients.

Consider the following equation for the diffusion tensor corresponding to an ideal fibre,

$$g_k^T (F F^T) g_k, \quad (2.12)$$

which will be discussed fully in the section (3.1.2). The signal attenuation is given by:

$$\frac{S_k}{S_0} = e^{-b g_k^T (F F^T) g_k} \quad (2.13)$$

If $b = 1$ then

$$\frac{S_k}{S_0} = e^{-g_k^T (F F^T) g_k}. \quad (2.14)$$

Now consider two fibre directions:

$$A = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, B = \begin{pmatrix} 1 \\ -1 \end{pmatrix},$$

and *linearly independent* applied gradients,

$$g_1 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{pmatrix}, g_2 = \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}.$$

The attenuation is shown in the table

gradient	fibre direction	$e^{-g_k^T (FF^T) g_k}$
g_1	A	0.6065
g_2	A	0.6065
g_1	B	0.6065
g_2	B	0.6065

The last column shows that all attenuation are equal, which makes it impossible to distinguish between the tensors. Whereas, adding one more gradient such as

$$g_4 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$$

distinguishes the two fibre directions, as seen in the following table

g	F	$e^{-g_k^T (FF^T) g_k}$
g_1	A	0.6065
g_2	A	0.6065
g_4	A	0.1353
g_1	B	0.6065
g_2	B	0.6065
g_4	B	1

where g_4 gives a different attenuation.

2.5 Diffusion Tensor Eigensystem

The diffusion tensor can be rewritten as a 3×3 diagonal tensor matrix A in the right basis:

$$D = PAP^{-1} \tag{2.15}$$

where

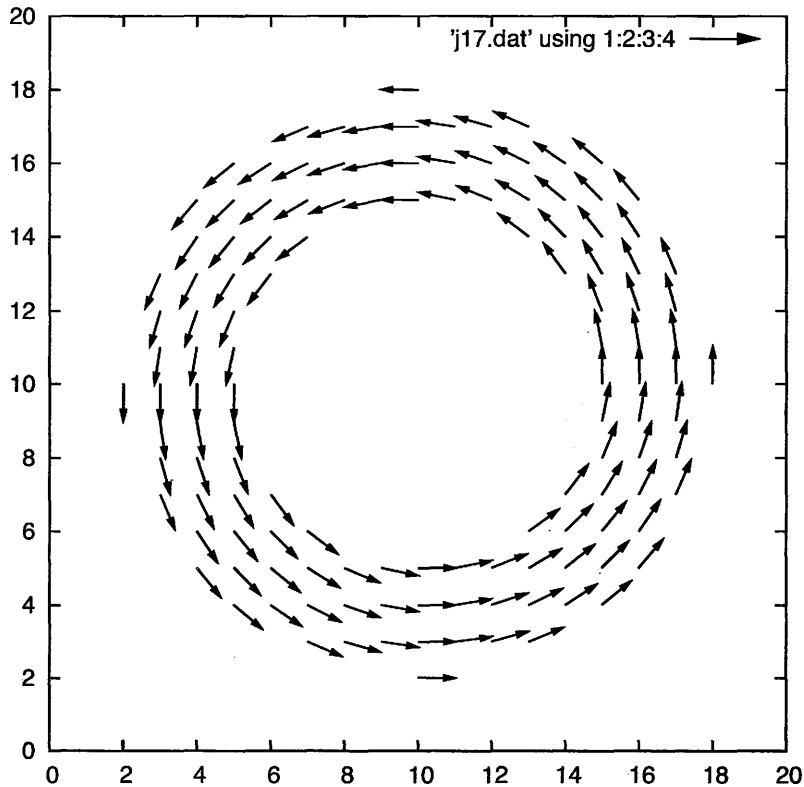


Figure 2.3: Graphical representation of fibre direction.

- $P = (e_1, e_2, e_3)$ is a matrix of column vectors representing independent eigenvectors, and
- $A = \text{diag}(\lambda_1, \lambda_2, \lambda_3)$ which simply represents the corresponding eigenvalues ($\lambda_1 \geq \lambda_2 \geq \lambda_3$).

Fig. 2.5 shows that e_1 which corresponds to λ_1 is the Principle Diffusion Direction (PDD).

2.6 Fibre Tracking

Fibre tracking or tractography is an application of diffusion tensor imaging (DTI). This field is concerned about reconstructing fibres in the anisotropic

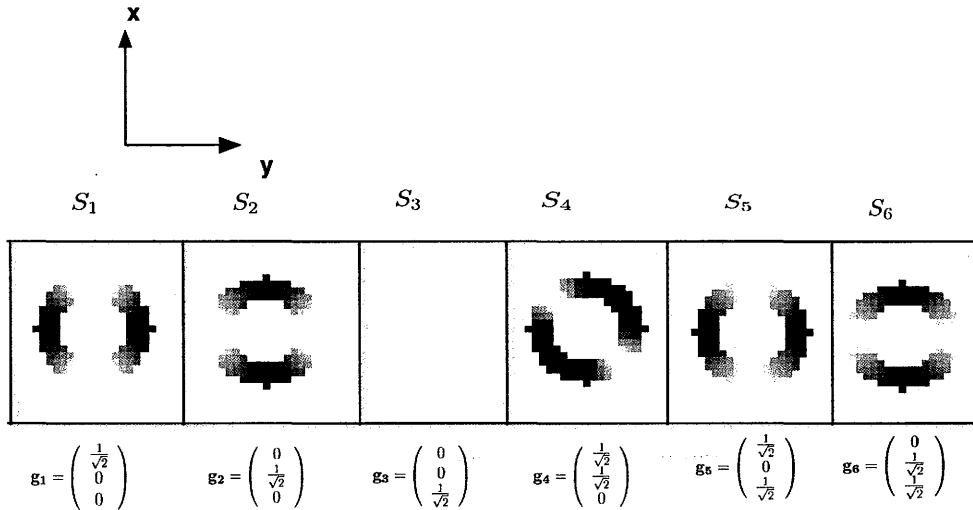


Figure 2.4: Weighted-images with corresponding gradients magnetic field.

environment such as nerves and brain white matter (WM).

Many tractography methods have already been presented in the literature [21, 22, 23], which basically vary in the way that they use the local information of the tensors. But they are all using the same key idea, in that they are using the PDD to reconstruct the fibre directions.

The user chooses a point called seed point to use its PDD to follow the fibre direction Fig. 2.6. Tracing the fibre within a certain voxel continues until the voxel's edge is encountered. Then same process is repeated with the next voxel. Conventionally, in fibre tracking color is used to display fibre direction Fig. 2.7:

1. The fibres in the X-direction are colored by red.
2. The fibres in the Y-direction are colored by green.
3. The fibres in the Z-direction are colored by blue.

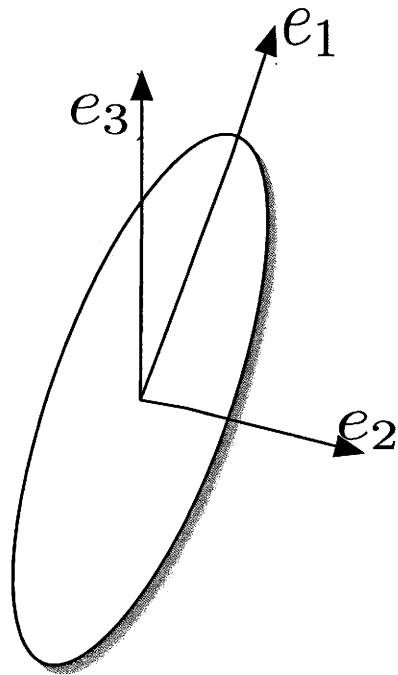


Figure 2.5: Tensor eigensystem.

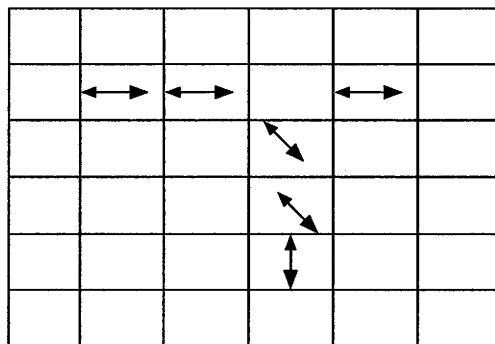


Figure 2.6: Simple interpretation for the Seed Point Method.

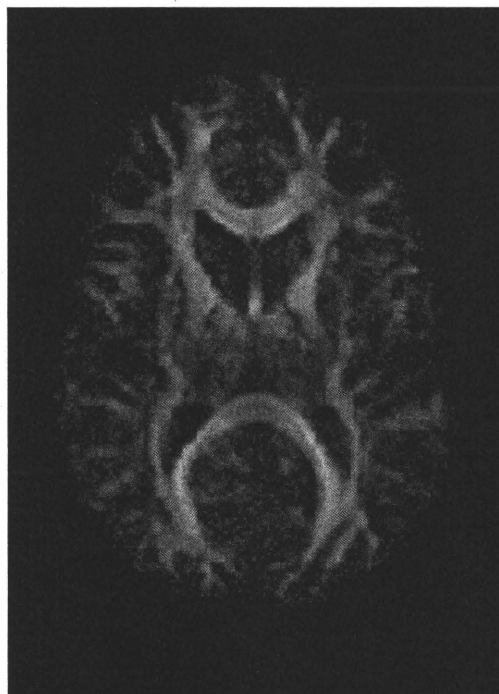


Figure 2.7: Reconstructed fibre direction images using MRI-DTI.

Chapter 3

Diffusion Tensor Estimation and Regularization

3.1 Diffusion Tensor Estimation

The Stejskal-Tanner equation plays the pivotal role in almost all DT estimation methods where it relates two types of information:

1. Diffusivity of the tissue.
2. Sensitizing gradient (both the strength and direction).

$$S_k = S_0 e^{-b' g_k^T D g_k} \quad (3.1)$$

where

$$b' = \gamma^2 \delta^2 (\Delta - \delta/3)$$

Note that b' differs from b , defined by Eq.(2.10), in that the factor g is not included in b' .

3.1.1 Least Squares Estimation Method

The likelihood of model values based on measurements containing normally distributed errors is maximized by minimizing the difference between model predictions and measurements. This explains the widespread use of least-squares minimization, and why it is appropriate in our model.

In the DT model the objective function is defined as:

$$\forall i = 1, ..Nr \quad \forall j = 1, ..Nc$$

$$\min_{D(i,j)} \sum_{k=1}^n \|S_{k(i,j)} - S_{0(i,j)} e^{-bg_k^T D(i,j) g_k}\|^2 \quad (3.2)$$

where Nr and Nc are the number of the rows and columns in the matrix (image) respectively.

3.1.2 Our Method

In our method we pay attention to the fact that in the presence of MRI, most water molecules within the biological tissues diffuse in the same direction as the fibre tissues. Since the principle diffusion direction (PDD) is the important target, we attempt to obtain with its corresponding eigenvalue (λ_1), we assume that the diffusivity in perpendicular directions to the PDD (λ_2 and λ_3) are equal.

$$\lambda_2 = \lambda_3 < \lambda_1$$

Based on this assumption, and when we have that the voxels have only a single fibre orientation, the diffusion tensor can be written as:

$$D = dI + FF^T \quad (3.3)$$

where

- d is a scalar.
- I is the identity matrix.
- F is a vector representing the PDD (fibre direction) and the increased diffusivity in that direction.

$$F = \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix}$$

Note that in our numerical phantom, we set the Z-component to zero, which indicates no fibre in the Z-direction. Eq.(3.3) can be rewritten as follows:

$$D = \begin{pmatrix} d & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & d \end{pmatrix} + \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix}^T$$

where d now represents the diffusivity amount in all directions perpendicular to the PDD. In the concept of the eigensystem, the PDD points to the direction

of the fibre (F), then in our model F is the eigenvector e_1 and its corresponding eigenvalue is:

$$\lambda_1 = d + F^T F$$

$$\begin{aligned} \text{where } (d + FF^T)F &= (dF + |F|^2 F) \\ &= (d + |F|^2)F \end{aligned}$$

So, $\lambda_F = (d + |F|^2)$ is the diffusivity amount in the direction of F eigenvector. The other eigenvalue can be detected as follows:

Let us say the eigenvector is W.

$$\begin{aligned} (d + FF^T)W &= (dW + 0) \quad \text{F and W are orthogonal} \\ &= dW \end{aligned}$$

Thus, the diffusivity amount in the directions perpendicular to fibre direction are $\lambda_W = d$.

The objective function of our model is defined as:

$$\forall i = 1, \dots, Nr \quad \forall j = 1, \dots, Nc$$

$$\min_{d(i,j), F(i,j)} \sum_{k=1}^n \|S_{k(i,j)} - S_{0(i,j)} e^{-bg_k^T d(i,j)g_k - (F_{(i,j)}^T g_k)^2}\|^2 \quad (3.4)$$

where b as mentioned before is:

$$b = \gamma^2 \delta^2 (\Delta - \delta/3)$$

Nr and Nc are the number of the rows and columns in the matrix (image) respectively.

3.2 Diffusion Tensor Regularization

Diffusion imaging involves image attenuation, which reduces the ratio of signal to noise. Regularization can be used to reduce the amount of noise visible in images and other representations of diffusivity.

3.2.1 Tikhonov Regularization

It is the most commonly used method for ill-posed problems [16, 22]. In the case of a nonlinear function, $f(x) = y$, where

- x and y are unknown variables.

- y_{meas} is an available noisy measurement.

we are concerned with solving the following equation:

$$f(x) = y_{meas} \quad (3.5)$$

Then the basic Tikhonov formula in this case is given by :

$$\min_{x \in \text{dom}(f)} \|f(x) - y_{meas}\|^2 + \alpha \|x - \bar{x}\|^2 \quad (3.6)$$

where

- \bar{x} is an initial guess for the solution x .
- $\|x - \bar{x}\|$ is the penalty term.
- α is the regularization parameter.

The objective function with the penalty term becomes:

$$\min_{d, F} \sum_{k=1}^n \sum_{i, j} \|S_{k(i, j)} - S_{0(i, j)} e^{-bg_k^T d_{(i, j)} g_k - (F_{(i, j)}^T * g_k)^2}\|^2 + R(d, F) \quad (3.7)$$

where $R(d, F)$ is a regularizer to be chosen later.

The regularization process at each pixel basically depends on the number of neighbour-pixels that are taken in account. These neighbour-pixels can be chosen by considering a certain shape that includes the required neighbour-pixels. In our model we picked the pixels by using a square in two different experiments, as is shown in Fig. 3.1.

In the first experiment we picked the four neighbour-pixels (white boxes) as shown in Fig. 3.1 (a), thus the objective function was extended to be:

$$\forall i = 2, ..Nr - 1 \quad \forall j = 2, ..Nc - 1$$

$$\begin{aligned} \min_V & \sum_{k=1}^n (\|S_{k(i-1, j)} - S_{0(i-1, j)} e^{-bg_k^T d_{(i-1, j)} g_k - (F_{(i-1, j)}^T * g_k)^2}\|^2) \\ & + \sum_{k=1}^n (\|S_{k(i, j-1)} - S_{0(i, j-1)} e^{-bg_k^T d_{(i, j-1)} g_k - (F_{(i, j-1)}^T * g_k)^2}\|^2) \\ & + \sum_{k=1}^n (\|S_{k(i, j)} - S_{0(i, j)} e^{-bg_k^T d_{(i, j)} g_k - (F_{(i, j)}^T * g_k)^2}\|^2) \end{aligned}$$

$$\begin{aligned}
& + \sum_{k=1}^n (\|S_{k(i,j+1)} - S_{0(i,j+1)} e^{-bg_k^T d_{(i,j+1)} g_k - (F_{(i,j+1)}^T g_k)^2}\|^2) \\
& + \sum_{k=1}^n (\|S_{k(i+1,j)} - S_{0(i+1,j)} e^{-bg_k^T d_{(i+1,j)} g_k - (F_{(i+1,j)}^T g_k)^2}\|^2) \\
& + \alpha(\|d_{(i,j)} - d_{(i-1,j)}\|^2 + \|d_{(i,j)} - d_{(i,j-1)}\|^2 + \|d_{(i,j)} - d_{(i,j+1)}\|^2 \\
& + \|d_{(i,j)} - d_{(i+1,j)}\|^2 + \|F_{(i,j)} - F_{(i-1,j)}\|^2 + \|F_{(i,j)} - F_{(i,j-1)}\|^2 \\
& + \|F_{(i,j)} - F_{(i,j+1)}\|^2 + \|F_{(i,j)} - F_{(i+1,j)}\|^2), \tag{3.8}
\end{aligned}$$

where

$$V = \{d(i + \bar{i}, j + \bar{j}), F(i + \bar{i}, j + \bar{j}) | (\bar{i}, \bar{j}) \in (0, 0), (0, 1), (0, -1), (1, 0), (-1, 0)\}.$$

The previous equation states that when solving this equation for each pixel we store the results of the pixel of index (i, j) in Fig. 3.1(a) and ignore the rest of the results which belong to pixels $(i-1, j), (i, j-1), (i, j+1), (i+1, j)$.

In the second experiment we used the eight neighbour-pixels (gray boxes) Fig. 3.1(b), thus the objective function became:

$$\forall i = 1, \dots, Nr - 2 \quad \forall j = 1, \dots, Nc - 2$$

$$\begin{aligned}
\min_V \sum_{\bar{i}=0}^2 \sum_{\bar{j}=0}^2 \sum_{k=1}^n & (\|S_{k(i+\bar{i}, j+\bar{j})} - S_{0(i+\bar{i}, j+\bar{j})} e^{-bg_k^T d_{(i+\bar{i}, j+\bar{j})} g_k - (F_{(i+\bar{i}, j+\bar{j})}^T g_k)^2}\|^2) \\
& + \alpha \left(\sum_{\bar{i}=0}^2 \sum_{\bar{j}=0}^1 (d_{(i+\bar{i}, j+\bar{j})} - d_{(i+\bar{i}, j+\bar{j}+1)})^2 \right. \\
& + \sum_{\bar{j}=0}^2 \sum_{\bar{i}=0}^1 (d_{(i+\bar{i}, j+\bar{j})} - d_{(i+\bar{i}+1, j+\bar{j})})^2 \\
& + \sum_{\bar{i}=0}^2 \sum_{\bar{j}=0}^1 (F_{(i+\bar{i}, j+\bar{j})} - F_{(i+\bar{i}, j+\bar{j}+1)})^2 \\
& \left. + \sum_{\bar{j}=0}^2 \sum_{\bar{i}=0}^1 (F_{(i+\bar{i}, j+\bar{j})} - F_{(i+\bar{i}+1, j+\bar{j})})^2 \right) \tag{3.9}
\end{aligned}$$

$$\begin{aligned}
V = \{ & d(i + \bar{i}, j + \bar{j}), F(i + \bar{i}, j + \bar{j}) \\
& | (\bar{i}, \bar{j}) \in (0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)\}.
\end{aligned}$$

In this experiment we follow these steps:

```

Initialize array initalObjective to 1000000.
for  $i \in \{1, \dots, rows - 2\}$  do
  for  $j \in \{1, \dots, cols - 2\}$  do
    Solve the optimization problem Eq.(3.9) which returns the objective and
    the solution.
    for  $(i', j') \in \text{neighbours of } (i, j)$  do
      if  $objective < initalObjective(i', j')$  then
         $initalObjective(i', j') \leftarrow objective$ 
         $d(i', j') \leftarrow d \text{ in } solution.$ 
         $F(i', j') \leftarrow F \text{ in } solution.$ 
      end if
    end for
  end for
end for

```

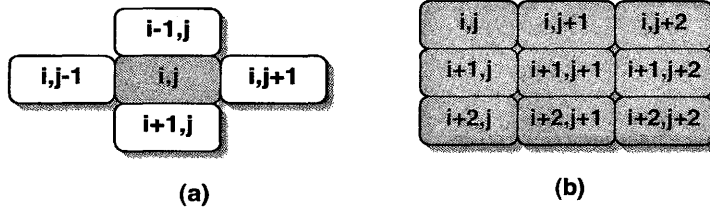


Figure 3.1: Graphical representation of the neighbour-pixels for a certain pixel.

Chapter 4

Numerical and Experimental Results

In this chapter we present the results of regularizing our unconstrained non-linear problem, (3.7). We compare two regularization experiments:

- centred, four neighbour pixel experiment, and
- best-of-nine neighbourhood experiment,

using both numerical difference with numerical phantom and visual results.

This chapter is divided into two parts, in the first one we show the numerical results for the circle fibre orientations we introduced in Fig. 2.2, by simulating the diffusion-weighted images, with white noise added, and reconstructing the model variables (d and F) using optimization with and without regularization. These results show that regularization reduces noise without altering the structure of the phantom.

In the second part we present the results of a single slice cross section of a leg. These results are preliminary, and suggest a faithful reproduction of the calf muscles, but require more work (at higher resolution, comparable to other methods) to be considered a validation of the method.

4.1 Experiment Environment

1. Machine Name: Power Mac G5.
2. CPU Speed: 2GHz.
3. RAM: 2GB.

4.2 Numerical Results

4.2.1 Solvers

It was a good step in the beginning to solve the problem using some extra solver's package that can be installed to work with the Matlab environment. Since the validation of this model depends on the quality of the optimizer used, we benchmarked both the built-in Matlab solver and the commercial package, Tomlab. [30] The Tomlab tool contains many solvers and you can find out the one suitable for your problem conditions. We first tried UnSolve solver, this one is for unconstrained non-linear optimization problems. With using an initial guess $X_0 = [0 \ 0 \ 0 \ 0]$ we realized that the solver got stuck many times while the program was executing and a warning message appeared that the results were inaccurate. At that point we tried to use multiple initial guesses, ($X_{0a} = [0 \ 0 \ 0 \ 0]$ and $X_{0b} = [0 \ 0.1 \ 0.1 \ 0]$), to solve the problem, which means each pixel in our (20×20) image was solved two times and this clearly results in a wasted time. The results were good as you see in Fig. 4.1(a). Although the estimated results were contaminated by noise, the structure of the numerical phantom was clearly recognizable. Fig. 4.1(a) shows the fibre orientations which were plotted in 2D, *i.e.*, of the three components of the fibre variable,

$$F = \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix}$$

we plotted (f_x, f_y) .

NpSol is another solver we tried to use with our problem, this solver is for constrained non-linear problems. The results were similar to the ones we got by using the UnSolve solver.

During our work with Tomlab we encountered two problems:

1. Tomlab solvers spent a long time (approximately 10 hours) solving our problem even though the image size was small (20×20) . The poor performance may be the result of the non-linearity of our problem even though the problem is small.
2. The license of the Tomlab package is free for a period of time, and our department does not have a license.

For these reasons, we tried to solve the problem using Matlab's built-in functions. The function we used is `Fminsearch()`.

The `Fminsearch` function (solver) is used to minimize the unconstrained non-linear problems using an initial guess X_0

```
[X, Fval] = Fminsearch (@MyFun, X0, Options)
```

where X_0 is the initial guess which can be a scalar or a vector, X is the estimated solution, and $Fval$ is the objective function at the solution X . $Options$ is a property to modify some parameter's values such as the maximum number of the iteration ($MaxIter$).

```
Options=optimset ('MatIter', 1000000)
```

The objective function (optimization problem) is defined in a Matlab function as:

```
Function MyFun (X0, alpha)  
MyFun= X0^2 + alpha
```

The evaluation of the objective function at the solution can be returned in the parameter $Fval$. Using this solver to only estimate the PDD costs approximately 4 hours, so people can use this algorithm without a need to buy any license.

4.2.2 Initial Guess X_0

Since our problem is non-convex, starting with different initial points may result in different solutions. As we have mentioned in the previous section, we used multiple initial guesses with Tomlab solvers to get better results Fig. 4.1(a). When we switched to work with $Fminsearch$ solver we used the same multiple initial guesses. As you can see in Fig. 4.1(b), the results were much better in terms of the circle pattern where Fig. 4.1(b) is closer to the correct model Fig. 2.2 than Fig. 4.1(a).

In terms of the running time there was a really big difference where Tomlab solvers took approximately 10 hours to solve our problem while $Fminsearch$ took about 4 hours. We also tried to determine what the results are by using a single initial guess ($X_0 = [0 \ 0 \ 0 \ 0]$) with $Fminsearch$ solver, and in fact there was no difference between the results when using single or multiple initial guesses as you can compare between Fig. 4.1(b) and Fig. 4.2. So based on that we considered using single initial guess ($X_0 = [0 \ 0 \ 0 \ 0]$) to solve the problem.

4.2.3 Choice of Gradient Encodings

In the process of data acquisition we found by experiments that an acceptable magnitude gradients for our data are the following:

$$g_1 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{pmatrix} g_2 = \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} g_3 = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} g_4 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} g_5 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} g_6 = \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

4.2.4 Noise

As was mentioned before, in practice the process of data acquisition results in distributed noise. Consequently, we used a Matlab function called `wgn` to add a white gaussian noise to our simulated diffusion weighted images.

`wgn (row,col,n)`

where

- *row* is the number of rows in the matrix (image).
- *col* is the number of columns in the matrix.
- *n* is the noise amount that is added to the reconstructed images.

In our experiment we used $n = 0.01$ and because it was not possible to see the difference between the various experiments results we increased the amount of noise to $n = 20$ as shown in Fig. 4.6.

4.2.5 Regularized Fiber Direction Images

In the process of removing the errors from the estimated results using the Tikhonov technique, you can choose any shape containing the desired neighbour-pixels for the specific pixel such as the shapes in Fig. 4.3. To evaluate the effect on the quality of the results, we tried two neighbourhoods:

1. using the 4 neighbour-pixels for the certain pixel Fig. 3.1 (a).
2. using the 8 neighbour-pixels for the certain pixel Fig. 3.1 (b)

The results of the first experiment on the fiber direction is depicted in Fig. 4.4(a). The errors were reduced a little bit by this regularization in some parts of the fiber direction image as you can see in Fig. 4.4(a). A large number of errors were around the fiber orientations precisely above and inside the circle pattern. Taking more neighbour-pixels into account can increase the amount that errors are reduced, as is shown in Fig. 4.4(b).

To further compare the two regularization experiments, we used the L_2 norm of the difference between the correct model (circle pattern) and the calculated results from both experiments. The L_2 norm is lower when the result is better and this is what we observed in the second regularization experiment in our model. The L_2 norm of the first regularization experiment (4 neighbour-pixels) was 11.7368, whereas was 10.8647 with the second regularization experiment (8 neighbour-pixels).

Looking at images is important in identify weaknesses in specific situations, with boundaries in different orientations being the most likely to pose difficulties. The comparison can be enhanced by using the error visual presentation of the two experiments. We calculated the difference in the two experiments between the estimated results and the real fiber direction, and it is clear that the results from the second experiment Fig. 4.5(b) are better than the first one Fig. 4.5(a). We compare between the two figures based on the arrows magnitude, when the arrow magnitude is small then the regularized results are close to the correct model. Note that the numbers in Fig. 4.5(b) point to some parts where the noise is reduced more than that in Fig. 4.5(a). We also look for a correlation between the errors and the structure of the phantom ring, which would indicate a loss of image structure.

4.3 Experimental Results

After making sure that our model works well with the simulated data, the second step in our project was checking the validity of our model with real data. The data we obtained from the Imaging Research Centre in Hamilton is a single slice cross-section of a leg. Fig. 4.8 shows the unweighted image (no sensitizing gradient magnetic field). Whereas Fig. 4.9 shows 6 weighted images and and their respective sensitivity gradient directions.

Fig. 4.10 shows the estimated results when no regularization method is applied. These results were good so far and we were eager to see what the regularization experiments can add to these partial results.

In the first regularization experiment (4 neighbour-pixels) we noted some error reduction Fig. 4.11(a), which in turn gave us a hope to get better

results by applying the second regularization experiment (8 neighbour-pixels) as shown in Fig. 4.11(b) . We observed that taking more neighbour pixels in account increases the effectiveness of the regularization in removing noise. Trying bigger α (regularization parameter) makes the noise penalization more robust. With the two previous experiments we used $\alpha= 1000$ after that we made a big jump by choosing $\alpha= 10000$ and applied the second regularization experiment (8 neighbour-pixels). The image Fig. 4.12 (c) ($\alpha= 10000$) is smoother than that with $\alpha= 1000$ Fig. 4.12 (b).

The effect of regularization can be more readily seen by plotting the Z-component in all the previous experiments. With no regularization process it is really hard to recognize anything in Fig. 4.13 (a), where the image is completely obscured by noise. In the first regularization experiment ($\alpha= 1000$) Fig. 4.13 (b) we observed that amount of the noise was removed compared with Fig. 4.13 (a), but it is hard to tell what is in the image. Some parts of the muscles appeared in Fig. 4.13(c) as a result of the second regularization experiment ($\alpha= 1000$). In Fig. 4.13 (d) the image is much better, with the muscle apparent and largely free of noise.

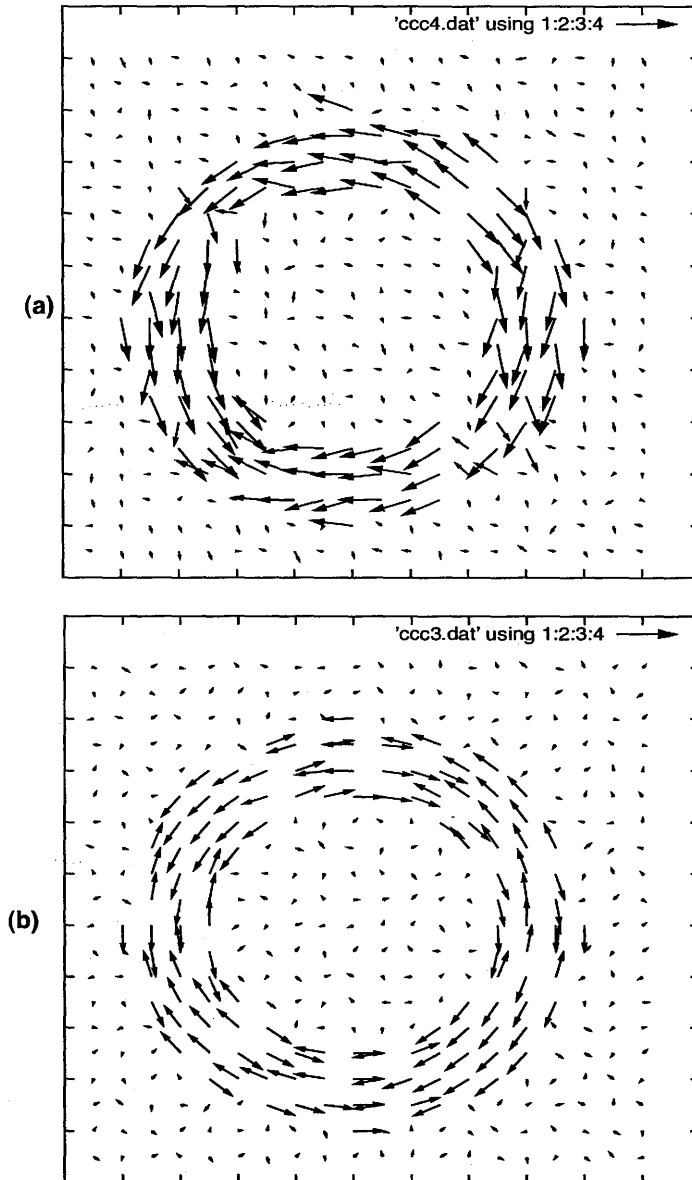


Figure 4.1: Graphical representation of fibre direction results from using using two initial guesses $[0 \ 0 \ 0 \ 0]$ and $[0 \ 0.1 \ 0.1 \ 0]$: (a) by Tomlab solvers, (b) Fminsearch solver

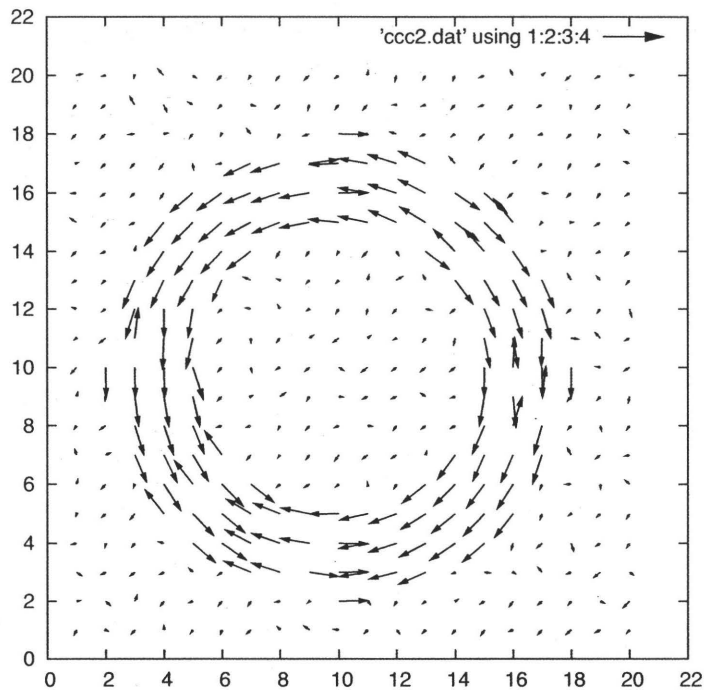


Figure 4.2: Graphical representation of fibre direction results from Fminsearch with an initial guess $[0\ 0\ 0\ 0]$

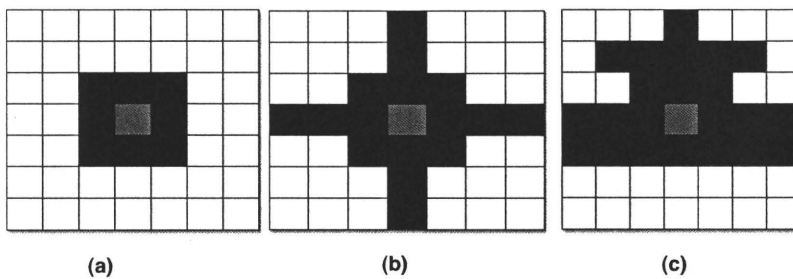


Figure 4.3: Created shapes each of which surround a certain pixel (gray color) by different neighbour-pixels (black color).

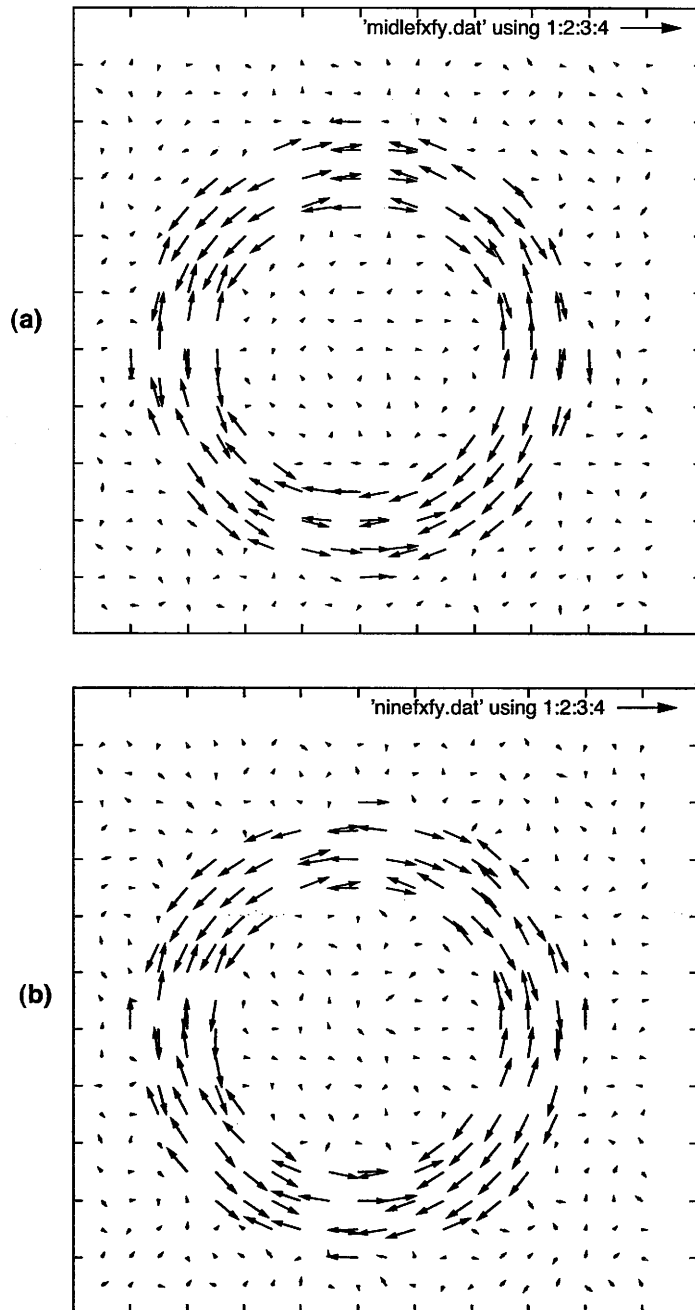


Figure 4.4: Graphical representation of regularized fibre direction (a) centred, four neighbour pixel experiment, (b) best-of-nine neighbourhood experiment.

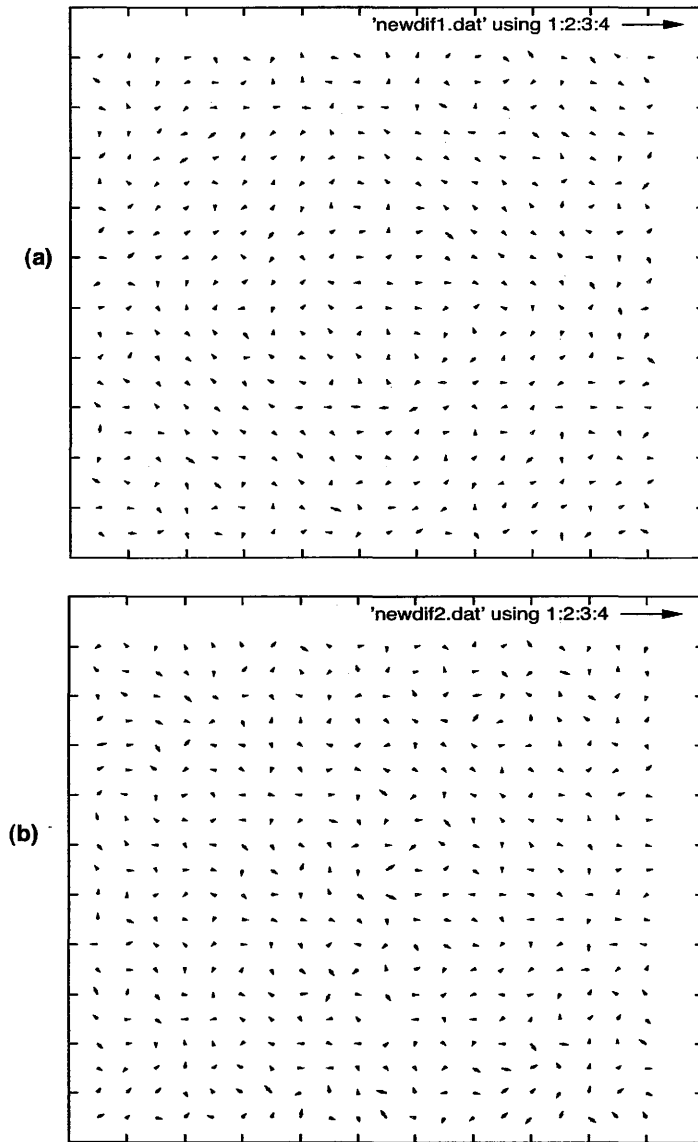


Figure 4.5: Graphical representation of the calculated differences between the regularized fibre direction and the correct model which results from (a) centred, four neighbour pixel experiment, (b) best-of-nine neighbourhood experiment . Small arrows indicate small errors, without variation across the image or correlation with the ring structure.

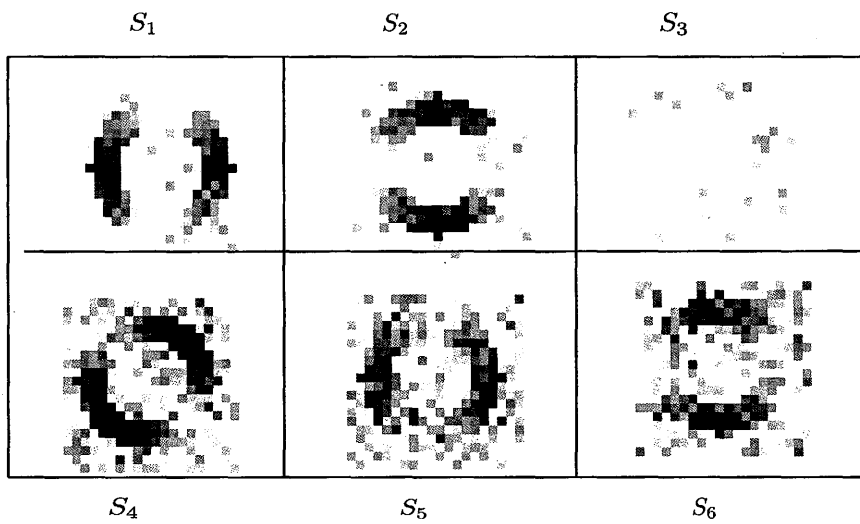


Figure 4.6: Diffusion weighted images obtained by adding more noise (20 instead of .01).

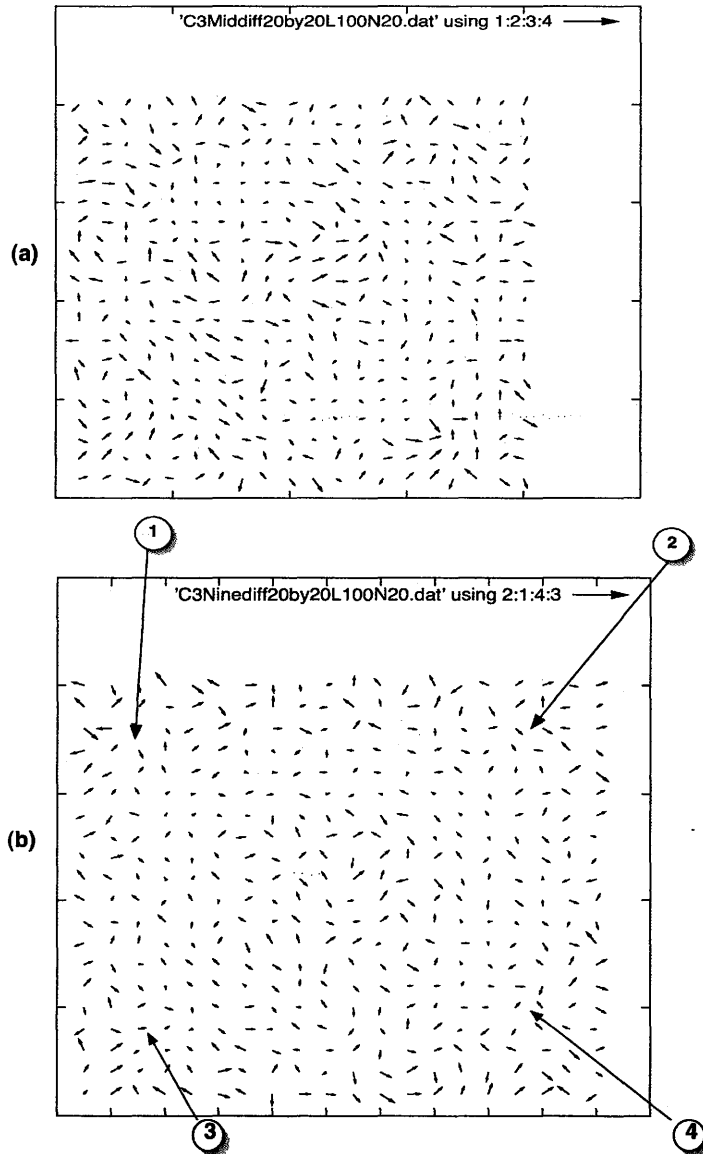


Figure 4.7: Graphical representation shows the calculated difference between the correct model and the results of (a) centred, four neighbour pixel experiment (long arrows specially around edges indicate big errors) and (b) best-of-nine neighbourhood experiment (error reductions can be seen specially around the edges (1), (2), (3), and (4)).

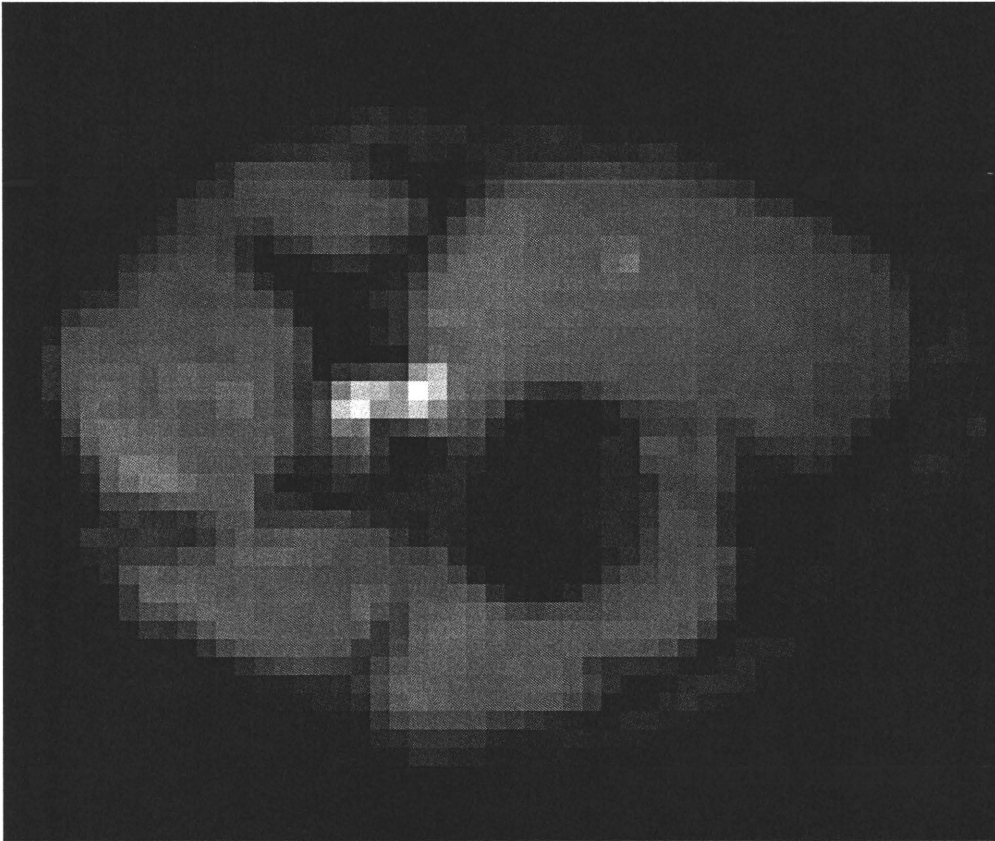


Figure 4.8: T_2 weighted image through the thigh of a healthy volunteer. Here an EPI diffusion encoding sequence was used with $b = 0$.

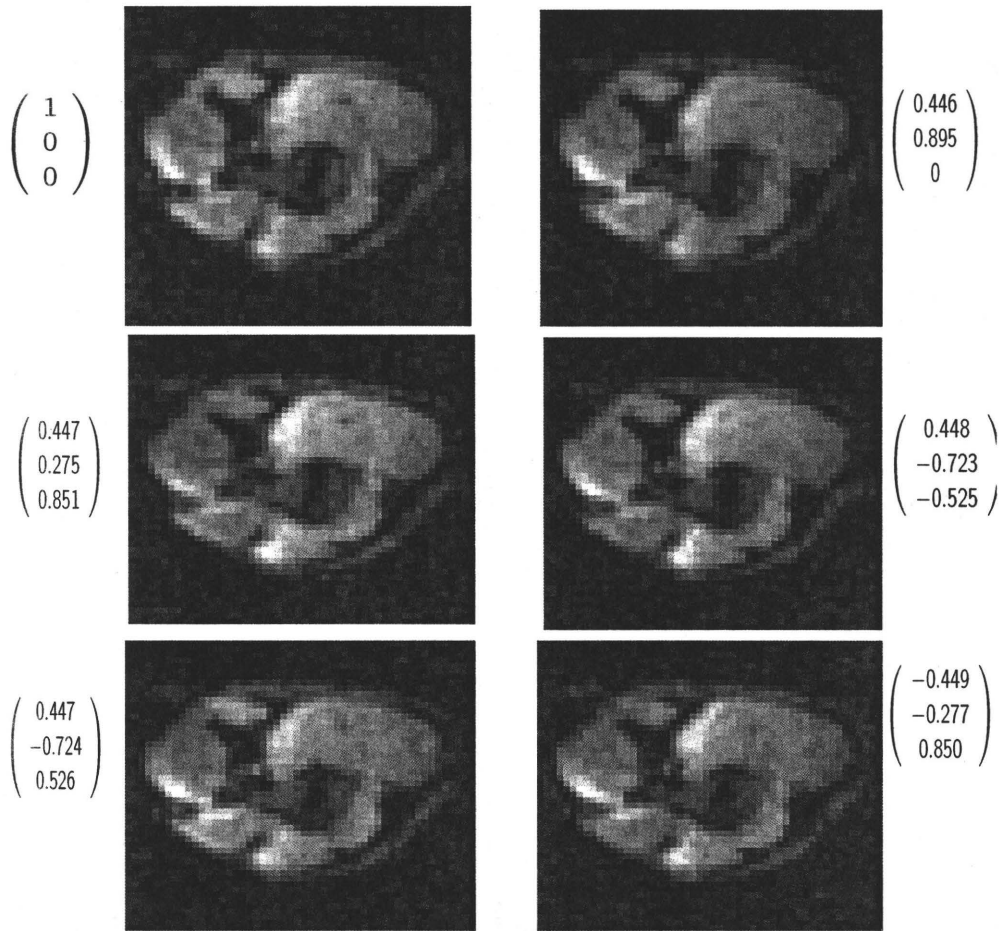


Figure 4.9: The diffusion weighted images with the used gradient at each case.

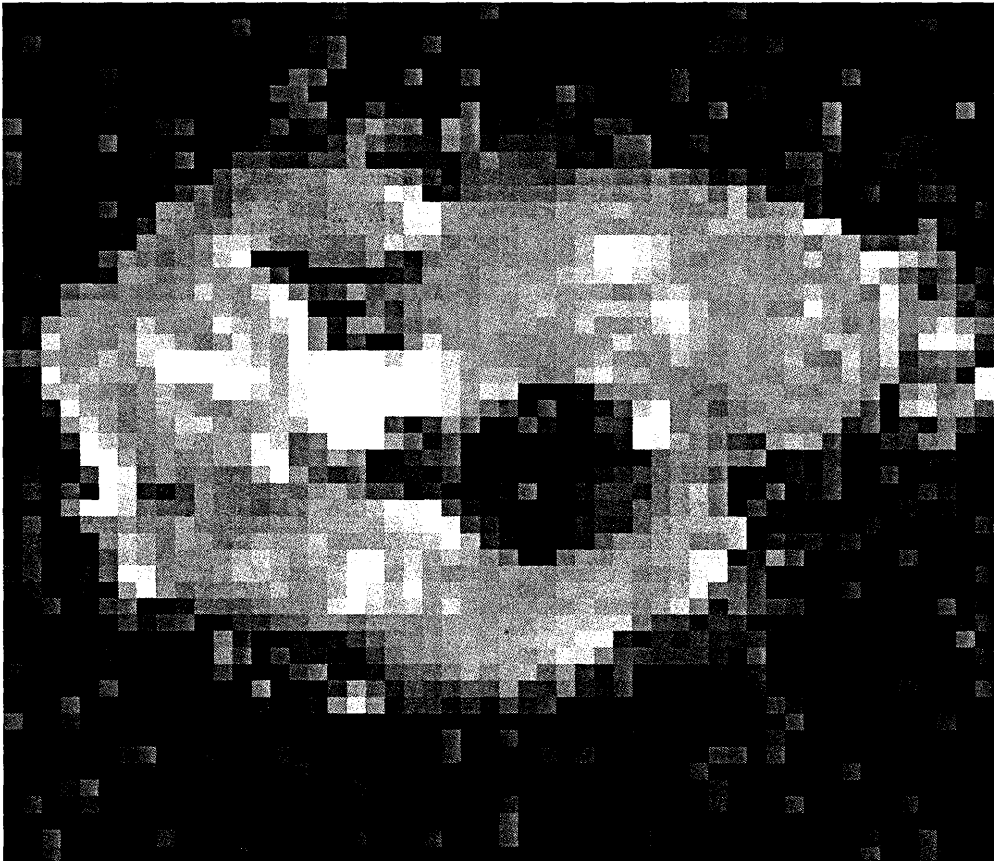


Figure 4.10: Reconstructed image (anisotropic component) without regularization.

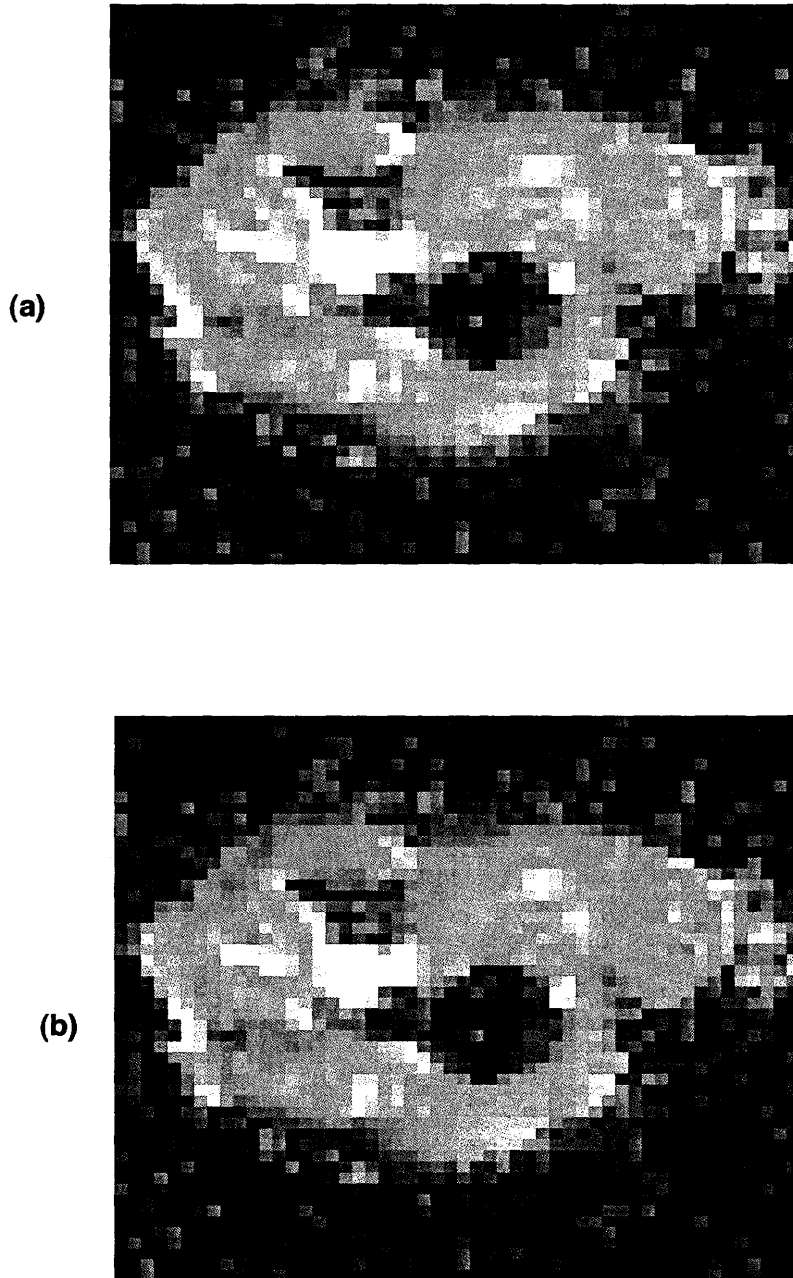


Figure 4.11: Reconstructed images when $\alpha=1000$, (a) 4 neighbour-pixels regularization method (b) 8 neighbour-pixels regularization method.

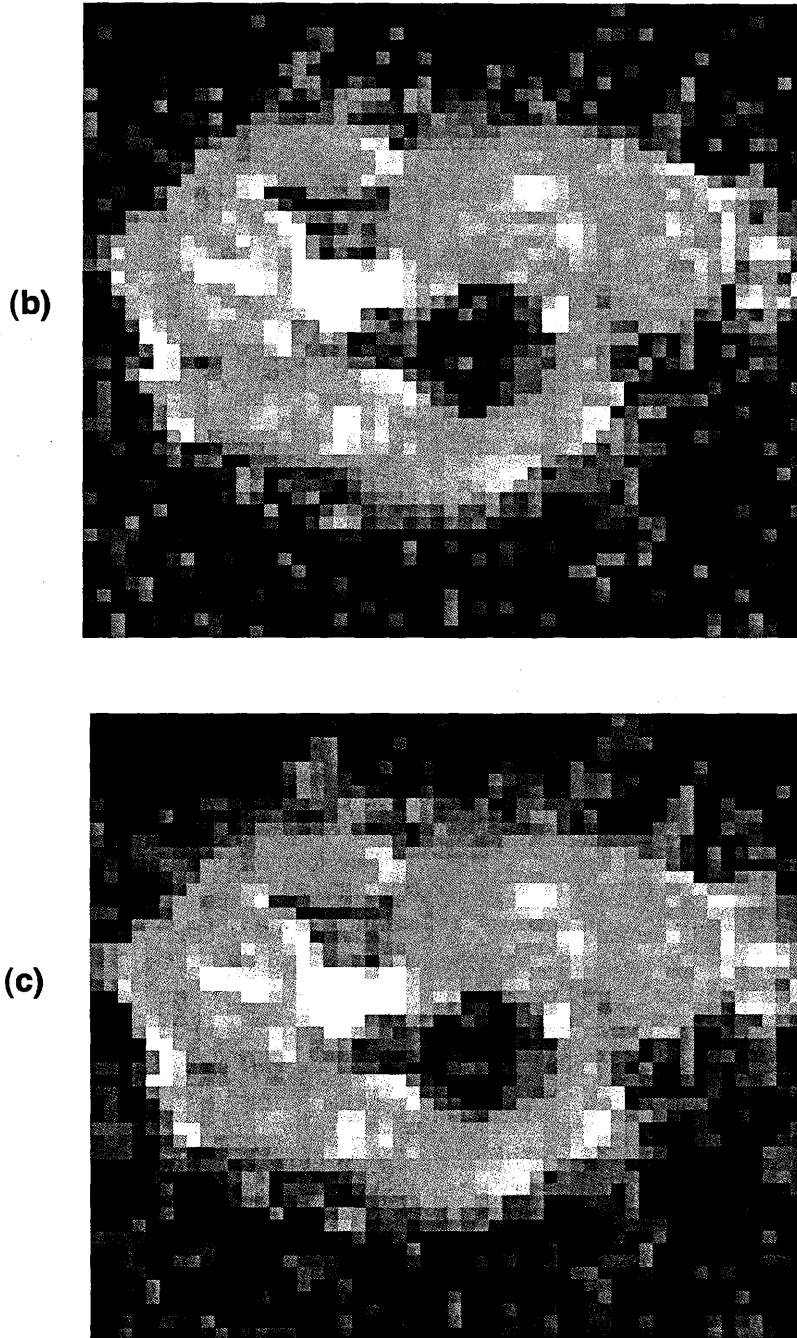


Figure 4.12: Reconstructed images (b) 8 neighbour-pixels regularization method ($\alpha=1000$) (c) 8 neighbour-pixels regularization method $\alpha=10000$.

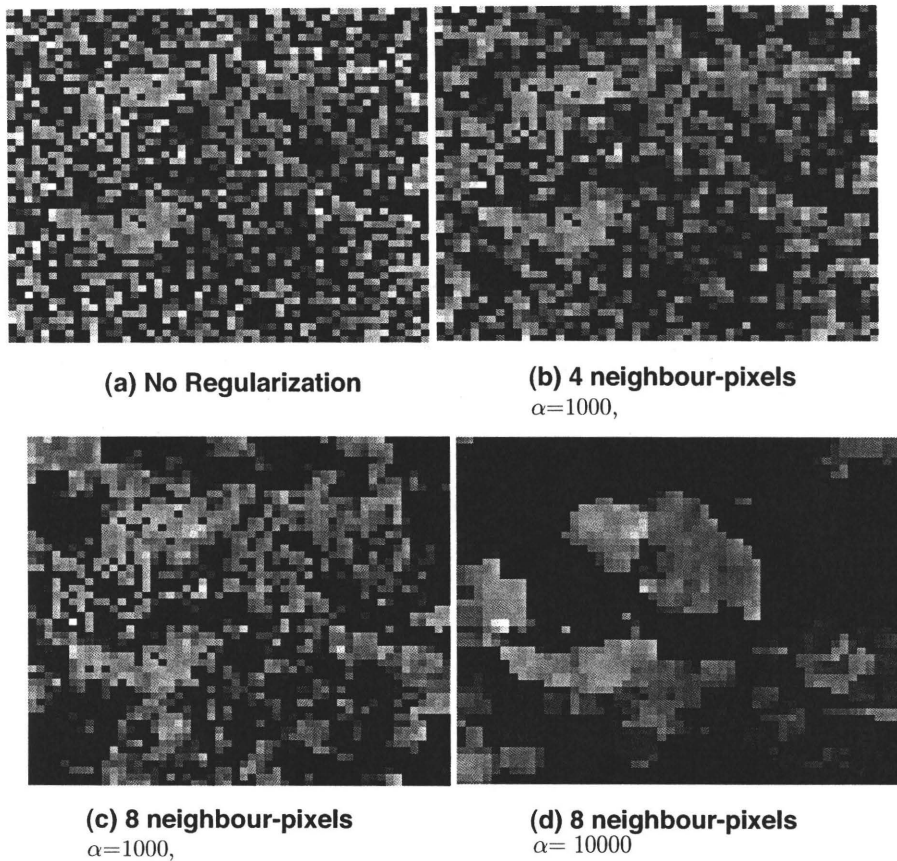


Figure 4.13: Reconstructed images F_z -components

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, we have presented a novel method for regularizing the principle diffusion direction (PDD). This method is a new attempt in this field, building on a restricted DT model. The restriction reduces the number of variables to be estimated from six to four as explained in chapter 3, and exposes the PDD as a model variable, making it possible to apply regularization in a single optimization problem.

Both numerical simulation and preliminary experimental results showed that using regularization causes a visible error reduction and smoothes the estimates, without hiding the structure. Numerical results also showed that using more neighbouring-pixels in the regularization is better.

5.2 Future Work

Our method showed good results, which we think can be further improved by:

1. Increasing the number of acquired diffusion weighted images.
2. Using higher resolution and higher regularization parameter α .
3. Using more than one slice at a time will probably make as much difference as going from 4 to 8 neighbour-pixels.
4. Using solvers that take advantage of exact derivatives, to improve the accuracy of the estimated results.

5. Trying to reduce the implementation time by using another programming language such as C.
6. Adding multiple-fibre orientations to the model.

Appendix: MATLAB Codes

1. Main Function RDTM.m

```
%-----  
% File name is RDTM.m  
% This is the main function in the implementation code.  
% It calls several functions that contain the objective  
% functions in several cases.  
%-----  
% Turn the clock on to calculate the implementation time  
tic  
clear all;  
% Define a global variables  
global s1 s2 s3 s4 s5 s6 s0 i j lam rn cn ;  
global g1x g1y g1z g2x g2y g2z g3x g3y g3z g4x ;  
global g4y g4z g5x g5y g5z g6x g6y g6z;  
% Define image's size  
rn=20;  
cn=20;  
  
%Define S0 to be the non-weighted image  
s0=250*ones(rn,cn);  
  
% Define a scalar d that represents the diffusion amount  
% within directions that are anti-parallel to fiber tissues.  
d=1;  
  
% Reconstruct the fiber orientations to be as a circle pattern  
% (the correct model)  
for i=1:rn  
    for j=1:cn  
        rad = sqrt((i-10)*(i-10)+(j-10)*(j-10));
```

```
    if ((rad > 8) || (rad < 5))
        fx((rn*i-rn)+j)=0;
        fy((rn*i-rn)+j)=0;
        fz((rn*i-rn)+j)=0;
    % Save the X and Y coordinates of the fiber directions
    % F=(fx; fy; fz)^T in a file.
        c=[i j fx((rn*i-rn)+j) fy((rn*i-rn)+j)];
        save xyfiber.dat c -ascii -append;
    else
        fx((rn*i-rn)+j)=-(j-(10))/rad;
        fy((rn*i-rn)+j)= (i-(10))/rad;
        fz((rn*i-rn)+j)=0;
        c=[i j fx((rn*i-rn)+j) fy((rn*i-rn)+j)];
        save xyfiber.dat c -ascii -append;
    end % if
end % j loop
end % i loop

% Save the correct model values.
fxold=fx;
fyold=fy;

%-----
%Reconstructing the weighted images.
%With each image we use different magnetic gradient.
%-----

%*****image 1 *****%
%g1 is a 3D vector represents the first gradient field.
%g1=[1/sqrt(2); 0; 0];
g1x=1/sqrt(2);
g1y=0;
g1z=0;

for i=1:(rn*cn)
    pow1(i)=-d*(g1x^2+g1y^2+g1z^2)-(fx(i)*g1x+fy(i)*g1y+...
    fz(i)* g1z)^2;
end

for i=1:rn
```

```
        for j=1:cn
            s1(i,j)=abs(s0(i,j))*exp(pow1((rn*i-cn)+j));
        end
    end
%*****image 2 *****%
% g2 is a 3D vector represents the second gradient field.
%g2=[0; 1/sqrt(2); 0];
g2x= 0;
g2y=1/sqrt(2);
g2z=0;
for i=1:(rn*cn)
    pow2(i)=-d*(g2x^2+g2y^2+g2z^2)-(fx(i)*g2x+fy(i)*g2y+...
    fz(i)*g2z)^2;
end

for i=1:rn
    for j=1:cn
        s2(i,j)=abs(s0(i,j))*exp(pow2((rn*i-rn)+j));
    end
end
%*****image 3 *****%
% g3 is a 3D vector represents the third gradient field.
%g3=[0; 0; 1/sqrt(2)];
g3x= 0;
g3y=0;
g3z=1/sqrt(2);
for i=1:(rn*cn)
    pow3(i)=-d*(g3x^2+g3y^2+g3z^2)-(fx(i)*g3x+fy(i)*g3y+...
    fz(i)*g3z)^2;
end

for i=1:rn
    for j=1:cn
        s3(i,j)=abs(s0(i,j))*exp(pow3((rn*i-cn)+j));
    end
end

%*****image 4 *****%
% g4 is a 3D vector represents the fourth gradient field.
% g4=[ 1/sqrt(2); 1/sqrt(2); 0];
```

```
g4x= 1/sqrt(2);
g4y=1/sqrt(2);
g4z=0;
for i=1:(rn*cn)
    pow4(i)=-d*(g4x^2+g4y^2+g4z^2)-(fx(i)*g4x+fy(i)*g4y+...
    fz(i)*g4z)^2;
end

for i=1:rn
    for j=1:cn
        s4(i,j)=abs(s0(i,j))*exp(pow4((rn*i-rn)+j));
    end
end

%*****image 5 *****%
% g5 is a 3D vector represents the fifth gradient field.
%g5=[ 1/sqrt(2); 0; 1/sqrt(2)];
g5x= 1/sqrt(2);
g5y=0;
g5z=1/sqrt(2);
for i=1:(rn*cn)
    pow5(i)=-d*(g5x^2+g5y^2+g5z^2)-(fx(i)*g5x+fy(i)*g5y+...
    fz(i)*g5z)^2;
end

for i=1:rn
    for j=1:cn
        s5(i,j)=abs(s0(i,j))*exp(pow5((rn*i-rn)+j));
    end
end

%*****image 6 *****%
% g6 is a 3D vector represents the sixth gradient field.
%g6=[ 0;1/sqrt(2); 1/sqrt(2)];
g6x= 0;
g6y=1/sqrt(2);
g6z=1/sqrt(2);
for i=1:(rn*cn)
    pow6(i)=-d*(g6x^2+g6y^2+g6z^2)-(fx(i)*g6x+fy(i)*g6y+...
    fz(i)*g6z)^2;
```

```
end

for i=1:rn
    for j=1:cn
        s6(i,j)=abs(s0(i,j))*exp(pow6((rn*i-rn)+j));
    end
end
%***** CHECKING THE OBJECTIVE FUNCTION *****%

for i=1:rn
    for j=1:cn
        tb(i,j)=(s1(i,j)-abs(s0(i,j))*exp(-d*(g1x^2+g1y^2+...
            g1z^2)-(fx((rn* i-rn)+j)*g1x+fy((rn*i-rn)+j)*...
            g1y+fz((rn* i-rn)+j)*g1z)^2)+(s2(i,j)-...
            abs(s0(i,j))*exp(-d*(g2x^2+g2y^2+g2z^2)-...
            (fx((rn* i-rn)+j)*g2x+ fy((rn*i-rn)+j)*g2y+...
            fz((rn*i-rn)+j)*g2z)^2)+(s3(i,j)-...
            abs(s0(i,j))*exp(-d*(g3x^2+g3y^2g3z^2)-...
            (fx((rn*i-rn)+j)*g3x+ fy((rn*i-rn)+j)*g3y+...
            fz((rn*i-rn)+j)*g3z)^2)+(s4(i,j)-abs(...
            s0(i,j))*exp(-d*(g4x^2+g4y^2+g4z^2)-...
            (fx((rn*i-rn)+j)*g4x+fy((rn*i-rn)+j)*g4y+...
            fz((rn*i-rn)+j)*g4z)^2)+(s(i,j)-abs(s0...
            (i,j))*exp(-d*(g5x^2+g5y^2+g5z^2)-...
            (fx((rn*i-rn)+j)*g5x+fy((rn*i-rn)+j)*g5y+...
            fz((rn*i-rn)+j)*g5z)^2)+(s6(i,j)-abs(...
            s0(i,j))*exp(-d*(g6x^2+ g6y^2+g6z^2)-...
            (fx((rn*i-rn)+j)*g6x+fy((rn*i-rn)+j)*g6y+...
            fz((rn*i-rn)+j)*g6z)^2))^2;
    end
end
end
noise1=0.01;
% Adding Noise to the weighted images.
s1 = s1 + wgn(rn,cn,noise1);
s2 = s2 + wgn(rn,cn,noise1);
s3 = s3 + wgn(rn,cn,noise1);
s4 = s4 + wgn(rn,cn,noise1);
s5 = s5 + wgn(rn,cn,noise1);
s6 = s6 + wgn(rn,cn,noise1);
```

```
%*****  
%*****Diffusion Estimation *****  
%*****  
    for i=1:rn  
        for j=1:cn  
  
% Using the Options property to increase the maximum  
% number of function evaluations.  
options=optimset('MaxFunEvals',10000000000000000,  
'MaxIter',1000000);  
    x0 = [0 0.0 0.0 0]'; % Starting values for the optimization.  
    % using Fminsearch solver to solve the problem.  
[xn,fval]=fminsearch(@fb1,x0,options);  
    obj1(i,j)=fval;  
d((rn*i-rn)+j)=xn(1);  
fx((rn*i-rn)+j)=xn(2);  
fy((rn*i-rn)+j)=xn(3);  
fz((rn*i-rn)+j)=xn(4);  
  
        end  
    end  
    %Scaling the estimated results  
dmax=d(1);  
fxmax=fx(1);  
fymax=fy(1);  
fzmax=fz(1);  
  
for i=1:rn*cn  
    if dmax <d(i)  
        dmax=d(i);  
    end  
    if fxmax <fx(i)  
        fxmax=fx(i);  
    end  
    if fymax <fy(i)  
        fymax=fy(i);  
    end  
    if fzmax <fz(i)  
        fzmax=fz(i);  
    end  
end
```

```

end

for i=1:rn*cn
    d(i)=d(i)/dmax;
    fx(i)=fx(i)/fxmax;
    fy(i)=fy(i)/fymax;
    fz(i)=fz(i)/fzmax;
end

for i=1:rn
    for j=1:cn
        c1=[i j fx((rn*i-rn)+j) fy((rn*i-rn)+j) ];
        save NoRegL300N10.dat c1 -ascii -append;
    end
end

d7=d;
fx7=fx;
fy7=fy;
fz7=fz;

%*****
%***** Regularization*****
%*****
%Give a value to the Regularization parameter.
lam=300;

%*****
%First Regularization experminet
% Using 4 neighbor-pixels for the certain pixel.
%*****
obj2=obj1;

for i=2:(rn-1)
    for j=2:(cn-1)
% Starting values for the optimization.
x_0 = [d7((rn*(i-1)-rn)+j) fx7((rn*(i-1)-rn)+j)
        fy7((rn*(i-1)-rn)+j) fz7((rn*(i-1)-rn)+j)

```

```

        d7((rn*i-rn)+(j-1))    fx7((rn*i-rn)+(j-1))
        fy7((rn*i-rn)+(j-1))  fz7((rn*i-rn)+(j-1))
        d7((rn*i-rn)+j)       fx7((rn*i-rn)+j)
        fy7((rn*i-rn)+j)       fz7((rn*i-rn)+j)
        d7((rn*i-rn)+(j+1))    fx7((rn*i-rn)+(j+1))
        fy7((rn*i-rn)+(j+1))  fz7((rn*i-rn)+(j+1))
        d7((rn*(i+1)-rn)+j)    fx7((rn*(i+1)-rn)+j)
        fy7((rn*(i+1)-rn)+j)  fz7((rn*(i+1)-rn)+j) ]';

[xn,fval]=fminsearch(@fb2,x_0,options);

    d7((rn*i-rn)+j)=xn((9));
    fx7((rn*i-rn)+j)=xn((10));
    fy7((rn*i-rn)+j)=xn((11));
    fz7((rn*i-rn)+j)=xn(12);
    obj2(i,j) =fval;
        end
    end

for i=1:rn
    for j=1:cn
        c4=[i j fx7((rn*i-rn)+j) fy7((rn*i-rn)+j) ];
        save MidL300N10.dat c4 -ascii -append;
    end
end

for i=1:rn
    for j=1:cn
        c4=[i j d7((rn*i-rn)+j) fx7((rn*i-rn)+j)
            fy7((rn*i-rn)+j) fz7((rn*i-rn)+j) obj2(i,j)];
        save MidDataL300N10.dat c4 -ascii -append;
    end
end

%*****
%*****Regularization Test*****
% Calculating the difference between the correct model and
% the regularized results by first experiment.
%*****

```

```

for i=1:rn
    for j=1:cn
        if fx7((rn*i-rn)+j)*fxold((rn*i-rn)+j)+fy7((rn*...
            i-rn)+j)*
            fyold((rn*i-rn)+j)<0
            difx1((rn*i-rn)+j)=fx7((rn*i-rn)+j)+fxold((rn*...
                i-rn)+j);
            dify1((rn*i-rn)+j)=fy7((rn*i-rn)+j)+fyold((rn*...
                i-rn)+j);
        else
            difx1((rn*i-rn)+j)=fx7((rn*i-rn)+j)-fxold((rn*...
                i-rn)+j);
            dify1((rn*i-rn)+j)=fy7((rn*i-rn)+j)-fyold((rn*...
                i-rn)+j);
        end
    end
end

for i=1:rn
    for j=1:cn
        w1=[i j difx1((rn*i-rn)+j) dify1((rn*i-rn)+j) ];
        save MiddiffL300N10.dat w1 -ascii -append;
    end
end

%*****
%Second Regularization experiment.
% Using 8 neighbor-pixels for the certain pixel.
%*****
obj3=1000000;
for i=1:(rn-2)
    for j=1:(cn-2)
% Starting values for the optimization.
x0=[d((rn*(i)-rn)+(j))    fx((rn*(i)-rn)+(j))
    fy((rn*(i)-rn)+(j))    fz((rn*(i)-rn)+(j))
    d((rn*(i)-rn)+(j+1))  fx((rn*(i)-rn)+(j+1))
    fy((rn*(i)-rn)+(j+1))  fz((rn*(i)-rn)+(j+1))
    d((rn*(i)-rn)+(j+2))  fx((rn*(i)-rn)+(j+2))

```

```
fy((rn*(i)-rn)+(j+2))    fz((rn*(i)-rn)+(j+2))
d((rn*(i+1)-rn)+(j))    fx((rn*(i+1)-rn)+(j))
fy((rn*(i+1)-rn)+(j))    fz((rn*(i+1)-rn)+(j))
d((rn*(i+1)-rn)+(j+1))  fx((rn*(i+1)-rn)+(j+1))
fy((rn*(i+1)-rn)+(j+1)) fz((rn*(i+1)-rn)+(j+1))
d((rn*(i+1)-rn)+(j+2))  fx((rn*(i+1)-rn)+(j+2))
fy((rn*(i+1)-rn)+(j+2)) fz((rn*(i+1)-rn)+(j+2))
d((rn*(i+2)-rn)+(j))    fx((rn*(i+2)-rn)+(j))
fy((rn*(i+2)-rn)+(j))  fz((rn*(i+2)-rn)+(j))
d((rn*(i+2)-rn)+(j+1))  fx((rn*(i+2)-rn)+(j+1))
fy((rn*(i+2)-rn)+(j+1)) fz((rn*(i+2)-rn)+(j+1))
d((rn*(i+2)-rn)+(j+2))  fx((rn*(i+2)-rn)+(j+2))
fy((rn*(i+2)-rn)+(j+2)) fz((rn*(i+2)-rn)+(j+2))];

[xn,fval]=fminsearch(@fb4,x_0,options);

if fval < obj3(i,j)
    obj3(i,j) = fval;
    d((rn*(i)-rn)+(j))=xn(1);
    fx((rn*(i)-rn)+(j)) =xn(2);
    fy((rn*(i)-rn)+(j))=xn(3);
    fz((rn*(i)-rn)+(j)) =xn(4);
end

if fval < obj3(i,j+1)
    obj3(i,j+1) = fval;
    d((rn*(i)-rn)+(j+1))=xn(5);
    fx((rn*(i)-rn)+(j+1)) =xn(6);
    fy((rn*(i)-rn)+(j+1)) =xn(7);
    fz((rn*(i)-rn)+(j+1)) =xn(8);
end

if fval < obj3(i,j+2)
    obj3(i,j+2) = fval;
    d((rn*(i)-rn)+(j+2)) =xn(9);
    fx((rn*(i)-rn)+(j+2)) =xn(10);
    fy((rn*(i)-rn)+(j+2)) =xn(11);
    fz((rn*(i)-rn)+(j+2)) =xn(12);
end
```

```
if fval < obj3(i+1,j)
    obj3(i+1,j) = fval;
    d((rn*(i+1)-rn)+(j)) =xn(13);
    fx((rn*(i+1)-rn)+(j)) =xn(14);
    fy((rn*(i+1)-rn)+(j)) =xn(15);
    fz((rn*(i+1)-rn)+(j)) =xn(16);
end

if fval < obj3(i+1,j+1)
    obj3(i+1,j+1) = fval;
    d((rn*(i+1)-rn)+(j+1))=xn(17);
    fx((rn*(i+1)-rn)+(j+1)) =xn(18);
    fy((rn*(i+1)-rn)+(j+1)) =xn(19);
    fz((rn*(i+1)-rn)+(j+1)) =xn(20);
end

if fval < obj3(i+1,j+2)
    obj3(i+1,j+2) = fval;
    d((rn*(i+1)-rn)+(j+2)) =xn(21);
    fx((rn*(i+1)-rn)+(j+2)) =xn(22);
    fy((rn*(i+1)-rn)+(j+2)) =xn(23);
    fz((rn*(i+1)-rn)+(j+2)) =xn(24);
end

if fval < obj3(i+2,j)
    obj3(i+2,j) = fval;
    d((rn*(i+2)-rn)+(j))=xn(25);
    fx((rn*(i+2)-rn)+(j)) =xn(26);
    fy((rn*(i+2)-rn)+(j)) =xn(27);
    fz((rn*(i+2)-rn)+(j)) =xn(28);
end

if fval < obj3(i+2,j+1)
    obj3(i+2,j+1) = fval;
    d((rn*(i+2)-rn)+(j+1)) =xn(29);
    fx((rn*(i+2)-rn)+(j+1)) =xn(30);
    fy((rn*(i+2)-rn)+(j+1)) =xn(31);
    fz((rn*(i+2)-rn)+(j+1)) =xn(32);
end
```

```

    if fval < obj3(i+2,j+2)
        obj3(i+2,j+2) = fval;
        d((rn*(i+2)-rn)+(j+2)) =xn(33);
        fx((rn*(i+2)-rn)+(j+2))=xn(34);
        fy((rn*(i+2)-rn)+(j+2))=xn(35);
        fz((rn*(i+2)-rn)+(j+2)) =xn(36);
    end
    end %for i
end %for j

for i=1:rn
    for j=1:cn
        c5=[i j fx((rn*i-rn)+j) fy((rn*i-rn)+j) ];

        save NinefxfyL300N10.dat c5 -ascii -append;
    end
end

for i=1:rn
    for j=1:cn
        c5=[i j d((rn*i-rn)+j) fx((rn*i-rn)+j)
            fy((rn*i-rn)+j) fz((rn*i-rn)+j)];
        save NineDataL300N10.dat c5 -ascii -append;
    end
end

%*****
%*****Regularization Test*****
% Calculating the difference between the correct model
%and the regularized results by the second experiment
%*****
for i=1:rn
    for j=1:cn
        if fx((rn*i-rn)+j)*fxold((rn*i-rn)+j)+fy((rn*...
            i-rn)+j)*
            fyold((rn*i-rn)+j)<0
            difx2((rn*i-rn)+j)=fx((rn*i-rn)+j)+fxold((rn*...
                i-rn)+j);
            dify2((rn*i-rn)+j)=fy((rn*i-rn)+j)+fyold((rn*...
                i-rn)+j);
        else

```

```
        difx2((rn*i-rn)+j)=fx((rn*i-rn)+j)-fxold((rn*...
            i-rn)+j);
        dify2((rn*i-rn)+j)=fy((rn*i-rn)+j)-fyold((rn*...
            i-rn)+j);
    end
end
end

for i=1:rn
    for j=1:cn
        w1=[i j difx2((cn*i-cn)+j) dify2((cn*i-cn)+j) ];
        save NinediffL300N10.dat w1 -ascii -append;
    end
end

end

%Plotting the weighted images
subplot(3,2,1);imshow(s1,[92 140]);
subplot(3,2,2);imshow(s2,[92 140]);
subplot(3,2,3);imshow(s3,[92 140]);
subplot(3,2,4);imshow(s4,[55 91]);
subplot(3,2,5);imshow(s5,[55 91]);
subplot(3,2,6);imshow(s6,[55 91]);
tim=toc
```

Function Fb1.m

```
function fun = fb1(x)

global s1 s2 s3 s4 s5 s6 s0 i j lam rn cn ;
global g1x g1y g1z g2x g2y g2z g3x g3y g3z g4x ;
global g4y g4z g5x g5y g5z g6x g6y g6z;

fun=(s1(i,j)-abs(s0(i,j))*exp(-x(1)*(g1x^2+g1y^2+g1z^2)-...
(x(2)*g1x+x(3)* g1y+x(4)*g1z)^2)+(s2(i,j)-abs(s0(i,j))*...
exp(-x(1)*(g2x^2+g2y^2+g2z^2)-(x(2)*g2x+x(3)*g2y+x(4)*...
g2z)^2))^2+(s3(i,j)-abs(s0(i,j))*exp(-x(1)*(g3x^2+g3y^2+...
g3z^2)-(x(2)*g3x+x(3)*g3y+x(4)*g3z)^2))^2+(s4(i,j)-...
abs(s0(i,j))*exp(-x(1)*(g4x^2+g4y^2+g4z^2)-(x(2)*g4x+...
x(3)*g4y+x(4)*g4z)^2))^2+(s5(i,j)-abs(s0(i,j))*exp(-x(1)*...
(g5x^2+g5y^2+g5z^2)-(x(2)*g5x+x(3)*g5y+x(4)*g5z)^...

```

$$2))^2+(s_6(i,j)-\text{abs}(s_0(i,j))\cdot\exp(-x(1)\cdot(g_6x^2+g_6y^2+\dots g_6z^2)-(x(2)\cdot g_6x+x(3)\cdot g_6y+x(4)\cdot g_6z)^2))^2;$$

Function Fb2.m

```
function fun = mfb(x)

global s1 s2 s3 s4 s5 s6 s0 i j lam rn cn;
global g1x g1y g1z g2x g2y g2z g3x g3y g3z g4x ;
global g4y g4z g5x g5y g5z g6x g6y g6z;

fun=(s1(i-1,j)-abs(s0(i-1,j))*exp(-x(1)*(g1x^2+g1y^2+g1z^2)-...
(x(2)*g1x+x(3)*g1y+x(4)*g1z)^2))^2+(s2(i-1,j)-abs(s0(i-1,j))*...
exp(-x(1)*(g2x^2+g2y^2+g2z^2)-(x(2)*g2x+x(3)*g2y+x(4)*...
g2z)^2))^2+(s3(i-1,j)-abs(s0(i-1,j))*exp(-x(1)*(g3x^2+...
g3y^2+g3z^2)-(x(2)*g3x+x(3)*g3y+x(4)*g3z)^2))^2+...
(s4(i-1,j)-abs(s0(i-1,j))*exp(-x(1)*(g4x^2+g4y^2+g4z^2)-...
(x(2)*g4x+x(3)*g4y+x(4)*g4z)^2))^2+(s5(i-1,j)-abs(s0(i-...
1,j))*exp(-x(1)*(g5x^2+g5y^2+g5z^2)-(x(2)*g5x+x(3)*g5y+...
x(4)*g5z)^2))^2+(s6(i-1,j)-abs(s0(i-1,j))*exp(-x(1)*(g6x^2+...
g6y^2+g6z^2)-(x(2)*g6x+x(3)*g6y+x(4)* g6z)^2))^2+...
(s1(i,j-1)-abs(s0(i,j-1))*exp(-x(5)*(g1x^2+g1y^2+g1z^2)-...
(x(6)*g1x+x(7)*g1y+x(8)*g1z)^2))^2+(s2(i,j-1)-abs(s0(i,j-1))*...
exp(-x(5)*(g2x^2+g2y^2+g2z^2)-(x(6)*g2x+x(7)*g2y+x(8)*...
g2z)^2))^2+(s3(i,j-1)-abs(s0(i,j-1))*exp(-x(5)*(g3x^2+g3y^2+...
g3z^2)-(x(6)*g3x+x(7)*g3y+x(8)*g3z)^2))^2+(s4(i,j-1)-abs(s0...
(i,j-1))*exp(-x(5)*(g4x^2+g4y^2+g4z^2)-(x(6)*g4x+x(7)*...
g4y+x(8)*g4z)^2))^2+(s5(i,j-1)-abs(s0(i,j-1))*exp(-x(5)*(g5x^2+...
g5y^2+g5z^2)-(x(6)*g5x+x(7)*g5y+x(8)*g5z)^2))^2+(s6(i,j-1)-...
abs(s0(i,j-1))* exp(-x(5)*(g6x^2+g6y^2+g6z^2)-(x(6)*g6x+x(7)*...
g6y+x(8)*g6z)^2))^2+(s1(i,j)-abs(s0(i,j))*exp(-x(9)*(g1x^2+...
g1y^2+g1z^2)-(x(10)*g1x+x(11)*g1y+x(12)*g1z)^2))^2+(s2(i,j)-...
abs(s0(i,j))*exp(-x(9)*(g2x^2+g2y^2+g2z^2)-(x(10)*g2x+x(11)*...
g2y+x(12)*g2z)^2))^2+(s3(i,j)-abs(s0(i,j))*exp(-x(9)*(g3x^2+...
g3y^2+g3z^2)-(x(10)*g3x+x(11)*g3y+x(12)*g3z)^2))^2+...
(s4(i,j)-abs(s0(i,j))*exp(-x(9)*(g4x^2+g4y^2+g4z^2)-(x(10)*...
g4x+x(11)*g4y+x(12)*g4z)^2))^2+(s5(i,j)-abs(s0(i,j))*exp(-x(9)*...
(g5x^2+g5y^2+ g5z^2)-(x(10)*g5x+x(11)*g5y+x(12)*...
g5z)^2))^2+(s6(i,j)-abs(s0(i,j))*exp(-x(9)*(g6x^2+g6y^2+...
```

```

g6z^2)-(x(10)*g6x+x(11)*g6y+x(12)*g6z)^2+(s1(i,j+1)-...
abs(s0(i,j+1))*exp(-x(13)*(g1x^2+g1y^2+g1z^2)-(x(14)*g1x+...
x(15)*g1y+x(16)*g1z)^2)+(s2(i,j+1)-abs(s0(i,j+1))*exp(...
-x(13)*(g2x^2+g2y^2+g2z^2)-(x(14)*g2x+x(15)*g2y+x(16)*...
g2z)^2))^2+(s3(i,j+1)-abs(s0(i,j+1))*exp(-x(13)*(g3x^2+...
g3y^2+g3z^2)-(x(14)*g3x+x(15)*g3y+x(16)*g3z)^2))^2+...
(s4(i,j+1)-abs(s0(i,j+1))*exp(-x(13)*(g4x^2+g4y^2+g4z^2)-...
(x(14)*g4x+x(15)*g4y+x(16)*g4z)^2))^2+(s5(i,j+1)-abs(s0(i,j+...
1))*exp(-x(13)*(g5x^2+g5y^2+g5z^2)-(x(14)*g5x+x(15)*g5y+...
x(16)*g5z)^2))^2+(s6(i,j+1)-abs(s0(i,j+1))*exp(-x(13)*(g6x^2+...
g6y^2+g6z^2)-(x(14)*g6x+x(15)*g6y+x(16)*g6z)^2))^2+(s1(i+...
1,j)-abs(s0(i+1,j))*exp(-x(17)*(g1x^2+g1y^2+g1z^2)-(x(18)*...
g1x+x(19)*g1y+x(20)*g1z)^2))^2+(s2(i+1,j)-abs(s0(i+1,j))*...
exp(-x(17)*(g2x^2+g2y^2+g2z^2)-(x(18)*g2x+x(19)*g2y+...
x(20)*g2z)^2))^2+(s3(i+1,j)-abs(s0(i+1,j))*exp(-x(17)*(g3x^2+...
g3y^2+g3z^2)-(x(18)*g3x+x(19)*g3y+x(20)*g3z)^2))^2+(s4(i+...
1,j)-abs(s0(i+1,j))*exp(-x(17)*(g4x^2+g4y^2+g4z^2)-(x(18)*...
g4x+x(19)*g4y+x(20)*g4z)^2))^2+(s5(i+1,j)-abs(s0(i+1,j))*...
exp(-x(17)*(g5x^2+g5y^2+g5z^2)-(x(18)*g5x+x(19)*g5y+...
x(20)*g5z)^2))^2+(s6(i+1,j)-abs(s0(i+1,j))*exp(-x(17)*...
(g6x^2+g6y^2+g6z^2)-(x(18)*g6x+x(19)*g6y+x(20)*...
g6z)^2))^2+lam*(((x(2)-x(10))^2+(x(3)-x(11))^2+(x(4)-...
x(12))^2+(x(6)-x(10))^2+(x(7)-x(11))^2+(x(8)-x(12))^2+...
(x(14)-x(10))^2+(x(15)-x(11))^2+(x(16)-x(12))^2+(x(18)-...
x(10))^2+(x(19)-x(11))^2+(x(20)-x(12))^2+(x(1)-x(9))^2+...
(x(5)-x(9))^2+(x(13)-x(9))^2+(x(17)-x(9))^2);

```

Fb3.m function fun = fb3(x)

```

global s1 s2 s3 s4 s5 s6 s0 i j lam rn cn;
global g1x g1y g1z g2x g2y g2z g3x g3y g3z g4x ;
global g4y g4z g5x g5y g5z g6x g6y g6z;

```

```

fun=(s1(i,j)-abs(s0(i,j))*exp(-x(1)*(g1x^2+g1y^2+g1z^2)-(x(2)*
g1x+x(3)*g1y+x(4)*g1z)^2))^2+(s2(i,j)-abs(s0(i,j))*exp(-x(1)*
(g2x^2+g2y^2+g2z^2)-(x(2)*g2x+x(3)*g2y+x(4)*g2z)^2))^2+
(s3(i,j)-abs(s0(i,j))*exp(-x(1)*(g3x^2+g3y^2+g3z^2)-(x(2)*

```

$$\begin{aligned}
 &g3x+x(3)*g3y+x(4)*g3z)^2+(s4(i,j)-abs(s0(i,j))*exp(-x(1)*\dots \\
 &(g4x^2+g4y^2+g4z^2)-(x(2)*g4x+x(3)*g4y+x(4)*g4z)^2)^2+ \\
 &(s5(i,j)-abs(s0(i,j))*exp(-x(1)*(g5x^2+g5y^2+g5z^2)-(x(2)* \\
 &g5x+x(3)*g5y+x(4)*g5z)^2))^2+(s6(i,j)-abs(s0(i,j))*exp(-x(1)* \\
 &(g6x^2+g6y^2+g6z^2)-(x(2)*g6x+x(3)*g6y+x(4)*g6z)^2))^2+ \\
 &(s1(i,j+1)-abs(s0(i,j+1))*exp(-x(5)*(g1x^2+g1y^2+g1z^2)- \\
 &(x(6)*g1x+x(7)*g1y+x(8)*g1z)^2))^2+(s2(i,j+1)-abs(s0(i,j+1))*\dots \\
 &exp(-x(5)*(g2x^2+g2y^2+g2z^2)-(x(6)*g2x+x(7)*g2y+x(8)* \\
 &g2z)^2))^2+(s3(i,j+1)-abs(s0(i,j+1))*exp(-x(5)*(g3x^2+g3y^2+ \\
 &g3z^2)-(x(6)*g3x+x(7)*g3y+x(8)*g3z)^2))^2+(s4(i,j+1)- \\
 &abs(s0(i,j+1))*exp(-x(5)*(g4x^2+g4y^2+g4z^2)-(x(6)*g4x+ \\
 &x(7)*g4y+x(8)*g4z)^2))^2+(s5(i,j+1)-abs(s0(i,j+1))*exp(-x(5)* \\
 &(g5x^2+g5y^2+g5z^2)-(x(6)*g5x+x(7)*g5y+x(8)*g5z)^2))^2+\dots \\
 &(s6(i,j+1)-abs(s0(i,j+1))*exp(-x(5)*(g6x^2+g6y^2+g6z^2)-(x(6)* \\
 &g6x+x(7)*g6y+x(8)*g6z)^2))^2+(s1(i,j+2)-abs(s0(i,j+2))*exp(-x(9)* \\
 &(g1x^2+g1y^2+g1z^2)-(x(10)*g1x+x(11)*g1y+x(12)*g1z)^2))^2+ \\
 &(s2(i,j+2)-abs(s0(i,j+2))*exp(-x(9)*(g2x^2+g2y^2+g2z^2)-(x(10)* \\
 &g2x+x(11)*g2y+x(12)*g2z)^2))^2+(s3(i,j+2)-abs(s0(i,j+2))* \\
 &exp(-x(9)*(g3x^2+g3y^2+g3z^2)-(x(10)*g3x+x(11)*g3y+ \\
 &x(12)*g3z)^2))^2+(s4(i,j+2)-abs(s0(i,j+2))*exp(-x(9)*(g4x^2+ \\
 &g4y^2+g4z^2)-(x(10)*g4x+x(11)*g4y+x(12)*g4z)^2))^2+(s5(i,j+ \\
 &2)-abs(s0(i,j+2))*exp(-x(9)*(g5x^2+g5y^2+g5z^2)-(x(10)*g5x+ \\
 &x(11)*g5y+x(12)*g5z)^2))^2+(s6(i,j+2)-abs(s0(i,j+2))*exp(-x(9)* \\
 &(g6x^2+g6y^2+g6z^2)-(x(10)*g6x+x(11)*g6y+x(12)*g6z)^2))^2+ \\
 &(s1(i+1,j)-abs(s0(i+1,j))*exp(-x(13)*(g1x^2+g1y^2+g1z^2)-\dots \\
 &(x(14)*g1x+x(15)*g1y+x(16)*g1z)^2))^2+(s2(i+1,j)-abs(s0(i+ \\
 &1,j))*exp(-x(13)*(g2x^2+g2y^2+g2z^2)-(x(14)*g2x+x(15)*g2y+ \\
 &x(16)*g2z)^2))^2+(s3(i+1,j)-abs(s0(i+1,j))*exp(-x(13)*(g3x^2+ \\
 &g3y^2+g3z^2)-(x(14)*g3x+x(15)*g3y+x(16)*g3z)^2))^2+(s4(i+... \\
 &1,j)-abs(s0(i+1,j))*exp(-x(13)*(g4x^2+g4y^2+g4z^2)-(x(14)*g4x+... \\
 &x(15)*g4y+x(16)*g4z)^2))^2+(s5(i+1,j)-abs(s0(i+1,j))*exp(-x(13)*\dots \\
 &(g5x^2+g5y^2+g5z^2)-(x(14)*g5x+x(15)*g5y+x(16)*g5z)^2))^2+\dots \\
 &(s6(i+1,j)-abs(s0(i+1,j))*exp(-x(13)*(g6x^2+g6y^2+g6z^2)-(x(14)*\dots \\
 &g6x+x(15)*g6y+x(16)*g6z)^2))^2+(s1(i+1,j+1)-abs(s0(i+1,j+1))*\dots \\
 &exp(-x(17)*(g1x^2+g1y^2+g1z^2)-(x(18)*g1x+x(19)*g1y+x(20)*\dots \\
 &g1z)^2))^2+(s2(i+1,j+1)-abs(s0(i+1,j+1))*exp(-x(17)*(g2x^2+... \\
 &g2y^2+g2z^2)-(x(18)*g2x+x(19)*g2y+x(20)*g2z)^2))^2+(s3(i+... \\
 &1,j+1)-abs(s0(i+1,j+1))*exp(-x(17)*(g3x^2+g3y^2+g3z^2)-(x(18)*\dots \\
 &g3x+x(19)*g3y+x(20)*g3z)^2))^2+(s4(i+1,j+1)-abs(s0(i+1,j+1))*\dots \\
 &exp(-x(17)*(g4x^2+g4y^2+g4z^2)-(x(18)*g4x+x(19)*g4y+x(20)*\dots
 \end{aligned}$$

$$\begin{aligned}
 & (g4z)^2)^2+(s5(i+1,j+1)-abs(s0(i+1,j+1))*exp(-x(17)*(g5x^2+... \\
 & g5y^2+g5z^2)-(x(18)*g5x+x(19)*g5y+x(20)*g5z)^2))^2+(s6(i+... \\
 & 1,j+1)-abs(s0(i+1,j+1))*exp(-x(17)*(g6x^2+g6y^2+g6z^2)-(x(18)*... \\
 & g6x+x(19)*g6y+x(20)*g6z)^2))^2+(s1(i+1,j+2)-abs(s0(i+1,j+2))*... \\
 & exp(-x(21)*(g1x^2+g1y^2+g1z^2)-(x(22)*g1x+x(23)*g1y+x(24)*... \\
 & g1z)^2))^2+(s2(i+1,j+2)-abs(s0(i+1,j+2))*exp(-x(21)*(g2x^2+... \\
 & g2y^2+g2z^2)-(x(22)*g2x+x(23)*g2y+x(24)*g2z)^2))^2+... \\
 & (s3(i+1,j+2)-abs(s0(i+1,j+2))*exp(-x(21)*(g3x^2+g3y^2+... \\
 & g3z^2)-(x(22)*g3x+x(23)*g3y+x(24)*g3z)^2))^2+(s4(i+1,j+... \\
 & 2)-abs(s0(i+1,j+2))*exp(-x(21)*(g4x^2+g4y^2+g4z^2)-(x(22)*... \\
 & g4x+x(23)*g4y+x(24)*g4z)^2))^2+(s5(i+1,j+2)-abs(s0(i+1,j+2))*... \\
 & exp(-x(21)*(g5x^2+g5y^2+g5z^2)-(x(22)*g5x+x(23)*g5y+x(24)*... \\
 & g5z)^2))^2+(s6(i+1,j+2)-abs(s0(i+1,j+2))*exp(-x(21)*(g6x^2+... \\
 & g6y^2+g6z^2)-(x(22)*g6x+x(23)*g6y+x(24)*g6z)^2))^2+(s1(i+... \\
 & 2,j)-abs(s0(i+2,j))*exp(-x(25)*(g1x^2+g1y^2+g1z^2)-(x(26)*... \\
 & g1x+x(27)*g1y+x(28)*g1z)^2))^2+(s2(i+2,j)-abs(s0(i+2,j))*... \\
 & exp(-x(25)*(g2x^2+g2y^2+g2z^2)-(x(26)*g2x+x(27)*g2y+... \\
 & x(28)*g2z)^2))^2+(s3(i+2,j)-abs(s0(i+2,j))*exp(-x(25)*(g3x^2+... \\
 & g3y^2+g3z^2)-(x(26)*g3x+x(27)*g3y+x(28)*g3z)^2))^2+(s4(i+... \\
 & 2,j)-abs(s0(i+2,j))*exp(-x(25)*(g4x^2+g4y^2+g4z^2)-(x(26)*... \\
 & g4x+x(27)*g4y+x(28)*g4z)^2))^2+(s5(i+2,j)-abs(s0(i+2,j))*... \\
 & exp(-x(25)*(g5x^2+g5y^2+g5z^2)-(x(26)*g5x+x(27)*g5y+x(28)*... \\
 & g5z)^2))^2+(s6(i+2,j)-abs(s0(i+2,j))*exp(-x(25)*(g6x^2+g6y^2+... \\
 & g6z^2)-(x(26)*g6x+x(27)*g6y+x(28)*g6z)^2))^2+(s1(i+2,j+1)-... \\
 & abs(s0(i+2,j+1))*exp(-x(29)*(g1x^2+g1y^2+g1z^2)-(x(30)*g1x+... \\
 & x(31)*g1y+x(32)*g1z)^2))^2+(s2(i+2,j+1)-abs(s0(i+2,j+1))*exp(-... \\
 & x(29)*(g2x^2+g2y^2+g2z^2)-(x(30)*g2x+x(31)*g2y+x(32)*... \\
 & g2z)^2))^2+(s3(i+2,j+1)-abs(s0(i+2,j+1))*exp(-x(29)*(g3x^2+... \\
 & g3y^2+g3z^2)-(x(30)*g3x+x(31)*g3y+x(32)*g3z)^2))^2+(s4(i+... \\
 & 2,j+1)-abs(s0(i+2,j+1))*exp(-x(29)*(g4x^2+g4y^2+g4z^2)-... \\
 & (x(30)*g4x+x(31)*g4y+x(32)*g4z)^2))^2+(s5(i+2,j+1)-abs(s0(i+... \\
 & 2,j+1))*exp(-x(29)*(g5x^2+g5y^2+g5z^2)-(x(30)*g5x+... \\
 & x(31)*g5y+x(32)*g5z)^2))^2+(s6(i+2,j+1)-abs(s0(i+2,j+1))*... \\
 & exp(-x(29)*(g6x^2+g6y^2+g6z^2)-(x(30)*g6x+x(31)*g6y+... \\
 & x(32)*g6z)^2))^2+(s1(i+2,j+2)-abs(s0(i+2,j+2))*exp(-x(33)*... \\
 & (g1x^2+g1y^2+g1z^2)-(x(34)*g1x+x(35)*g1y+x(36)*... \\
 & g1z)^2))^2+(s2(i+2,j+2)-abs(s0(i+2,j+2))*exp(-x(33)*(g2x^2+... \\
 & g2y^2+g2z^2)-(x(34)*g2x+x(35)*g2y+x(36)*g2z)^2))^2+(s3(i+... \\
 & 2,j+2)-abs(s0(i+2,j+2))*exp(-x(33)*(g3x^2+g3y^2+g3z^2)-... \\
 & (x(34)*g3x+x(35)*g3y+x(36)*g3z)^2))^2+(s4(i+2,j+2)-...
 \end{aligned}$$

```

abs(s0(i+2, j+2))*exp(-x(33)*(g4x^2+g4y^2+g4z^2)-(x(34)*g4x+...
x(35)*g4y+x(36)*g4z)^2+(s5(i+2, j+2)-abs(s0(i+2, j+2))*...
exp(-x(33)*(g5x^2+g5y^2+g5z^2)-(x(34)*g5x+x(35)*g5y+...
x(36)*g5z)^2)+(s6(i+2, j+2)-abs(s0(i+2, j+2))*exp(-x(33)*...
(g6x^2+g6y^2+g6z^2)-(x(34)*g6x+x(35)*g6y+x(36)*...
g6z)^2))^2+lam*((x(1)-x(5))^2+(x(5)-x(9))^2+(x(13)-x(17))^2+...
(x(17)-x(21))^2+(x(25)-x(29))^2+(x(29)-x(33))^2+(x(1)-x(13))^2+...
(x(13)-x(25))^2+(x(5)-x(17))^2+(x(17)-x(29))^2+(x(9)-x(21))^2+...
(x(21)-x(33))^2+(x(2)-x(6))^2+(x(6)-x(10))^2+(x(14)-x(18))^2+...
(x(18)-x(22))^2+(x(26)-x(30))^2+(x(30)-x(34))^2+(x(2)-x(14))^2+...
(x(14)-x(26))^2+(x(6)-x(18))^2+(x(18)-x(30))^2+(x(10)-x(22))^2+...
(x(22)-x(34))^2+(x(3)-x(7))^2+(x(7)-x(11))^2+(x(15)-x(19))^2+...
(x(19)-x(23))^2+(x(27)-x(31))^2+(x(31)-x(35))^2+(x(3)-x(15))^2+...
(x(15)-x(27))^2+(x(7)-x(19))^2+(x(19)-x(31))^2+(x(11)-x(23))^2+...
(x(23)-x(35))^2+(x(4)-x(8))^2+(x(8)-x(12))^2+(x(16)-x(20))^2+...
(x(20)-x(24))^2+(x(28)-x(32))^2+(x(32)-x(36))^2+(x(4)-x(16))^2+...
(x(16)-x(28))^2+(x(8)-x(20))^2+(x(20)-x(32))^2+(x(12)-x(24))^2+...
(x(24)-x(36))^2);

```

Bibliography

- [1] Mark Haacke, Robert W. Brown, Michael R. Thompson, and Ramesh Venkatesan. *Magnetic Resonance Imaging: physical principles and sequence design*. Jhon Wiley and Sons, Toronto, 1999.
- [2] Denis Le Bihan, Jean-Francois Magin, et al. Diffusion Tensor Imaging: concepts and applications. *Journal of Magnetic Resonance Imaging* , 13:534-546, 2001.
- [3] Brown, Mark A., Semelka, and Richard. *MRI: Basic principles and applications*. Wiley-Liss, 1995.
- [4] C. F. Westin, S. E. Maier, H. Mamata, A. Nabavi, F. A. Jolesz, and R. Kikinis. Processing and visualization for diffusion tensor MRI. *Med. Image Anal.*, 6:93-108, 2002.
- [5] C. F. Westin, S. E. Maier, B. Khidhir, et al. Imaging Processing Diffusion Tensor Magnetic Resonance Imaging. *LNCS*, 1679:441-452, 1999.
- [6] Robert-Jan M. Van Geuns, Piotr A. Wielopolski, Hein G. de Bruin, Benno J. Rensing, Peter M. A. Van Ooijen, Marc Hulshoff, Matthijs Oudkerk, and Pim J. de Feyter. Basic Principles of Magnetic Resonance Imaging. *Magnetic Resonance Imaging*, 42:149-156, 1999.
- [7] D. Le Bihan. *Diffusion and perfusion magnetic resonance imaging*. Raven Press, Ltd., New York, 1995.
- [8] D. Tschumperle and R. Deriche. *Dt-MRI Images: Estimation, Regularization and Application*.
- [9] Ray Hashman Hashemi, William G. Bradley, Christopher J. Lisanti. *MRI: The Basics* , 2004.
- [10] Peter J. Basser. *Diffusion Primer. Section on Tissue Biophysics*, NICHD. National Institutes of Health, MD, USA.

- [11] Susumu Mori and Peter B. Barker. Diffusion magnetic resonance imaging, its principle and applications. *The anatomical Rec Ord (New Anat)*257:102-109,1999
- [12] M. Kopf, C. Corinth, O. Haferkamp, and T. F. Nonnenmacher .Anomalous Diffusion of Water in Biological Tissues. *Biophysical Journal* ,70:2950-2985 ,1996.
- [13] Mikael Skorpil, Mathias Engstrom, Anders Nordell.Diffusion-direction-dependent imaging: a novel MRI approach for peripheral nerve imaging. *Magnetic Resonance Imaging*, 25:406-411,2007.
- [14] G J Parker. Analysis of MR diffusion weighted images. *The British Journal of Radiology*, 77:S176-S185, 2004.
- [15] Curtis R. Vogel. *Computational Methods for Inverse Problems*, 2002.
- [16] Uibrich and Michael. *A Generalized Tikhonov Regularization for Non-linear Inverse Ill-posed Problems*.Techn. Univ. Munchen, Fak. F. Math, Report TUM M9810, 1998.
- [17] C.Poupon, C.A. Clark, V. Frouin, J. Regis, I. Bloch, D. Le Bihan and J. F. Mangin. Regularization of Diffusion-Based Direction Maps for the Tracking of Brain White Matter Fascicles. *NeuroImage*,12:184-195,2000.
- [18] Fan Zhang and Edwin R. Hancock. *Tensor MRI Regularization via Graph Diffusion*. UK, 2006.
- [19] Rachid Deriche, David Tschumperle and Christopher Lenglet. DT-MRI Estimation, Regularization and Fiber Tractography. *Biomedical Imaging,IEEE intr. sympo.*,1:9-12,2004.
- [20] D. Tschumperle and R. Deriche. DT-MRI Image:Estimation, Regularization, and Application. *LNCS*, 2809: 530-541,2004.
- [34] Mori S, Crain BJ, Chacko VP, Van Zijl PC. Three Dimensional Tracking of Axonal Projections in the Brain by Magnetic Resonance Imaging. *Ann Neurol* , 45: 265-269, 1999.
- [22] Jones DK, Simmons A, Williams SC, Horsfield MA. Non-invasive Assessment of Axonal Fiber Connectivity in the Human Brain via diffusion Tensor MRI. *Magnetic Resonance in Medicine*, 42: 37-41,1999.

- [23] Peter J. Basser, Sinisa Pajevic, Carlo Pierpaoli, Jeffrey Duda, and Akram Aldroubi. In Vivo Fiber Tractography Using DT-MRI Data. *Magnetic Resonance in Medicine*, 44:625-632, 2000.
- [24] Bin Chen and Edward W. Hsu. Noise Removal in Magnetic Resonance Diffusion Tensor Imaging. *Magnetic Resonance in Medicine*, 54: 393-407, 2005.
- [25] Heinz W Engl, Karl Kunisch and Andreas Neubauer. Convergence Rates for Tikhonov Regularization of Nonlinear Ill-posed Problem. *Inverse Problems*, 5:523-540,1989.
- [26] O. Coulon, D. C. Alexander and S Arridge. Diffusion Tensor Magnetic Resonance Image Regularization. *Medical Image Analysis*, 8:47-67, 2004.
- [27] Adam W. Anderson. Theoretical Analysis of the Effect of Noise on Diffusion Tensor Imaging. *Magnetic Resonance in Medicine*, 46:1174-1188, 2001.
- [28] David S. Tuch, Timothy G. Reese, Mette R. Wiegell, Nikos Makris, John W. Belliveau, and Van J. Wedeen. High Angular Resolution Diffusion Imaging Reveals Intravoxel White Matter Fiber Heterogeneity. *Magnetic Resonance in Medicine*, 48:577-582, 2002.
- [29] Dominik Weishaupt, Victor D. Koechli, and Borut Marincek. *How does MRI work? : an introduction to the physics and function of magnetic resonance imaging*. Berlin, New York : Springer, c2003.
- [30] Kenneth Holmstrm. The Tomlab Optimization Environment in Matlab. *Advanced Modeling and Optimization*, 1:47-69, 1999.
- [31] Susumu Mori and Peter C. M. Van Zijl. Fiber Tracking: Principles and strategies-a technical review. *NMR Biomed*, 15:468-480,2002.
- [32] Laurent Hermoye. <http://en.wikipedia.org/wiki/Image:Illus4ti.gif>.
- [33] E. O. Stejskal and J. E. Tanner. Spin Diffusion Measurements: Spin Echoes in the Presence of a Time-Dependent Field Gradient. *The Journal of Chemical Physics*, 42:288-292,1964.
- [34] Dwight G. Nishimura. *Principles of Magnetic Resonance Imaging*, April 1996.