Nuno Maria Sampaio Mesquita

Licenciatura em Ciências de Engenharia Biomédica

Diffusional Kurtosis Imaging using a fast Heuristic Constrained Linear Least Squares Algorithm: a plugin for OsiriX

Dissertação para obtenção do Grau de Mestre em Engenharia Biomédica

Orientador: José Manuel Fonseca, Professor Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Co-orientador: Mestre João Santinha, Investigador, Uninova

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2014

Diffusional Kurtosis Imaging using a fast heuristic constrained linear least squares algorithm: a plugin for OsiriX

“Para seres bem sucedido tens que trabalhar muito.”

Anónimo

Aknowledgements

I would like to thank Professor José Manuel Fonseca not only for his guidance and support abut also for giving me the opportunity to work on this subject. I am especially grateful to João Santinha for his insight and help on the project.

I would also like to thank all the people at CA3 for their crucial help on structuring the project. A special thank you to António for his help and insight, Luís for cheering me up whenever I was sad, Marília for the lifts, Leonardo for his jokes that helped me get through the darkest days, and Zé for his invaluable help.

Finally I would like to thank my mother and father for supporting me throught this journey, and my brother for never giving up on me.

Abstract

Diffusion Kurtosis Imaging (DKI) is a fairly new magnetic resonance imaging (MRI) technique that tackles the non-gaussian motion of water in biological tissues by taking into account the restrictions imposed by tissue microstructure, which are not considered in Diffusion Tensor Imaging (DTI), where the water diffusion is considered purely gaussian. As a result DKI provides more accurate information on biological structures and is able to detect important abnormalities which are not visible in standard DTI analysis.

This work regards the development of a tool for DKI computation to be implemented as an OsiriX plugin. Thus, as OsiriX runs under Mac OS X, the program is written in Objective-C and also makes use of Apple's Cocoa framework. The whole program is developed in the Xcode integrated development environment (IDE).

The plugin implements a fast heuristic constrained linear least squares algorithm (CLLS-H) for estimating the diffusion and kurtosis tensors, and offers the user the possibility to choose which maps are to be generated for not only standard DTI quantities such as Mean Diffusion (MD), Radial Diffusion (RD), Axial Diffusion (AD) and Fractional Anisotropy (FA), but also DKI metrics, Mean Kurtosis (MK), Radial Kurtosis (RK) and Axial Kurtosis (AK).

The plugin was subjected to both a qualitative and a semi-quantitative analysis which yielded convincing results. A more accurate validation process is still being developed, after which, and with some few minor adjustments the plugin shall become a valid option for DKI computation.

Keywords: Magnetic resonance imaging, diffusion kurtosis imaging, diffusion tensor imaging, OsiriX, heuristic constrained linear least squares.

Resumo

Imagem por curtose de difusão é uma técnica de imagem por ressonância magnética relativamente nova que estuda o movimento não-gaussiano da água em tecidos biológicos tendo em conta as restrições impostas por microestruturas presentes nos tecidos. Restrições essas que não são consideradas em imagem por tensor de difusão, que considera uma difusão gaussiana. Consequentemente, a técnica de imagem por curtose de difusão permite obter informação mais detalhada sobre as estruturas biologicas e possibilita a identificação de lesões que não são perceptíveis em imagem por tensor de difusão.

O presente trabalho consiste no desenvolvimento de uma ferramenta para computação de imagem por curtose de difusão. Esta ferramenta tem um formato de plugin para o software de imagem médica OsiriX. Como este software corre no sistema operativo Mac OS X, o programa é escrito em Objective-C, e usufrui da interface de programação Cocoa. O programa é integralmente desenvolvido no ambiente de desenvolvimento integrado Xcode.

O plugin implementa um algoritmo que segue uma aproximação heurística do método dos mínimos quadrados lineares para a estimar os tensores de difusão e curtose e, permite ao utilizador escolher que mapas gerar para tanto métricas já conhecidas da imagem por tensor de difusão como difusão, difusão radial,

difusão axial e anisotropia fraccionária assim como métricas de imagem por curtose de difusão, sendo estas a curtose média, curtose axial e curtose radial.

O plugin foi sujeito tanto a uma análise qualitativa como a uma análise semi-quantitativa, produzindo bons resultatdos para ambas. Uma validação mais precisa está em desenvolvimento, após a qual, e depois de alguns pequenos melhoramentos e ajustes, a ferramenta poderá ser uma opção válida para computação de imagem por curtose de difusão em ambiente clínico.

Palavras-chave: Imagem por ressonância magnética, imagem por curtose de difusão, imagem por tensor de difusão, OsiriX, método dos mínimos quadrados lineares heurístico.

Contents

AKNOWLEDGEMENTS **VII**

ABSTRACT **IX**

RESUMO **XI**

CONTENTS **XIII**

LIST OF FIGURES **XIX**

LIST OF TABLES **XXIII**

ACRONYMS **XXIV**

1 INTRODUCTION **1**

1.1 Motivation **1**

1.2 Objectives **2**

1.3 State of the art **3**

2 THEORETICAL BACKGROUND 7

2.1 Diffusion MRI 7

2.2 Definition of Kurtosis 8

2.3 Modelling water diffusion 10

2.3.1 Multiple compartment model 13

2.4 MRI signal 14

2.4.1 DKI as an extension of DTI 14

2.4.1.1 b-value range 16

2.5 Algorithm for DKI computation 18

2.5.1 Formulation 18

2.5.2 Solution 21

2.6 Standard DTI metrics 23

2.6.1 Mean diffusion (MD) 24

2.6.2 Axial diffusion (AD) 24

2.6.3 Radial diffusion (RD) 24

2.6.4 Fractional anisotropy (FA) 24

2.7 DKI metrics 25

2.7.1 Mean kurtosis (MK) 25

2.7.2 Axial kurtosis (AK) 26

2.7.3 Radial kurtosis (RK) 26

2.7.4 Singularities in F_1, F_2, G_1 and G_2 27

2.8 Anomalous estimates of tensor eigenvalues 31

2.8.1 Fractional Anisotropy Anomaly 31

2.8.2 Proposed Solutions 31

3 METHODOLOGY 33

3.1 Implementation **35**

3.1.1 Computation of A_D^+ and A_K^+ **38**

3.2 Voxel processing **41**

3.2.1 Computation of *D*'s eigenvalues and eigenvectors **44**

3.2.2 DTI and DKI metrics computation **45**

3.3 Installation **49**

4 RESULTS AND VALIDATION **51**

4.1 Results **51**

4.1.1 *ABS* and *ZERO* methods comparison **51**

4.1.2 Generated maps for two sets of DWIs **53**

4.2 Validation **63**

5 CONCLUSION **69**

5.1 Conclusion **69**

5.2 Future work **70**

6 BIBLIOGRAPHY **71**

List of Figures

FIGURE 1.1: DKE GRAPHICAL USER INTERFACE. 2

FIGURE 2.1: THREE ISOTROPIC DIFFUSION DISPLACEMENT PROBABILITY DISTRIBUTIONS. THE CORRESPONDING DIFFUSION COEFFICIENTS ARE IDENTICAL, BUT THE VALUES FOR DIFFUSIONAL KURTOSIS ARE DIFFERENT. THE SOLID CURVE WITH $K = 0$ IS THE GAUSSIAN FORM. TAKEN FROM [1]. 8

FIGURE 2.2: AN ILLUSTRATION OF TWO DISTRIBUTIONS WITH DIFFERENT KURTOSIS. THE DOTTED LINES SHOW GAUSSIAN DISTRIBUTIONS, THE SOLID LINES SHOW DISTRIBUTIONS WITH POSITIVE KURTOSIS (LEFT PANEL) AND NEGATIVE KURTOSIS (RIGHT PANEL). TAKEN FROM [18]. 9

FIGURE 2.3: DIFFUSION RESTRICTIONS IMPOSED BY AN AXON. 11

FIGURE 2.4: AN ILLUSTRATION OF THE DIFFUSION AND KURTOSIS DISTRIBUTION IN THE 3D SYSTEM DEFINED BY DIFFUSION EIGENVECTORS ($\lambda_1, \lambda_2, \lambda_3$). THE DIFFUSION DISTRIBUTION IS AN ELLIPSOID (BLUE) WITH THE PRINCIPLE DIRECTION POINTING AT λ_1. THE KURTOSIS DISTRIBUTION, FROM A SIMPLIFIED POINT OF VIEW, IS LIKE A PANCAKE (YELLOW) WITH HIGHER KURTOSIS ALONG RADIAL DIRECTION OF THE DIFFUSION ELLIPSOID, INDICATING RESTRICTED DIFFUSION. TAKEN FROM [5]. 12

FIGURE 2.5: COMPARISON OF DTI AND DKI FITTING MODELS. FOR DTI, THE LOGARITHM OF DIFFUSION-WEIGHTED SIGNAL INTENSITY (CIRCLES) AS A FUNCTION OF THE B-VALUE IS FIT, FOR SMALL B-VALUES, TO A STRAIGHT LINE. IN BRAIN, THIS FIT IS OFTEN BASED ON THE SIGNAL FOR B ¼ 0 AND B ¼ 1000 S/MM2. FOR DKI, THE LOGARITHM OF THE SIGNAL INTENSITY IS FIT, FOR SMALL B-VALUES, TO A PARABOLA. IN BRAIN, THIS FIT MAY BE BASED ON THE SIGNAL FOR B ¼ 0, B ¼ 1000, AND B ¼ 2000 S/MM2. TAKEN FROM [6]. 17

FIGURE 3.1: THE *BROWSERCONTROLLER* WINDOW PROVIDING A LIST OF AVAILABLE STUDIES IN THE DATABASE AND SETS OF TUMBNAIL IMAGES. 34

FIGURE 3.2: THE *VIEWERCONTROLLER*, CONTAINING BASIC IMAGE MANIPULATION TOOLS (DEFINITION OF ROIS (REGION OF INTEREST), CONTRAST ADJUSTMENT, ETC.) AND THE *DCMVIEW* OBJECT. 35
FIGURE 3.3: THE DKI PLUGIN GRAPHICAL USER INTERFACE (GUI). 36
FIGURE 3.4: CODE FOR ACQUIRING THE GRADIENT DIRECTIONS AND *B*-VALUES FROM THE METADATA OF THE DWI SET. 37
FIGURE 3.5: MEMORY ALLOCATION FOR THE A_K MATRIX. 38
FIGURE 3.6: FUNCTION THAT GIVEN 3 MATRICES, MULTIPLIES THEM AND RETURNS THE RESULTANT MATRIX *MULT*. THIS FUNCTION IS USED TO CALCULATE DE PSEUDO-INVERSE MATRICES BY MULTIPLYING THE 3 MATRICES RESULTANT FROM THE SVD PROCESS. 38
FIGURE 3.7: THE CREATION OF A NEW VIEWER FOR A SPECIFIC METRIC (IN THIS CASE, THE MEAN DIFFUSION), SHOULD THE USER HAVE SELECTED IT IN THE GUI. 39
FIGURE 3.8: THE RELEASE OF THE MEMORY ALLOCATED FOR THE A_D MATRIX. 40
FIGURE 3.9: FLOWCHART OF THE PLUGIN. THE COMPUTATION OF DKI QUANTITIES IS THE CENTRAL TASK OF THIS TOOL, AND IS REPEATED FOR EVERY SLICE IN THE SERIES. 41
FIGURE 3.10: THIS PIECE OF CODE SHOWS THE *FOR* CYCLE RESPONSIBLE FOR ITERATING THROUGH THE IMAGES RESPECTING THE ORDER DESCRIBED IN MATRIX (90). 42
FIGURE 3.11: THE DECLARATION OF POINTERS TO THE VOXEL CURRENTLY BEING PROCESSED, WHERE x IS THE INDEX OF THE VOXEL BEING COMPUTED. 42
FIGURE 3.12: FLOWCHART FOR THE *DKICOMPUTATION* FUNCTION 43
FIGURE 3.13: THE INDEX OF THE DIFFUSION TENSOR ELEMENTS IN THE X_D ARRAY. 44
FIGURE 3.14: THE POPULATION OF A IN THE ORDER DESCRIBED IN FIGURE 3.13 44
FIGURE 3.15: THE ORDERLY POPULATION OF THE EIGENVECTOR'S ARRAY 45
FIGURE 3.16: ONE OF THE CASES OF THE SWITCH STATEMENT DECLARED IN FIGURE 3.16. IN THIS CASE, THE SUM OF THE INDICES IS 2 WHICH OCCURS FOR BOTH W_{0002} AND W_{0011}, THUS, THESE TWO CASES ARE DISTINGUISHED BY A SUBSEQUENT IF STATEMENT. 46
FIGURE 3.17: FUNCTION FOR COMPUTING MK. VARIOUS *IF* STATEMENTS ARE IMPLEMENTED SO AS TO DEAL WITH ANY EVENTUAL SINGULARITIES IN COMPUTING F_1 AND F_2. 48
FIGURE 3.18: PLUGIN INSTALLATION EXECUTABLE. 49
FIGURE 4.1: MEAN DIFFUSION MAPS COMPUTED BY (A) DKI PLUGIN, (B) DKE SOFTWARE, USING THE CLLS-H AND CLLS-QP ALGORITHMS RESPECTIVELY. 55
FIGURE 4.2: RADIAL DIFFUSION MAPS COMPUTED BY (A) DKI PLUGIN, (B) DKE SOFTWARE, USING THE CLLS-H AND CLLS-QP ALGORITHMS RESPECTIVELY. 56
FIGURE 4.3: AXIAL DIFFUSION MAPS COMPUTED BY (A) DKI PLUGIN, (B) DKE SOFTWARE, USING THE CLLS-H AND CLLS-QP ALGORITHMS RESPECTIVELY. 56
FIGURE 4.4: FRACTIONAL ANISOTROPY MAPS COMPUTED BY (A) DKI PLUGIN, (B) DKE SOFTWARE, USING THE CLLS-H AND CLLS-QP ALGORITHMS RESPECTIVELY. 57
FIGURE 4.5: MEAN KURTOSIS MAPS COMPUTED BY (A) DKI PLUGIN, (B) DKE SOFTWARE, USING THE CLLS-H AND CLLS-QP ALGORITHMS RESPECTIVELY. 57

FIGURE 4.6: RADIAL KURTOSIS MAPS COMPUTED BY (A) DKI PLUGIN, (B) DKE SOFTWARE, USING THE CLLS-H AND CLLS-QP ALGORITHMS RESPECTIVELY. 58
FIGURE 4.7: AXIAL KURTOSIS MAPS COMPUTED BY (A) DKI PLUGIN, (B) DKE SOFTWARE, USING THE CLLS-H AND CLLS-QP ALGORITHMS RESPECTIVELY. 58
FIGURE 4.8: A LARGE ROI CONTAINING THE CENTRAL PART OF THE SLICE FOR THE MEAN KURTOSIS MAPS. 60
FIGURE 4.9: SMALL ROI IN MEAN DIFFUSION MAPS. 60
FIGURE 4.10: MAPS GENERATED BY THE PLUGIN, (A) MEAN DIFFUSION; (B) RADIAL DIFFUSON; (C) AXIAL DIFFUSION; (D) FRACTIONAL ANISOTROPY. 62
FIGURE 4.11: MAPS GENERATED BY THE PLUGIN, (A) MEAN KURTOSIS; (B) RADIAL KURTOSIS; (C) AXIAL KURTOSIS. 63
FIGURE 4.12: AN EXAMPLE OF AN ACQUIRED IMAGE FOR ($b = 0$). AS PREDICTED, THE SEPARATION OF THE SIGNAL FROM FAT, CREAM AND WATER IS VISIBLE DUE TO THE PREVIOUS HEATING TREATMENT. 65
FIGURE 4.13: THE MEAN KURTOSIS MAPS FOR THE PHANTOM AND TWO ROIS, (A) WATER REGION AND (B) CREAM REGION. 65

List of Tables

TABLE 2.1: MEAN SQUARED ERROR (MSE) AND STANDARD DEVIATION (SD) OF THE CORRECTED VOXEL VALUES FOR THE *ZERO* AND *ABS* METHODS. 52

TABLE 4.2: AVERAGE OF ALL MEAN VOXEL VALUES FOR THE ROIS DEFINED IN THE DKE GENERATED MAPS AND THE MEAN SQUARED ERROR OF THE VALUES IN THE MAPS GENERATED BY THE DEVELOPED PLUGIN..... 61

Acronyms

AD Axial Diffusion

AK Axial Kurtosis

CLLS Constrained Linear Least Squares

CLLS-H Constrained Linear Least Squares – Heuristic

CLLS-QP Constrained Linear Least Squares – Quadratic Programming

DICOM Digital Imaging and Communications in Medicine

DKE Diffusional Kurtosis Estimator

DKI Diffusion Kurtosis Imaging

DTI Diffusion Tensor Imaging

DWI Diffusion Weighted Image

FA Fractional Anisotropy

IID Independent and Identically Distributed

MRI Magnetic Resonance Imaging

MD Mean Diffusion

MK Mean Kurtosis

MSE Mean Squared Error

Nifti Neuroimaging Informatics Technology Initiative

PDF Probability Density Function

RD Radial Diffusion

RK Radial Kurtosis

SD Standard Deviation

SVD Singular Value Decomposition

1

Introduction

1.1 Motivation

Diffusion Kurtosis Imaging (DKI) is known to provide more accurate information on tissue microstructure than standard diffusion tensor imaging (DTI) and is applicable in situations where the latter is worthless, such as in investigating abnormalities in grey matter and other tissues of isotropic nature [1].

Software for DKI computation has already been developed and made available (diffusion kurtosis estimator (DKE) [31]), howbeit, it only works on windows platforms and its usage is not very straightforward. One setback of this software is that the gradient vectors, mandatory for DKI processing, are selected from a list that only comprises predefined Siemens™ protocols, as shown in figure 1.1. This means, that, for a set of diffusion weighted images (DWIs) acquired from a different MRI manufacturer (e.g., Philips™), the user must create the file with the gradient vectors. Additionally, DKE does not allow any type of image visualization and manipulation, which is very helpful in a medical environment. Moreover, it is not possible to choose which maps to generate, as this software computes maps for mean, radial and axial diffusion, fractional anisotropy and mean, radial and axial kurtoses metrics, increasing the processing time.

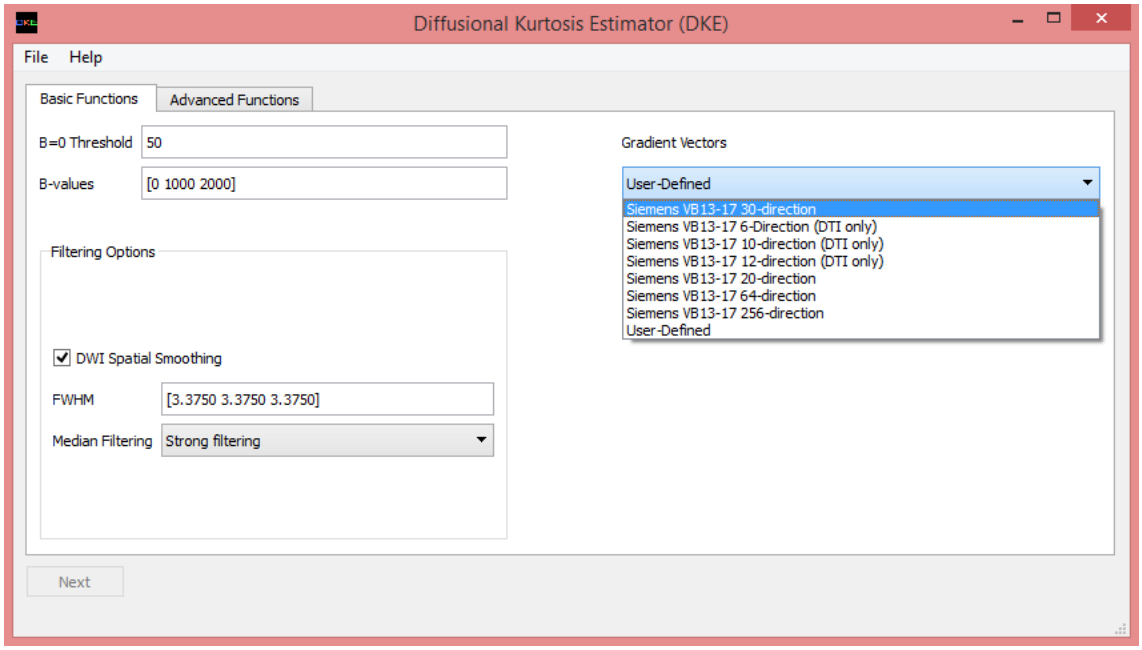

Figure 1.1: DKE graphical user interface.

All of this makes the development of an easy-learning and fast-processing DKI tool for medical usage of the utmost importance. This work attempts to answer this need by developing a tool for DKI computation in the form of a plugin for OsiriX.

The reason behind the choosing of OsiriX, was that it is an open source software dedicated to DICOM images ubiquitous in the medical world, with an architecture that permits an easy implementation of any developed tool in the form of a plugin. Also, all the required information for DKI computation is obtained directly from the image metadata, which excludes any need for additional inputs.

1.2 Objectives

The objective of this work is to answer the need for a fast and user-friendly tool for DKI computation by developing a plugin for OsiriX capable of rapidly generate maps for Mean Diffusion (MD), Radial Diffusion (RD), Axial Diffusion (AD), Fractional Anisotropy (FA), Mean Kurtosis (MK), Radial Kurtosis (RK) and Axial Kurtosis (AK).

One of the main goals for this tool is to assure its ergonomic usage as the user will only be required to select which maps are to be computed and to define a threshold for voxel computation and an additional control variable. No additional inputs are needed since all the information required by the program is assessed through the Diffusion Weighted Images (DWIs) set's metadata given that it is complete.

1.3 State of the art

Diffusion has been measured through spin echoes since the 1950. And, in the 1980s, a combination with magnetic resonance imaging (MRI) emerged which allowed for a better understanding of diffusion processes in human brain. This technique only described diffusion as a scalar constant, which was not accurate in assessing the heterogeneities of biological tissue. As a result, in 1994, Basser *et al.* proposed a new MRI modality, which consisted on, within a voxel, estimating a symmetric, positive definite 3x3 tensor, which would provide some insight on the diffusion on that very voxel [2]. This technique is commonly referred to as diffusion tensor imaging (DTI) and has been widely used ever since. By analysing the anisotropic displacement of water in several voxels, DTI was shown to provide useful information such as fibre tract orientation and mean particle displacements [2].

One key aspect of diffusion tensor imaging is the assumption that water displacement follows a gaussian probability distribution function (PDF), which only holds for simple homogeneous liquids (e.g., a glass of water) [3]. This assumption is translated into a monoexponential dependency of the diffusion-weighted (DW) signal on b-value. However, the relationship between the diffusion weighted signal and the b-value in neural tissues is known to be non-monoexponential [4]. The reason behind this is that in biological tissues, the restrictions to water diffusion imposed by microstructures, deviate the PDF from a

gaussian form. Therefore, DTI is not an accurate technique in describing water diffusion [5].

Before attempting to characterize water diffusion in biological tissues, one must have an underlying model to describe the mechanisms behind this process. However, an accurate modelling of water diffusion in non-homogeneous environments (more specifically in biological tissues) is extremely complex. Many models have been developed to characterize water diffusion in brain. These include the multicompartment model which is the simplest and most applied one [6], the Kärger model [7], the statistical diffusion model [8], the generalized diffusion tensors [9] and one-dimensional model with barriers [10]. These models serve as a basis for interpretation of the quantities measured in diffusion MRI techniques, when in fact, the actual calculated values do not depend on the adopted model.

Although it is possible to get the full PDF for water diffusion rather than just the excess kurtosis by means of q-space imaging using very well-defined imaging methods [11], this level of accuracy is very demanding in hardware and requires large b-values and a strong gradient. Because of this, it is very difficult to fit such protocols in a medical environment [6]. DKI sidesteps these costly requirements and emerges as a refined diffusion MRI technique by introducing a dimensionless metric known as diffusional kurtosis, the 4th central moment of the diffusion distribution. And, by doing so, DKI quantifies the extent to which water diffusion is non-Gaussian. The higher the diffusion kurtosis, the more the restrictions are imposed by the environment.

Since its introduction, and because DKI yields not only conventional diffusion information but also the diffusional kurtosis, it has been successful in several studies where conventional MRI was only partially succeeding. For instance, in an attempt to identify biological mechanisms underlying ischemia by analysis of diffusion in white matter (WM), DKI throve as the kurtosis maps exhibited heterogeneities in ischemic lesions which were not present on the diffusion maps [12]. Other studies have proven that DKI is crucial in detecting certain neural

pathologies such as Alzheimer's disease [13], attention-deficit hyperactivity disorder (ADHD) [14] and schizophrenia [15].

The DKI model is parameterized by the diffusion and kurtosis tensors (***D*** and ***K***). The estimation of these tensors for each and every voxel is the core of DKI computing as all the important metrics in DKI analysis are derived from them.

There are several ways to estimate the diffusion and kurtosis tensors. Methods such as unconstrained nonlinear least squares (UNLS) and unconstrained linear least squares (ULLS) have been used to estimate ***D*** and ***K*** and were shown not to provide satisfactory tensor estimates. And even after some adjustments, the algorithms are not viable when compared to constrained linear least squares (CLLS) formulations [16].

The CLLS formulation is simply a casting of the tensor estimation problem as linear least squares with linear constrains safeguarding the physical and biological meaning of directional diffusivities and kurtoses. Tabesh et al., 2011, proposed two algorithms for solving the CLLS formulation for ***D*** and ***K*** estimation, the quadratic programming (CLLS-QP) approach and the heuristic (CLLS-H) approach. Both these algorithms gave more reliable estimations of the tensors, with the CLLS-QP providing the exact solution for the estimation problem a higher computational cost and the CLLS-H yielding an approximate solution at a lower processing time.

The CLLS-QP algorithm is implemented in DKE, which, as mentioned before, is a software for processing DKI datasets. This software is currently supported on 32 and 64-bit Windows platforms. It consists on a graphical user interface (GUI) and a set of comand-line based programs. The software generates maps for standard DTI quantities (MD, AD, RD and FA) as well as for DKI metrics (MK, AK, RK). It accepts both DICOM and NIfTI formats and the user may choose the algorithm for ***D*** and ***K*** estimation from: constrained linear weighted, unconstrained linear unweighted and unconstrained nonlinear.

2

Theoretical Background

2.1 Diffusion MRI

The diffusion of a water molecule through a given medium can be characterized as a Brownian random process, meaning that the motion of the particle is the result of a large number of collisions with the surrounding particles. These collisions are independent and identically distributed (i.i.d.), hence, as stated by the central limit theorem, the sum of many i.i.d variables with finite mean and standard deviation results in a normal distribution, the probability that a particle undergoes a vectorial displacement $\boldsymbol{r}$ over a time interval $\boldsymbol{t}$ is given by a Gaussian distribution [17]. The describing of water diffusion in biological tissues by a Gaussian PDF is the core of diffusion tensor imaging. However, as mentioned before, this only holds for homogeneous liquids, whereas for biological tissues, the restrictions imposed by tissue microstructure deviate the PDF from a Gaussian form.

Diffusion Kurtosis Imaging (DKI) tackles the problem of non-gaussianity assumed in diffusion tensor imaging (DTI) by quantifying the kurtosis of the diffusion PDF. Figure 2.1 shows three different PDFs with identical diffusion coefficients but different values of kurtosis, which proves that the diffusion coefficient alone does not provide enough insight on the motion of water molecules.

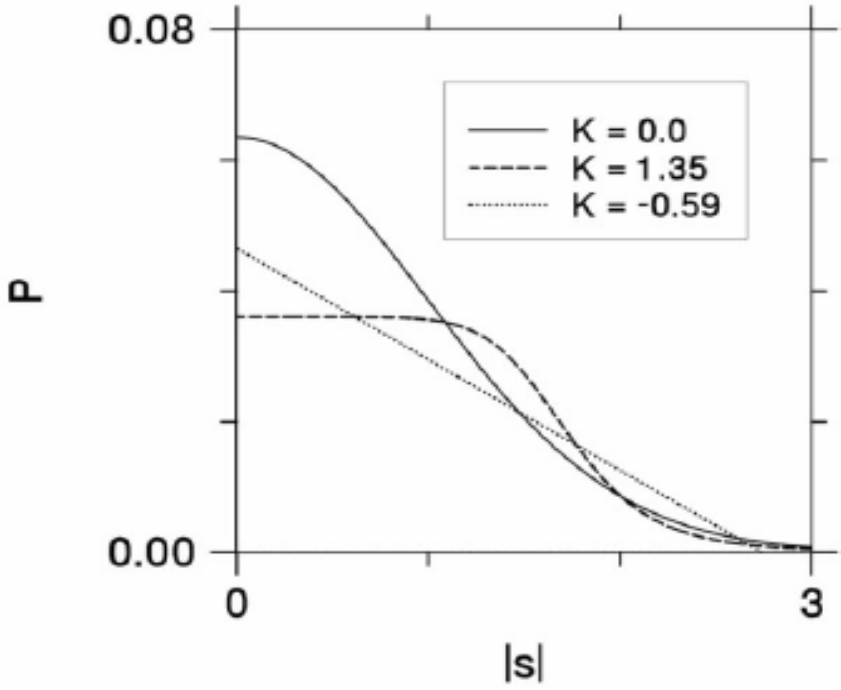

Figure 2.1: Three isotropic diffusion displacement probability distributions. The corresponding diffusion coefficients are identical, but the values for diffusional kurtosis are different. The solid curve with $\boldsymbol{K = 0}$ is the Gaussian form. Taken from [1].

2.2 Definition of Kurtosis

In statistics, distributions can be described according to their central tendency, variability and shape. Kurtosis sheds a light on the shape of a distribution for it measures the deviation from its Gaussian reference. It is defined as the standardized fourth population moment about the mean,

$$\beta_2 = \frac{E(X-\mu)^4}{(E(X-\mu)^2)^2} = \frac{\mu_4}{\sigma^4} \tag{1}$$

where E is the expectation operator, μ is the mean, μ_4 is the fourth moment about the mean and σ the standard deviation.

A Gaussian distribution has a kurtosis of 3. Consequently, $\beta_2 - 3$ is conveniently used so that the reference normal distribution has a kurtosis of zero. The minimum theoretical value for kurtosis is therefore -2.

Figure 2.2 illustrates two distributions, on the left a distribution with positive kurtosis (leptokurtic) and on the right, a distribution with negative kurtosis (platykurtic). It is clear that distributions with negative kurtosis are flatter and have lighter tails, whereas distributions with positive kurtosis have a higher peak and heavier tails.

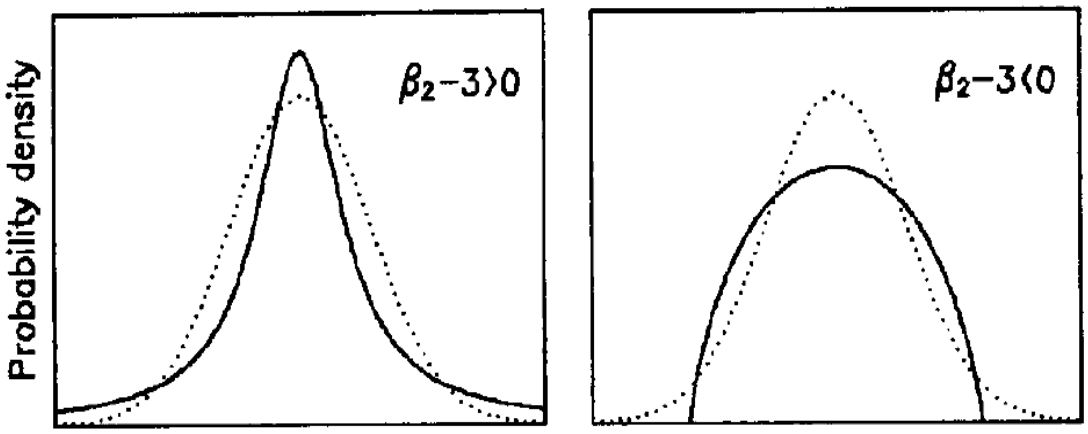

Figure 2.2: An illustration of two distributions with different kurtosis. The dotted lines show Gaussian distributions, the solid lines show distributions with positive kurtosis (left panel) and negative kurtosis (right panel). Taken from [18].

A common misconception about kurtosis is that it only concerns the peakedness of a distribution, however it also regards its tailedness. It may be represented as a movement of mass that does not alter the variance [18].

2.3 Modelling water diffusion

Considering $P(\boldsymbol{r}, t)$ as a PDF for water diffusion, with $\boldsymbol{r}$ being the vectorial displacement of a given molecule over a time interval $\boldsymbol{t}$. One may average an arbitary function $A(\boldsymbol{r})$ over that very PDF as follows

$$\langle A(\boldsymbol{r})\rangle = \int d^3\boldsymbol{r}\, P(\boldsymbol{r}, t) A(\boldsymbol{r}) \tag{2}$$

The diffusion coefficient of the water molecule in a direction $\boldsymbol{n}$, and assuming $|\boldsymbol{n}| = 1$, is given by

$$D(\boldsymbol{n}) = \frac{1}{2t}\langle(\boldsymbol{r}\cdot\boldsymbol{n})^2\rangle \tag{3}$$

The diffusional kurtosis is defined by taking equation (1) and subtracting 3 in order to assure a kurtosis of 0 for a Gaussian diffusion.

$$K(\boldsymbol{n}) = \frac{\langle(\boldsymbol{r}\cdot\boldsymbol{n})^4\rangle}{\langle(\boldsymbol{r}\cdot\boldsymbol{n})^2\rangle^2} - 3 \tag{4}$$

Both these metrics are defined as tensors in order to account for the anisotropic characteristics of the medium and quantify the directionality of the diffusion process. The diffusion tensor $\boldsymbol{D}$ is defined as

$$D_{ij} = \frac{1}{2t}\langle r_i r_j\rangle \tag{5}$$

and the coefficients of the kurtosis tensor $\boldsymbol{K}$ as

$$W_{ijkl} = \frac{9}{\langle\boldsymbol{r}\cdot\boldsymbol{r}\rangle^2}\big(\langle r_i r_j r_k r_l\rangle - \langle r_i r_j\rangle\langle r_k r_l\rangle - \langle r_l r_k\rangle\langle r_j r_i\rangle - \langle r_i r_l\rangle\langle r_j r_k\rangle\big) \tag{6}$$

with r_i being the i-th component of the displacement vector $\boldsymbol{r}$.

These two tensors allow the calculation of the diffusion coefficient and diffusional kurtosis in a given direction as follows

$$D(\boldsymbol{n}) = \sum_{i=1}^{3}\sum_{j=1}^{3} n_i n_j D_{ij} \tag{7}$$

and

$$K(\boldsymbol{n}) = \frac{\overline{D}^2}{[D(\boldsymbol{n})]^2} \sum_{i=1}^{3} \sum_{j=1}^{3} \sum_{k=1}^{3} \sum_{l=1}^{3} n_i n_j n_k n_l W_{ijkl} \tag{8}$$

The components of $\boldsymbol{D}$ may be used to describe the diffusion geometrically. An isotropic diffusion is represented by a 3D sphere as there is no preferential direction to the diffusion process whereas, for anisotropic diffusion the 3D representative is an ellipsoid. A much more complex structure is needed to describe the diffusional kurtosis due to $\boldsymbol{K}$ being a 4-th order tensor. Figure 2.4 illustrates the diffusion ellipsoid for an anisotropic environment (e.g. white matter axons) and attempts to represent the diffusional kurtosis as a pancake-like ellipsoid. Since diffusion is most freely along the fibres, as illustrated in figure 2.3, the diffusion ellipsoid is wider on that direction while the kurtosis is very small. On the other hand, in the perpendicular direction, fibre barriers prevent a free diffusion deviating the diffusion PDF from a gaussian behaviour. And since kurtosis quantifies this deviation, the kurtosis ellipsoid is wider along that very direction.

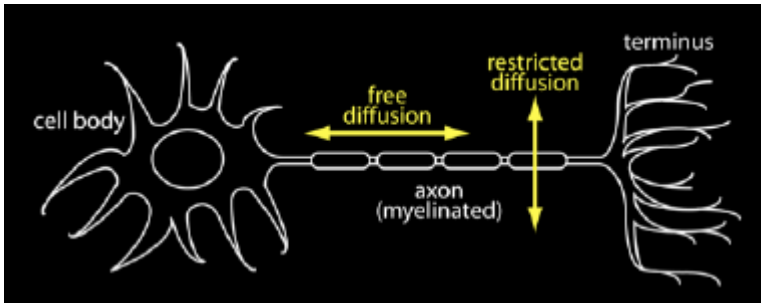

Figure 2.3: Diffusion restrictions imposed by an axon.

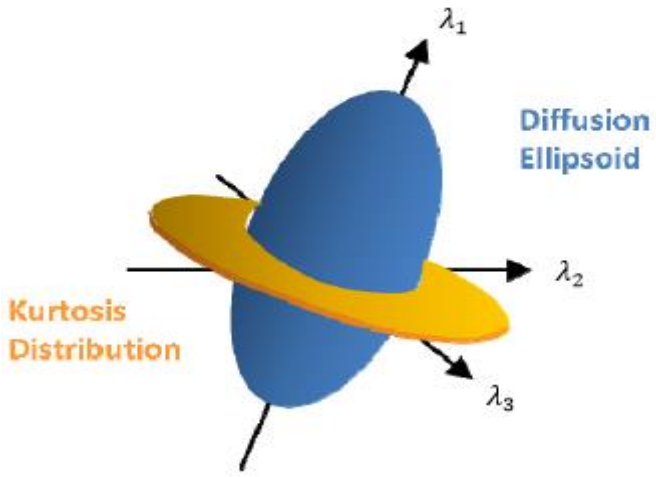

Figure 2.4**:** An illustration of the diffusion and kurtosis distribution in the 3D system defined by diffusion eigenvectors ($\boldsymbol{\lambda_1}$, $\boldsymbol{\lambda_2}$, $\boldsymbol{\lambda_3}$). The diffusion distribution is an ellipsoid (blue) with the principle direction pointing at $\boldsymbol{\lambda_1}$. The kurtosis distribution, from a simplified point of view, is like a pancake (yellow) with higher kurtosis along radial direction of the diffusion ellipsoid, indicating restricted diffusion. Taken from [5].

Despite being model-independent and therefore physically well-defined, both the diffusion coefficient and the diffusional kurtosis demand an underlying model for water diffusion in order to understand the information they hold regarding tissue structure.

The simplest model is the multiple compartment model with no water exchange. This model attempts to characterize water diffusion in brain by considering multiple compartments that represent diverse microstructures such as extracellular and intracellular spaces and white matter tracts. Although there are models that cover the flow of water from one compartment to the other such as the two-comparment model with water exchange, also reffered to as the Kärger model [7], and the one-dimensional model with barriers [10], this work will adopt the multiple compartment model instead [6].

2.3.1 Multiple compartment model

The PDF that describes the Gaussian diffusion of a water molecule within a single compartment is given by

$$P(r,t) = \frac{1}{(4\pi t)^{3/2}|D|^{1/2}} e^{-\boldsymbol{r}^T \boldsymbol{D}^{-1} \frac{r}{4t}} \tag{9}$$

with $\boldsymbol{D}$ representing the diffusion tensor.

If one considers N compartments, equation (9) is generalized to

$$P(r,t) = \sum_{m=1}^{N} f_m \frac{1}{(4\pi t)^{3/2}|D^{(m)}|^{1/2}} e^{-\boldsymbol{r}^T \boldsymbol{D^{(m)}}^{-1} \frac{r}{4t}} \tag{10}$$

with $D^{(m)}$ being diffusion tensor and f_m the water fraction for the m-th compartment [19].

Note that the sum of all water fractions must add up to 1.

$$\sum_{m=1}^{N} f_m = 1 \tag{11}$$

The total diffusion is the weighted sum of the diffusion coefficients of all the N compartments.

$$D(\boldsymbol{n}) = \sum_{m=1}^{N} f_m D^{(m)}(\boldsymbol{n}) \tag{12}$$

As for the diffusional kurtosis, equations (7), (8) and (10) give

$$K(\boldsymbol{n}) = 3\frac{\delta^2 D(\boldsymbol{n})}{[D(\boldsymbol{n})]^2} \tag{13}$$

with the term $\delta^2 D(\boldsymbol{n})$ being the variance of the diffusion coefficient given by

$$\delta^2 D(\boldsymbol{n}) = \sum_{m=1}^{N} f_m \left\{ \left[D^{(m)}(\boldsymbol{n}) - D(\boldsymbol{n}) \right]^2 \right\} \tag{14}$$

This way, the multiple compartment model, in its attempt to explain the meaning of diffusional kurtosis, gives a qualitative definition for this metric as stated in [6]: *"kurtosis is a measure of the heterogeneity of the diffusion environment".*

2.4 MRI signal

Having discussed the theory behind diffusion and kurtosis, the respective tensors and the adopted multi compartment model for water diffusion in brain, it is necessary to elaborate on the relationship to the measured MRI signal.

An example of a very accurate yet very demanding, not only in terms of hardware but also because of the long acquisition times, technique for characterizing water diffusion in brain is the q-space approach. It essentially fully determines the PDF which makes it possible to calculate the kurtosis directly from equation (4). However, this technique is not easily incorporated into clinical imaging protocols, as the b-values are very large. Another powerful and demanding technique is diffusion spectrum imaging (DSI), which is a 6D MRI technique similar to q-space. It also uses large b-values (e.g. $b = 17000\ \mathrm{s/mm^2}$) and long acquisition times (e.g. 25 min.) [11].

Diffusion kurtosis imaging, on the other hand, is a less demanding technique that consists on an extension of traditional diffusion tensor imaging.

2.4.1 DKI as an extension of DTI

The key factor in DTI is that the diffusion process produces an attenuation of the measured echo signal. This attenuation, given by equation (16) is a function of the b-value, which, for a Stejskal-Tanner sequence is defined by

$$b = \gamma^2 G^2 \delta^2 (\Delta - \frac{\delta}{3}) \tag{15}$$

with γ being the proton gyromagnetic ratio, G is the amplitude of the diffusion sensitizing magnetic field gradient pulses, δ the duration of the gradient pulses and Δ the time interval between centers of gradient pulses. The b-value is altered by varying G.

The dependence of the diffusion-weighted signal on the b-value is given by

$$\ln[S(b)] = \ln[S_0] - bD_{app} + O(b^2) \tag{16}$$

where $S(b)$ and S_0 are echo magnitudes of the diffusion weighted and non-diffusion-weighted ($b = 0$) signals respectively [20], and D_{app} is the 'apparent' diffusion coefficient which, for the non-exchanging multiple compartment described above, is essentially the true diffusion coefficient D for a diffusion time $t = \Delta$ with an infinitesimal pulse duration ($\delta \to 0$) [6]. Moreover, if the $O(b^2)$ term is made negligible by using sufficiently small b-values, the following approximation is valid

$$\ln[S(b)] \approx \ln[S_0] - bD(t) \tag{17}$$

This equation would be exact should the diffusion's PDF be purely Gaussian, and it could be rewritten as

$$S(b) = S_0 e^{-bD} \tag{18}$$

Since equation (17) has two unknowns, at least two b-values are needed in order to estimate D. If exactly two b-values are used, the following closed form solution arises

$$D \approx \frac{1}{b_2 - b_1} \ln\left[\frac{S(b_1)}{S(b_2)}\right] \tag{19}$$

DKI is built upon DTI by considering the contribution of the $O(b^2)$ term in equation (16) which concerns the diffusional kurtosis. This translates into the following cumulant expansion [21]

$$\ln[S(b)] = \ln[S_0] - bD_{app} + \frac{1}{6} b^2 D_{app}^2 K_{app} + O(b^3) \tag{20}$$

with K_{app} being the 'apparent' diffusional kurtosis which, as mentioned above, for the adopted multiple compartment model, is the true diffusional kurtosis K.

In addition to this and if, once again, the chosen b-values are small enough to assure that the $O(b^3)$ term is negligible, equation (16) may be rewritten as

$$\ln[S(b)] = \ln[S_0] - bD + \frac{1}{6}b^2D^2K \tag{21}$$

or

$$S(b) = S_0 e^{-bD+\frac{1}{6}b^2D^2K} \tag{22}$$

Similarly to DTI, in DKI the range of b-values must be so that precision is assured whilst maintaining the $O(b^3)$ term meaningless. However, the cumulant expansion of the diffusion-weighted signal allows for higher values.

The expansion of equation (16) adds a new unknown to the estimation problem. Thus, to reach a closed-form solution, there must be at least three b-values. Should that be the case, the following expressions are applicable

$$D \approx \frac{(b_3 + b_1)D^{(12)} - (b_2 + b_1)D^{(13)}}{(b_3 - b_2)} \tag{23}$$

and

$$K \approx 6\frac{D^{(12)} - D^{(13)}}{(b_3 - b_2)D^2} \tag{24}$$

with

$$D^{(1i)} = \frac{ln\left[\frac{S(b_1)}{S(b_i)}\right]}{(b_i - b_1)} \tag{25}$$

2.4.1.1 b-value range

To guarantee accurate parameter estimation, an upper bound for the b-value must be set for DTI as well as for DKI. The DKI model is the better approximation to the true diffusion-weighted signal as shown in figure 2.5. This is due

to the consideration of the $O(b^2)$ term whose contributions grow with increasing b-values.

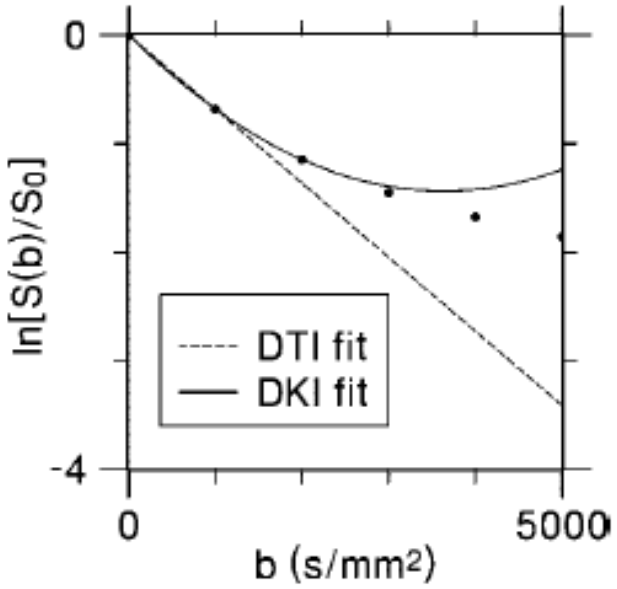

Figure 2.5: Comparison of DTI and DKI fitting models. For DTI, the logarithm of diffusion-weighted signal intensity (circles) as a function of the b-value is fit, for small b-values, to a straight line. In brain, this fit is often based on the signal for b = 0 and b = 1000 s/mm2. For DKI, the logarithm of the signal intensity is fit, for small b-values, to a parabola. In brain, this fit may be based on the signal for b = 0, b = 1000, and b = 2000 s/mm2. Taken from [6].

For DTI, the maximum b-value that assures precision whilst guaranteeing small contributions from the $O(b^2)$ term is about 1000 s/mm^2[6].

As for DKI, by assuming $S(b)$ as a monotonically decreasing function of b, which is empirically true for biological tissues [6], the maximum allowed value for b that assures this is calculated as

$$\frac{\partial}{\partial b}\left(-bD + \frac{1}{6}b^2D^2K\right) \leq 0 \tag{26}$$

which leads to

$$b < \frac{3}{DK} \tag{27}$$

Typical values in brain are around $D \approx 1\mu m^2/ms$ and $K \approx 1$, thus, the maximum b-value should be somewhere around 3000 s/mm^2 [19].

2.5 Algorithm for DKI computation

There are several ways to estimate the diffusion and kurtosis tensors $\boldsymbol{D}$ and $\boldsymbol{K}$. Methods such as unconstrained nonlinear least squares (UNLS) and unconstrained linear least squares (ULLS) have been used and were shown not to provide good tensor estimations [16].

In this work, the tensor estimation problem will be cast as a linear least squares with linear constrains safeguarding the physical and biological meaning of directional diffusivities and kurtoses. The formulation of the problem is presented ahead.

2.5.1 Formulation

The diffusion-weighted signal for a given direction $\boldsymbol{n}$ and b-value b given by equation (21) may be rewritten using equations (7) and (8) as

$$\ln\left[\frac{S(\boldsymbol{n},b)}{S_0}\right] = -b\sum_{i=1}^{3}\sum_{j=1}^{3} n_i n_j D_{ij} + \frac{1}{6}b^2\overline{D}^2\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3}\sum_{l=1}^{3} n_i n_j n_k n_l W_{ijkl} \tag{28}$$

Note that both D_{ij} and W_{ijkl} are symmetric as to the interchangeability of its indices, thus, the diffusion tensor $\boldsymbol{D}$ has 6 independent degrees of freedom, whereas the kurtosis tensor $\boldsymbol{W}$ has 15 [19]. Hence, there are $6 + 15 = 21$ parameters to be estimated.

The set of linear constraints is given by the following equations

$$D(\boldsymbol{n}) \geq 0 \tag{29}$$

$$K_{max}(\boldsymbol{n}) \geq K(\boldsymbol{n}) \geq K_{min} \tag{30}$$

where K_{max} denotes the maximum allowed value for kurtosis and may be expressed as

$$K_{max}(\boldsymbol{n}) = \frac{C}{b_{max} D(\boldsymbol{n})}, \qquad 3 \geq C \geq 0 \tag{31}$$

by rearranging equation (27). The variation on C regulates the level of restriction applied to the kurtosis values, nevertheless, C is usually set to 3 so as to be consistent with equation (27).

As for the minimum value for kurtosis (K_{min}), although its theoretical minimum is -2, it is normally set to zero because, as predicted by equation (13), the multiple compartment diffusion model suggests a super-Gaussian displacement distribution, meaning $K \geq 0$. Also, all the empirical evidence in brain gathered so far never yielded negative kurtosis values [16].

Simply put, the estimation problem consists in finding $\boldsymbol{D}$ and $\boldsymbol{W}$ that not only assure the exactness of equation (28) but also meet the constraints in equations (29) and (30).

All in all, the problem is

$$\begin{gathered} \text{Minimize } \|\boldsymbol{AX} - \boldsymbol{B}\|^2 \\ \text{such that } \boldsymbol{CX} \leq \boldsymbol{d}, \end{gathered} \tag{32}$$

with

$$\boldsymbol{X} = \left[D_{11}, D_{22}, D_{33}, D_{12}, D_{13}, D_{23}, \overline{D}^2 W_{1111}, \dots, \overline{D}^2 W_{1122}, \dots, \overline{D}^2 W_{1233}\right]^T \tag{33}$$

being a vector with the 21 unknown parameters of $\boldsymbol{D}$ and $\boldsymbol{W}$, $\boldsymbol{A}$ a matrix defined as

$$\boldsymbol{A} = \begin{bmatrix} -b_1 A_D^{(1)} & \frac{1}{6} b_1^2 A_K^{(1)} \\ \vdots & \vdots \\ -b_M A_D^{(M)} & \frac{1}{6} b_M^2 A_K^{(M)} \end{bmatrix} \tag{34}$$

with M being the number of nonzero b-values. The row k of $A_D^{(i)}$ and $A_K^{(i)}$ is given by

$$\left[A_D^{(i)}\right]_k = \left[(n_{k1}^{(i)})^2, (n_{k2}^{(i)})^2, (n_{k3}^{(i)})^2, n_{k1}^{(i)}n_{k2}^{(i)}, n_{k1}^{(i)}n_{k3}^{(i)}, n_{k2}^{(i)}n_{k3}^{(i)}\right] \tag{35}$$

and

$$\begin{aligned}\left[A_K^{(i)}\right]_k = \Big[&\left(n_{k1}^{(i)}\right)^4, \left(n_{k2}^{(i)}\right)^4, \left(n_{k3}^{(i)}\right)^4, 4\left(n_{k1}^{(i)}\right)^3 n_{k2}^{(i)}, 4\left(n_{k1}^{(i)}\right)^3 n_{k3}^{(i)}, 4\left(n_{k2}^{(i)}\right)^3 n_{k1}^{(i)},\\ &4\left(n_{k2}^{(i)}\right)^3 n_{k3}^{(i)}, 4\left(n_{k3}^{(i)}\right)^3 n_{k1}^{(i)}, 4\left(n_{k3}^{(i)}\right)^3 n_{k2}^{(i)}, 6\left(n_{k1}^{(i)}\right)^2\left(n_{k2}^{(i)}\right)^2, 6\left(n_{k1}^{(i)}\right)^2\left(n_{k3}^{(i)}\right)^2,\\ &6\left(n_{k2}^{(i)}\right)^2\left(n_{k3}^{(i)}\right)^2, 12n_{k2}^{(i)}n_{k3}^{(i)}\left(n_{k1}^{(i)}\right)^2, 12n_{k1}^{(i)}n_{k3}^{(i)}\left(n_{k2}^{(i)}\right)^2, 12n_{k1}^{(i)}n_{k2}^{(i)}\left(n_{k3}^{(i)}\right)^2\Big]\end{aligned} \tag{36}$$

respectively, with $n_{kj}^{(i)}$ being the *j*-th component of the *k*-th gradient direction for the *i*-th *b*-value.

Additionally, the vector $\boldsymbol{B}$ is expressed as

$$\boldsymbol{B} = \left[\ln\left(\frac{S(\boldsymbol{n}_k^{(i)}, b_i)}{S_0}\right) \ldots \ln\left(\frac{S(\boldsymbol{n}_k^{(M)}, b_M)}{S_0}\right)\right]^T \tag{37}$$

where $S(\boldsymbol{n}_k^{(i)}, b_i)$ is the signal intensity for the *k*-th direction of the *i*-th *b*-value.

The linear constraints are described by matrix $\boldsymbol{C}$ which is built as follows

$$\boldsymbol{C} = \begin{bmatrix} -A_D & 0 \\ 0 & -A_K \\ -\dfrac{C}{b_{max}}A_D & -A_K \end{bmatrix} \tag{38}$$

where

$$A_D = \begin{bmatrix} A_D^{(1)} \\ \vdots \\ A_D^{(M)} \end{bmatrix} \tag{39}$$

and

$$A_K = \begin{bmatrix} A_K^{(1)} \\ \vdots \\ A_K^{(M)} \end{bmatrix} \tag{40}$$

Finally, vector $\boldsymbol{d}$ is given by

$$\boldsymbol{d} = \left[\boldsymbol{0}\ K_{min}D(\boldsymbol{n}_1^{(1)})^2 \ldots K_{min}D\left(\boldsymbol{n}_N^{(1)}\right)^2 \ldots K_{min}D\left(\boldsymbol{n}_1^{(M)}\right)^2 \ldots K_{min}D\left(\boldsymbol{n}_N^{(M)}\right)^2\right] \tag{41}$$

where **0** is a one-dimensional vector with N elements.

2.5.2 Solution

The exact solution to the described problem is attainable via quadratic programming using standard algorithms such as active set and interior-point [16]. However, in this work, the estimation of $\boldsymbol{D}$ and $\boldsymbol{K}$ will be executed using a heuristic constrained linear least squares algorithm (CLLS-H) described ahead.

Firstly, unlike the quadratic programming methods that not only allow any number of b-values but also accept different gradient directions for each of the b-values, the CLLS-H algorithm uses exactly two nonzero b-values and equal gradient directions for both of them, meaning

$$\boldsymbol{N}^{(1)} = \boldsymbol{N}^{(2)} = \{\boldsymbol{n}_1, \dots, \boldsymbol{n}_N\} \tag{42}$$

where N is the number of gradient directions. This way, there are $2\boldsymbol{N} + 1$ images for every slice, where the term $+1$ regards the b_0 image.

The algorithm starts by estimating $D(\boldsymbol{n}_i)$ and $K(\boldsymbol{n}_i)$ along each direction using equations (23) and (24) and assuming $b_1$to be the zero b-value and b_3 together with b_2 to be the nonzero b-values now denoted as b_2 and $b_1$respectively. Thus we have

$$D_i = \frac{b_2 D_i^{(1)} - b_1 D_i^{(2)}}{b_2 - b_1} \tag{43}$$

$$K_i = \frac{D_i^{(1)} - D_i^{(2)}}{(b_2 - b_1) D(n_i)^2} \tag{44}$$

with $D_i^{(1)}$ and $D_i^{(2)}$ calculated from equation (25).

The next step is to constrain D_i using a the following set of rules

$$\begin{aligned} &\text{If } D_i \leq 0, \text{ set } D_i = 0. \\ &\quad \text{Else, if } D_i^{(1)} < 0, \text{ set } D_i = 0. \end{aligned} \tag{45}$$

Else, if $D_i > 0$ and $K_i < K_{min}$,

If $K_{min} = 0$, set $D_i = D_i^{(1)}$.

Else, set $D_i = \frac{1}{x}\left[\sqrt{1 - \frac{2}{3}K_{min}b_1 D_i^{(1)}} - 1\right]$.

Else, if $D_i > 0$ and $K_i > K_{max}$ ($K_{max_i} = K_{max}(n_i)$) set $D_i = \frac{D_i^{(1)}}{1 - C\frac{b1}{6b_{max}}}$

where b_{max} is the largest b-value used.

The $\boldsymbol{D}$ tensor may now be estimated as

$$\hat{X}_D = A_D^{+} b_D \tag{46}$$

where $\hat{X}_D$ is the unconstrained estimate of the 6 independent indices of the diffusion tensor $\boldsymbol{D}$,

$$X_D = [D_{11}, D_{22}, D_{33}, D_{12}, D_{13}, D_{23}]^T \tag{47}$$

and

$$b_D = [D_1, \ldots, D_N] \tag{48}$$

The A_D^{+} term represents the pseudoinverse of the A_D matrix defined in equations (35) and (39). This pseudoinverse matrix A_D^{+} is simply a generalization of the inverse matrix ${A_D}^{-1}$ which is normally used to solve systems of linear equations by the least squares method. This matrix is necessary when A_D is not a square, invertible matrix, which is the case.

Following this step, the D_i set is re-estimated using the following relationship

$$b_D^{(R)} = A_D \hat{X}_D \tag{49}$$

This re-estimation as a noise removal effect on the D_i set by assuring its consistency with $\hat{X}_D$ [16].

At this point, we can re-estimate K_i

$$K_i^{(R)} = \begin{cases} 6\dfrac{D_i^{(R)} - D_i^{(2)}}{b_2\left(D_i^{(R)}\right)^2} & if\ D_i^{(R)} > 0 \\ 0 & otherwise \end{cases} \tag{50}$$

Every $K_i^{(R)}$ is further constrained by the following set of rules

$$\begin{aligned} &\text{If } K_i^{(R)} > K_{max}^{(R)}, \text{ set } K_i^{(R)} = K_{max}^{(R)}. \\ &\text{If } K_i^{(R)} < K_{min}, \text{ set } K_i^{(R)} = K_{min}. \end{aligned} \tag{51}$$

The previous steps assure that the calculated diffusion coefficients are positive and the diffusional kurtoses lie within a physically acceptable range. If the calculated value for the diffusion coefficient is negative, that very coefficient as well as the kurtosis are automatically set to zero. Consequently, the effects of noise, motion and imaging artifacts that cause the misestimation are reduced considerably [6].

Lastly, the kurtosis tensor $\boldsymbol{K}$ is estimated as

$$\hat{X}_K = \frac{1}{\overline{D}^2} A_K^+ b_K \tag{52}$$

where $\hat{X}_K$ denotes the ULLS estimate of $\boldsymbol{K}$, the vector b_K is acquired as follows, from the re-estimated the K_i values

$$b_K = \left[\left(D_i^{(R)} \right)^2 K_1^{(R)}, \dots, \left(D_N^{(R)} \right)^2 K_N^{(R)} \right]^T \tag{53}$$

and, similarly to A_D^+, A_K^+ represents the pseudoinverse of the matrix A_K, which is described by equations (36) and (40).

2.6 Standard DTI metrics

Having estimated the diffusion and kurtosis tensors, the requested metrics may be computed. To do so, the coordinate system of each voxel is set to a reference frame that diagonalizes $\boldsymbol{D}$. The ensuing step is to get the diffusion tensor eigenvalues λ_1, λ_2, and λ_3, which are ordered as

$$\lambda_1 > \lambda_2 > \lambda_3. \tag{54}$$

At this point, all the standard DTI quantities may be calculated.

2.6.1 Mean diffusion (MD)

Mean diffusion, calculated as (55), represents the overall displacement of the water molecules and is independent of the orientation of the reference frame [22].

$$MD = \frac{\lambda_1 + \lambda_2 + \lambda_3}{3} \tag{55}$$

2.6.2 Axial diffusion (AD)

Axial diffusion ($D_{||}$) measures the diffusion along the main axis of the diffusion ellipsoid and is therefore given by

$$D_{||} = \lambda_1 \tag{56}$$

Axial diffusion is of the utmost importance because, for example, in white matter, as the main axis of the diffusion ellipsoid is aligned with the fibre direction, the axial diffusion is simply the diffusion value along the fibres. This way, it is believed that it may shed a light on axonal integrity [23].

2.6.3 Radial diffusion (RD)

Radial diffusion ($D_{\perp}$) represents the diffusion along the axis orthogonal to the main axis of the ellipsoid, thus, it is calculated as the average between λ_2 and λ_3, and, following the previous example, in white matter, it quantifies the diffusion across the fibers and may help to study myelin integrity [23].

$$D_{\perp} = \frac{\lambda_2 + \lambda_3}{2} \tag{57}$$

2.6.4 Fractional anisotropy (FA)

The last of the standard DTI quantities is fractional anisotropy. This metric is crucial in tractography as it quantifies the level of diffusion anisotropy in a range that goes from zero (meaning there is no isotropic diffusion, $\lambda_1 = \lambda_2 = \lambda_3$) to one (fully isotropic diffusion $\lambda_1 \gg \lambda_2 = \lambda_3 = 0$). This way, structures with high anisotropic properties such as white matter fibres show FA levels close to one, while highly isotropic tissues have a FA close to zero.

$$FA = \sqrt{\frac{3}{2}}\sqrt{\frac{(\lambda_1 - MD)^2 + (\lambda_2 - MD)^2 + (\lambda_3 - MD)^2}{{\lambda_1}^2 + {\lambda_2}^2 + {\lambda_3}^2}} \tag{58}$$

2.7 DKI metrics

As for DKI quantities, some more intricate calculations are demanded.

2.7.1 Mean kurtosis (MK)

Mean kurtosis, similarly to mean diffusion, is the overall kurtosis in all directions of the diffusion ellipsoid, and is calculated as

$$\begin{aligned} MK = {} & F_1(\lambda_1, \lambda_2, \lambda_3)\widetilde{W}_{1111} + F_1(\lambda_2, \lambda_1, \lambda_3)\widetilde{W}_{2222} + F_1(\lambda_3, \lambda_2, \lambda_1)\widetilde{W}_{3333} \\ & + F_2(\lambda_1, \lambda_2, \lambda_3)\widetilde{W}_{2233} + F_2(\lambda_2, \lambda_1, \lambda_3)\widetilde{W}_{1133} \\ & + F_2(\lambda_3, \lambda_2, \lambda_1)\widetilde{W}_{1122} \end{aligned} \tag{59}$$

where $\widetilde{W}_{ijkl}$ are the components of the kurtosis tensor in the reference frame that diagonalizes the diffusion tensor, and are calculated as follows

$$\widetilde{W}_{ijkl} = \sum_{i\prime=1}^{3}\sum_{j\prime=1}^{3}\sum_{k\prime=1}^{3}\sum_{l\prime=1}^{3} e_{i\prime i}e_{j\prime j}e_{k\prime k}e_{l\prime l}W_{i\prime j\prime k\prime l\prime} \tag{60}$$

with e_{ij} being the *i-th* component of the eigenvector corresponding to λ_j, and $W_{i\prime j\prime k\prime l\prime}$ the component of $\boldsymbol{K}$ for indices (i, j, k, l) [24].

The functions F_1 and F_2 are given by

$$\begin{aligned} F_1(\lambda_1, \lambda_2, \lambda_3) = {} & \frac{(\lambda_1 + \lambda_2 + \lambda_3)^2}{18(\lambda_1 - \lambda_2)(\lambda_1 - \lambda_3)}\left[\frac{\sqrt{\lambda_2\lambda_3}}{\lambda_1}R_F\left(\frac{\lambda_1}{\lambda_2}, \frac{\lambda_1}{\lambda_3}, 1\right)\right. \\ & \left. + \frac{3{\lambda_1}^2 - \lambda_1\lambda_2 - \lambda_1\lambda_3 - \lambda_2\lambda_3}{3\lambda_1\sqrt{\lambda_2\lambda_3}}R_D\left(\frac{\lambda_1}{\lambda_2}, \frac{\lambda_1}{\lambda_3}, 1\right) - 1\right] \end{aligned} \tag{61}$$

$$F_2(\lambda_1,\lambda_2,\lambda_3)=\frac{(\lambda_1+\lambda_2+\lambda_3)^2}{3(\lambda_2-\lambda_3)^2}\left[\frac{\lambda_2+\lambda_3}{\sqrt{\lambda_2\lambda_3}}R_F\left(\frac{\lambda_1}{\lambda_2},\frac{\lambda_1}{\lambda_3},1\right)+\frac{2\lambda_1-\lambda_2-\lambda_3}{3\sqrt{\lambda_2\lambda_3}}R_D\left(\frac{\lambda_1}{\lambda_2},\frac{\lambda_1}{\lambda_3},1\right)-2\right] \tag{62}$$

where R_F and R_D are Carlson's elliptic integrals [25], which are defined as

$$R_F(x,y,z)=\frac{1}{2}\int_0^\infty[(t+x)(t+y)(t+z)]^{-1/2}dt \tag{63}$$

and

$$R_D(x,y,z)=\frac{3}{2}\int_0^\infty[(t+x)(t+y)]^{-1/2}(t+z)^{-3/2}dt \tag{64}$$

It is important to acknowledge the singularities that these expressions hold when $\lambda_1=\lambda_2$ for F_1, and $\lambda_2=\lambda_3$ for F_2. These singularities may be removed as explained ahead.

2.7.2 Axial kurtosis (AK)

Axial kurtosis ($K_{||}$) , is the diffusional kurtosis in the directions of λ_1. In the reference frame that diagonalizes $\boldsymbol{D}$ is given by

$$K_{||}=\frac{(\lambda_1+\lambda_2+\lambda_3)^2}{9{\lambda_1}^2}\widetilde{W}_{1111} \tag{65}$$

2.7.3 Radial kurtosis (RK)

Finally, the radial kurtosis, in agreement with radial diffusion, represents the diffusional kurtosis perpendicular to the direction of λ_1 and is calculated by the following equations

$$K_\perp=G_1(\lambda_1,\lambda_2,\lambda_3)\widetilde{W}_{2222}+G_1(\lambda_1,\lambda_3,\lambda_2)\widetilde{W}_{3333}+G_2(\lambda_1,\lambda_2,\lambda_3)\widetilde{W}_{2233} \tag{66}$$

where G_1 and G_2 are given by

$$G_1(\lambda_1,\lambda_2,\lambda_3) = \frac{(\lambda_1+\lambda_2+\lambda_3)^2}{18\lambda_2(\lambda_2-\lambda_3)^2}\left(2\lambda_2 + \frac{{\lambda_3}^2 - 3\lambda_2\lambda_3}{\sqrt{\lambda_2\lambda_3}}\right) \tag{67}$$

$$G_2(\lambda_1,\lambda_2,\lambda_3) = \frac{(\lambda_1+\lambda_2+\lambda_3)^2}{3(\lambda_2-\lambda_3)^2}\left(\frac{\lambda_2+\lambda_3}{\sqrt{\lambda_2\lambda_3}} - 2\right) \tag{68}$$

Analogously to F_1 and F_2, both G_1 and G_2 have singularities, more specifically when $\lambda_2 = \lambda_3$.

2.7.4 Singularities in F_1, F_2, G_1 and G_2

The mean kurtosis may be calculated from equation (59) using the components of the kurtosis tensor in the reference frame that diagonalizes the diffusion tensor, or, it may simply be calculated from the original kurtosis tensor as

$$MK = \sum_{i,j,k,l=1}^{3} A_{ijkl} W_{ijkl} \tag{69}$$

where

$$A_{ijkl} = \frac{\overline{D}^2}{4\pi}\int_0^{\pi} d\theta \int_0^{2\pi} d\phi \, \frac{n_i n_j n_k n_l}{\left(\sum_{p,q=1}^{3} n_p n_q D_{pq}\right)^2} \sin\theta \tag{70}$$

with n_i being the i-th element of the direction vector $\boldsymbol{n}$.

However, numerical computation of A_{ijkl} is very demanding, therefore its implementation was discarded. The faster approach to this problem is to consider, as in equation (59), the reference frame that diagonalizes the diffusion tensor. Thus equation (69) is redefined as

$$MK = \sum_{i,j,k,l=1}^{3} A_{ijkl} \widetilde{W}_{ijkl} \tag{71}$$

and, this time, A_{ijkl} are given by

$$A_{ijkl}(\lambda_1, \lambda_2, \lambda_3)$$
$$= \frac{\overline{D}^2}{4\pi} \int_0^{\pi} d\theta \int_0^{2\pi} d\phi \frac{n_i(\theta,\phi) n_j(\theta,\phi) n_k(\theta,\phi) n_l(\theta,\phi) \sin\theta}{(\lambda_1 \sin^2\theta \cos^2\phi + \lambda_2 \sin^2\theta \sin^2\phi + \lambda_3 \cos^2\theta)^2} \tag{72}$$

with

$$\begin{aligned} n_1(\theta,\phi) &= \sin\theta \cos\phi, \\ n_2(\theta,\phi) &= \sin\theta \sin\phi, \\ n_3(\theta,\phi) &= \cos\theta. \end{aligned} \tag{73}$$

As A_{ijkl} is invariant with respect to permutation of its indices [16], as for the kurtosis tensor, only the 15 independent components need to be considered. What is more, from equation (70), it is possible to prove that

$$\begin{aligned} A_{1112} = A_{1113} = A_{1123} = A_{1222} = A_{1223} = A_{1233} = A_{1333} = A_{2223} \\ = A_{2333} = 0 \end{aligned} \tag{74}$$

leaving only 6 of the 15 original components. But the remaining 6 components may be reduced as well considering the following symmetries

$$A_{1111}(\lambda_1, \lambda_2, \lambda_3) = A_{2222}(\lambda_2, \lambda_1, \lambda_3) = A_{3333}(\lambda_3, \lambda_2, \lambda_1) \tag{75}$$

and

$$A_{2233}(\lambda_1, \lambda_2, \lambda_3) = A_{1133}(\lambda_2, \lambda_1, \lambda_3) = A_{1122}(\lambda_3, \lambda_2, \lambda_1). \tag{76}$$

Considering the relationships between A_{ijkl}'s of different indices described above, one needs only to compute A_{1111} and A_{2233} in order to calculate the mean kurtosis. Hence the mean kurtosis may be calculated as

$$\begin{aligned} MK = A_{1111}(\lambda_1, \lambda_2, \lambda_3)\widetilde{W}_{1111} + A_{1111}(\lambda_2, \lambda_1, \lambda_3)\widetilde{W}_{2222} + \\ A_{1111}(\lambda_3, \lambda_2, \lambda_1)\widetilde{W}_{3333} + 6A_{2233}(\lambda_1, \lambda_2, \lambda_3)\widetilde{W}_{2233} + \\ 6A_{2233}(\lambda_2, \lambda_1, \lambda_3)\widetilde{W}_{1133} + 6A_{2233}(\lambda_3, \lambda_2, \lambda_1)\widetilde{W}_{1122}. \end{aligned} \tag{77}$$

This formula for calculating the mean kurtosis makes it possible to avoid the singularities predicted in equations (61) and (62). For instance, when $\lambda_1 = \lambda_2$ or $\lambda_2 = \lambda_3$, F_1 has a singularity that can be eliminated by computing

$$A_{1111}(\lambda_1, \lambda_1, \lambda_3) = 3A_{2233}(\lambda_3, \lambda_1, \lambda_1) \text{ or } A_{1111}(\lambda_1, \lambda_2, \lambda_1) = 3A_{2233}(\lambda_2, \lambda_1, \lambda_1) \tag{78}$$

When $\lambda_2 = \lambda_3$, F_2 also has a singularity which is resolved by computing $A_{2233}(\lambda_1, \lambda_3, \lambda_3)$. This computation may be conducted using the following expression

$$A_{2233}(\lambda_1, \lambda_3, \lambda_3) = \frac{(\lambda_1 + 2\lambda_3)^2}{144{\lambda_3}^2(\lambda_1 - \lambda_3)^2}\left[\lambda_3(\lambda_1 + 2\lambda_3) + \lambda_1(\lambda_1 - 4\lambda_3)\alpha\left(1 - \frac{\lambda_1}{\lambda_3}\right)\right] \tag{79}$$

where

$$\alpha(x) = \begin{cases} \frac{1}{\sqrt{x}}\text{arctanh}\left(\sqrt{x}\right), & if\ x > 0 \\ \frac{1}{\sqrt{-x}}\text{arctanh}\left(\sqrt{-x}\right), & if\ x < 0 \end{cases} \tag{80}$$

Should $\lambda_1 = \lambda_2 = \lambda_3$ (isotropic diffusion), the singularity is easily eliminated as

$$A_{1111}(\lambda_1, \lambda_1, \lambda_1) = \frac{1}{5} \text{ and } A_{2233}(\lambda_1, \lambda_1, \lambda_1) = \frac{1}{15} \tag{81}$$

The calculation of radial kurtosis relies on equations (67) and (68) which also present singularities when $\lambda_2 = \lambda_3$. Thus, it is necessary to have an alternative set of calculations to allow bypassing these situations. The following alternative route follows the same idea applied in resolving the singularities for F_1 and F_2.

In much the same way as in equation (69), radial kurtosis may be defined as

$$K_{\perp} = \sum_{i,j,k,l=1}^{3} C_{ijkl}\widetilde{W}_{ijkl} \tag{82}$$

with

$$C_{ijkl}(\lambda_1, \lambda_2, \lambda_3) = \frac{\overline{D}^2}{2\pi}\int_0^{2\pi} d\phi \, \frac{n_i(\phi)n_j(\phi)n_k(\phi)n_l(\phi)}{(\lambda_2 \cos^2\phi + \lambda_3 \sin^2\phi)^2} \tag{83}$$

where

$$\begin{aligned} n_1(\phi) &= 0, \\ n_2(\phi) &= \cos\phi, \\ n_3(\phi) &= \sin\phi. \end{aligned} \tag{84}$$

As for A_{ijkl}, C_{ijkl} is also invariant with respect to permutations of its indices and, of the remaining 15 independent components, only 3 are to be considered because $C_{1111}, C_{1112}, C_{1113}, C_{1122}, C_{1123}, C_{1133}, C_{1222}, C_{1223}, C_{1233}, C_{1333}, C_{2223}$ and C_{2333} are all zero.

Additionally, as

$$C_{2222}(\lambda_1, \lambda_2, \lambda_3) = C_{3333}(\lambda_1, \lambda_3, \lambda_2), \tag{85}$$

one only needs to calculate C_{2222} and C_{2233} which are given by

$$C_{2222}(\lambda_1, \lambda_2, \lambda_3) = \frac{\overline{D}^2}{2\pi}\int_0^{2\pi} d\phi \, \frac{\cos^4\phi}{(\lambda_2 \cos^2\phi + \lambda_3 \sin^2\phi)^2} \tag{86}$$

and

$$C_{2233}(\lambda_1, \lambda_2, \lambda_3) = \frac{\overline{D}^2}{2\pi}\int_0^{2\pi} d\phi \, \frac{\cos^2\phi \sin^2\phi}{(\lambda_2 \cos^2\phi + \lambda_3 \sin^2\phi)^2} \tag{87}$$

These integrals lead to equation (64) for $G_1(\lambda_1, \lambda_2, \lambda_3) = C_{2222}(\lambda_1, \lambda_2, \lambda_3)$ and $G_2(\lambda_1, \lambda_2, \lambda_3) = 6C_{2233}(\lambda_1, \lambda_2, \lambda_3)$. Finally, the singularities in equations (67) and (68) when $\lambda_2 = \lambda_3$ are easily eliminated as C_{2222} and C_{2233} are condensed into the following expressions

$$C_{2222}(\lambda_1, \lambda_2, \lambda_3) = \frac{(\lambda_1 + 2\lambda_2)^2}{24{\lambda_2}^2} \tag{88}$$

and

$$C_{2233}(\lambda_1, \lambda_2, \lambda_3) = \frac{(\lambda_1 + 2\lambda_2)^2}{72{\lambda_2}^2} \tag{89}$$

respectively [16].

2.8 Anomalous estimates of tensor eigenvalues

The diffusion tensor $\boldsymbol{D}$ is assumed to be positive definite, i.e. not having negative eigenvalues which make no physical sense as there is no such thing as negative diffusion. However, the presence of artifacts and noise in the measured diffusion-weighted signal may cause erroneous estimations of $\boldsymbol{D}$ which may violate its assumed positive definiteness, and negative values for λ_3 and even λ_2 may appear.

2.8.1 Fractional Anisotropy Anomaly

An obvious consequence that negative eigenvalues, resultant from misestimations of $\boldsymbol{D}$, yield is fractional anisotropy values outside the allowed range.

As mentioned before, fractional anisotropy is a metric that quantifies the level of diffusion anisotropy in a range that goes from zero to one. Howbeit, the presence of negative eigenvalues in equation (58) may lead to values for FA bigger than one, which makes no physical sense whatsoever as the maximum level of anisotropy is represented by a FA value of one.

2.8.2 Proposed Solutions

In order to identify the best way of eliminating the problem of non-positive definiteness of erroneous $\boldsymbol{D}$ estimations, Koay et al., explored two constrained

least squares methods, the constrained linear least squares (CLLS) and constrained nonlinear least squares (CNLS) methods, and three additional algebraic methods namely, the *ZERO* method, the *ABS* method and the *LLS II* method. And, apart from the latter, all the explored methods were effective in eliminating negative eigenvalues. The constrained least squares methods assured the positive definiteness of $\boldsymbol{D}$ by means of Cholesky parameterization, whereas, the algebraic methods simply replaced the negative eigenvalue with either its absolute value (*ABS*) or zero (*ZERO*) [26].

3

Methodology

Prior to describing the implementation of the developed plugin for DKI computation, some insight on OsiriX and the developed framework is essential.

In order to analyse the set of DWI's, one must know some key OsiriX objects. Firstly, the main objects are the *BrowserController* and the *ViewerController*, which are illustrated in figures 3.1 and 3.2 respectively. *BrowserController* is basically an interface with a vast set of options whose main purpose is to allow the importation and exportation of medical imaging studies as well as select them for visualization. It includes a database that is automatically updated whenever a new exam is uploaded onto a particular directory. Therefore, for the developed plugin to process a given set of DWI's, that set must be imported to the database in the *BrowserController*.

On the other hand, the *ViewerController* object is responsible for the visualization and manipulation of the selected studies.

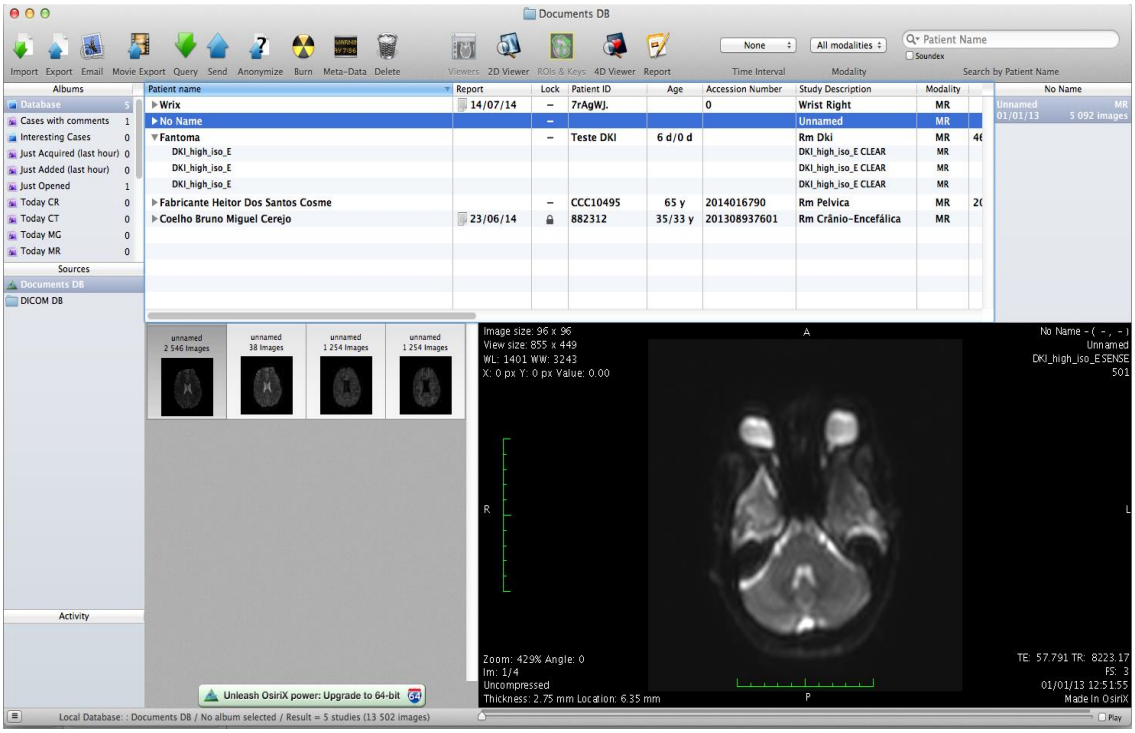

Figure 3.1: The *BrowserController* window providing a list of available studies in the database and sets of tumbnail images.

On opening the set of DWI's that are to be computed by the developed plugin, a new *ViewerController* object is created, and, within the *DCMView* object, which is the window containing the images, the user may run through the DWI's set, which, for the implemented CLLS-H algorithm, must have exactly two non-zero b-values.

Each slice of the selected study is a *DCMPix* object. This object is of the utmost importance for image manipulation as it holds crucial methods that allow, for example, the accessing of voxel information through pointers. Since there are $2\boldsymbol{N}+1$ images for every slice, each slice is an array of *DCMPix* objects. The structure of the study may be represented by the following matrix

$$\begin{matrix} (Sl_0, b_0, 1) & \cdots & (Sl_i, b_0, 1) & \cdots & (Sl_n, b_0, 1) \\ (Sl_0, b_1, \boldsymbol{N}) & \cdots & (Sl_i, b_1, \boldsymbol{N}) & \cdots & (Sl_n, b_1, \boldsymbol{N}) \\ (Sl_0, b_2, \boldsymbol{N}) & \cdots & (Sl_i, b_2, \boldsymbol{N}) & \cdots & (Sl_n, b_2, \boldsymbol{N}) \end{matrix} \tag{90}$$

where the last index of each triplet is the number of images with b-value b_j for the i-th slice Sl_i.

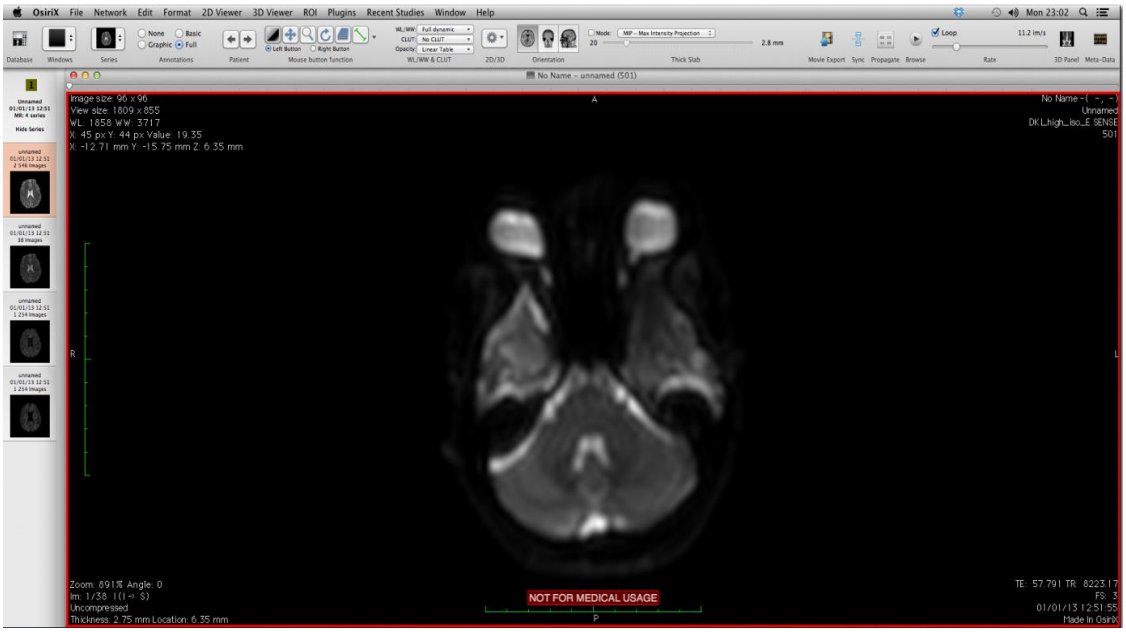

Figure 3.2: The *ViewerController*, containing basic image manipulation tools (definition of ROIs (region of interest), contrast adjustment, etc.) and the *DCMView* object.

3.1 Implementation

This section will cover the flow of the plugin since its selection up until the maps are generated and also elaborate on some developed code that would not be of immediate understanding to a developer.

The base class of the plugin is *DKIFilter*. It inherits from the *PluginFilter* class which is a basic OsiriX image processing plugin class.

As soon as the DKI plugin is selected, the method *filterImage* is activated and summons the user interface window shown in figure 3.3. This interface offers the user the possibility to select which maps are to be generated as well as define

the value for C, within the range mentioned in equation (31). It is also possible to set a threshold for voxel values under which no computation is conducted and all $D_i's$ and $K_i's$ are set to zero.

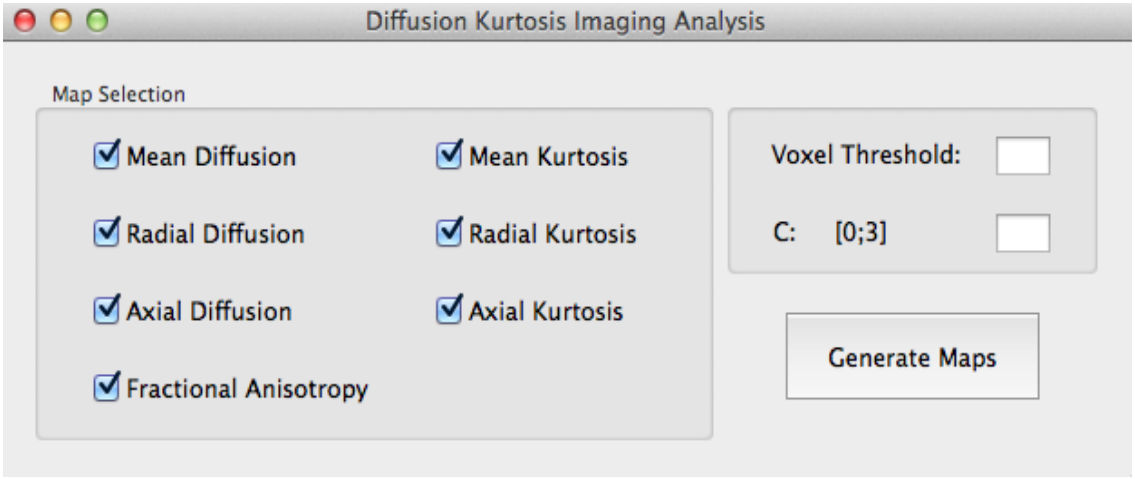

Figure 3.3: The DKI plugin graphical user interface (GUI).

On clicking the "Generate Maps" button, the user initiates the DKI computation by triggering the plugin's main function, *GenerateMaps*.

This function starts by reading the input values appointed by the user on the graphical interface for variables C and *MinPixValue*, where the latter stands for the threshold for voxel computation. The value for C is checked so as to assure it falls into the allowed range. Otherwise, the C is set to its default value of 3 following an alert box with the message "*Invalid C value. C was set to its default value of* 3.".

It is important to clarify that, although the set of rules (45) covers the possibility of $K_{min} > 0$ ($K_{min} < 0$, as mentioned before, is not predicted by the adopted model), the default value for K_{min} is set to zero and the user may not redefine this variable, because by altering K_{min}, isotropic diffusion is completely disregarded and a more detailed analysis of the physical meaning of this threshold is vital before the implementation of such option. Nevertheless, future works may elaborate on this subject and eventually allow the setting of a user-defined K_{min}.

Afterwards, the plugin reads the array (NSArray) containing the DWI's of the selected study and gets the number of slices (PIXCOUNT), the size of the images (SIZE) and the number of gradient directions (ndir).

At this point, the *b*-values and gradient directions are read from the image's metadata and their values are stored into two arrays using the functions in figure 3.4

```
[dirArray addObject:[img attributeArrayWithName:@"DiffusionGradientOrientation"]];
[bArray addObject:[img attributeArrayWithName:@"Diffusionb-value"]];
```

Figure 3.4: Code for acquiring the gradient directions and *b*-values from the metadata of the DWI set.

where img is the current image being read and "@"DiffusionGradientOrientation"" and "@"Diffusionb-value"" are the metadata tags for the gradient directions and *b*-values respectively.

Because the tags for these parameters may not be the same for different vendors, in order for the plugin to work regardless on the MRI vendor, a more versatile function must be developed. The implementation of such function is suggested in the *future work* section.

The *b*-values are subsequently compared and stored is ascending order as *B0, B1* and *B2*.

Having gathered all the gradient directions, memory must be allocated for two 2D arrays which will be populated according to equations (35, 36, 39 and 40) to represent both A_D and A_K matrices. To allocate the exact necessary memory, the dimensions of the matrices must be known. In this case the dimensions for A_D and A_K are [6 x ndir] and [15 x ndir] respectively.

The memory allocation for the A_K matrix is done as explained in figure 3.5.

```
AK = (double**) malloc(ndir*sizeof(double*));

for (int i = 0; i < ndir; i++)
{
    AK[i] = (double*) malloc(15*sizeof(double));
}
```

Figure 3.5: Memory allocation for the $\boldsymbol{A_K}$ matrix.

Note that any memory allocation for variables used ahead follow this same implementation.

After getting both matrices, similarly to the previous step, the plugin allocates memory for their respective pseudoinverse matrices A_D^+ and A_K^+ and then proceeds to their computation.

3.1.1 Computation of A_D^+ and A_K^+

The computation of the pseudoinverse matrices A_D^+ and A_K^+ was done using singular value decomposition (SVD). Considering

$$M = UWV^*$$

to be the SVD for matrix M, its pseudoinverse M^+ is easily calculated as

$$M^+ = VW^+U^*.$$

To compute the SVD for both A_D^+ and A_K^+, the plugin makes use of the svdcmp LAPACK routine. This function requires the following inputs: the matrix M, its dimensions, an empty array W and an empty 2D array V.

After execution, matrix M is replaced by U and both W and V^* are populated. Thereupon, it is necessary to transpose matrix U and get W^+ which is done simply by taking the reciprocal of each element, transposing the array and building a matrix whose diagonal is that very array. Afterwards, the only thing left to do is multiply the 3 matrices and get the pseudoinverse matrix M^+.

This multiplication is done by the function described in figure 3.6.

```
-(void) MULT:(double**)MULT M1:(double**)M1 M2:(double**)M2 M3:(double**)M3 n:(int)n
```

Figure 3.6: Function that given 3 matrices, multiplies them and returns the resultant matrix *MULT*. This function is used to calculate de pseudo-inverse matrices by multiplying the 3 matrices resultant from the SVD process.

where $MULT$ is the matrix resultant of the multiplication $M_1M_2M_3$ and n is its number of rows ($n = 6$ for A_D and $n = 15$ for A_K as predicted in equations 39, 40, 43 and 44).

Following the computation of A_D^+ and A_K^+, the program checks which maps were selected by the user in the graphical interface. If a map was chosen for computation, a new visualization window is opened by copying the original viewer. This is done using the *copyWindow* function which is basically a reformulated *copyViewerWindow* function (an OsiriX function defined in the *ViewerController.m* file) that does not require as much memory as the copied viewer contains only the images corresponding to the zero b-value and not the underlying images for the two nonzero b-values present in the original viewer. As a result, each selected quantity has its corresponding viewer, therefore the computation of those quantities may be conducted with the calculated values eventually being updated into these newly created windows generating the intended maps. The following piece of code (figure 3.7) exemplifies the described process

```
if(box1)
{
    MDviewer = [self copyWindow];
    MDpixlist = [MDviewer pixList:0];
    control = true;
}
else
{
    MDpixlist = AuxPixList;
}
```

Figure 3.7: The creation of a new viewer for a specific metric (in this case, the mean diffusion), should the user have selected it in the GUI.

Here, box1 is the boolean variable linked to the mean diffusion checkbox. If Mean diffusion was selected, a new viewer *MDviewer* is created and the MDpixlist array is simply the pixlist contained in that viewer. Otherwise, if the user decides not to choose a certain quantity in the GUI (e.g. mean diffusion), an auxiliary pixlist is assigned to the corresponding pixlist array (Mdpixlist in this case) because these arrays are a necessary input for the main function of the plugin, which will be described later on.

At this point, the plugin undergoes its main set of operations, which is, for every voxel in every slice, compute the quantities selected by the user and replace, in the according viewer, the voxel's original value with the computed one. This whole process is done by the *DKIcomputation* function described further

ahead.

In the ensuing steps, following the iteration through all the slices in the series, the selected maps are all updated into their respective viewers and the plugin finishes its execution.

The last step of the plugin execution is to free the memory allocated for A_D, A_K, A_D^+ and A_K^+. As an example, the following piece of code (figure 3.8) represents the freeing of allocated memory for the A_D matrix.

```
for(int i = 0; i < ndir; i++)
{
    free(AD[i]);
}
free(AD);
```

Figure 3.8: The release of the memory allocated for the $\boldsymbol{A_D}$ matrix.

Note that this process is equivalent for every obsolete variable throughout the program.

A simplified schematic view of the *GenerateMaps* function is presented in figure 3.9.

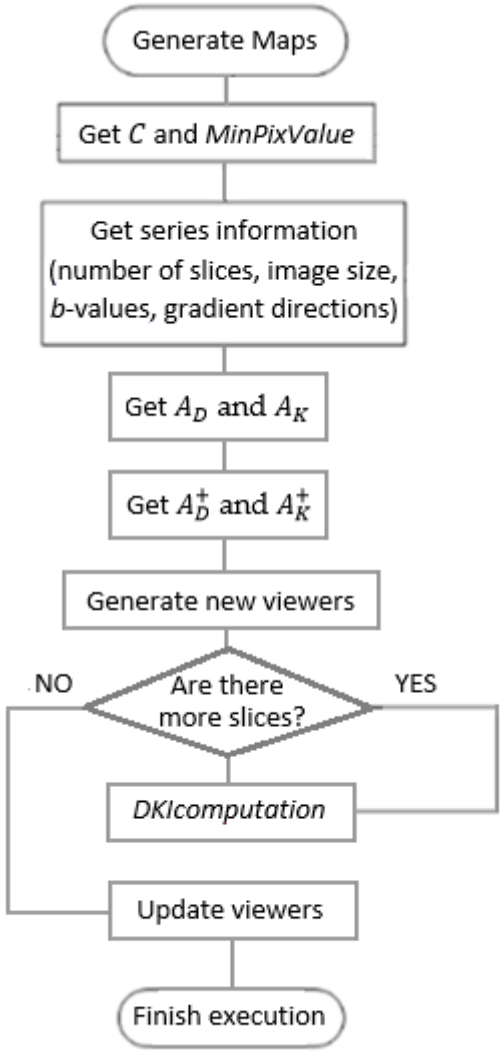

Figure 3.9: Flowchart of the plugin. The computation of DKI quantities is the central task of this tool, and is repeated for every slice in the series.

3.2 Voxel processing

The core of the *GenerateMaps* function is the computation of the desired quantities for every voxel in every slice. The operations necessary to compute these quantities are executed by the *DKIcomputation* function depicted schematically by figure 3.12.

This function is called for every slice in the series and, for a given slice, it iterates through its voxels using pointers.

Given a voxel in the image, the function starts by creating and arrays to hold both $D_i's$ and $K_i's$ as well as $D_i^{(2)}'s$ and calculating them. The set of $D_i^{(2)}'s$ is saved briefly because, contrary to $D_i^{(1)}'s$ that, according to equations (43) and (44) are only used for calculating $D_i's$ and $K_i's$, $D_i^{(2)}'s$ are used later on on the re-estimation of $K_i's$ as predicted in equation (50).

Because $D_i^{(1)}$ and $D_i^{(2)}$ are calculated from equation (25), its necessary to get, for direction i, both $S(b_1)$ and $S(b_2)$, which are the voxel values for the b-values b_1 and b_2 respectively. Figure 3.10 shows how the program runs through the DWI's of a given slice, with h being the index of the current slice and the pixList array holds all the images for a given direction and b-value.

```
currentPix0 = [[viewerController pixList: 0] objectAtIndex:h];

for(int t = 1; t <= ndir; t++)
{
    currentPix1 = [[viewerController pixList: t] objectAtIndex:h];
    currentPix2 = [[viewerController pixList: t + ndir] objectAtIndex:h];
```

Figure 3.10: This piece of code shows the *for* cycle responsible for iterating through the images respecting the order described in matrix (90).

The values of the voxels are subsequently retrieved by defining pointers to the images (figure 3.11).

```
fpix0 = [currentPix0 fImage];
fpix1 = [currentPix1 fImage];
fpix2 = [currentPix2 fImage];

float Sb0 = fpix0[x];
float Sb1 = fpix1[x];
float Sb2 = fpix2[x];
```

Figure 3.11: The declaration of pointers to the voxel currently being processed, where x is the index of the voxel being computed.

In the ensuing steps, the calculated $D_i's$ are constrained by undergoing the set of rules expressed in (45). The diffusion tensor $\boldsymbol{D}$ is subsequently calculated from the D_i set and matrix A_D^+ following equation (46). At this point, the eigenvalues and eigenvectors of the diffusion tensor may be acquired. This acquisition is discussed ahead.

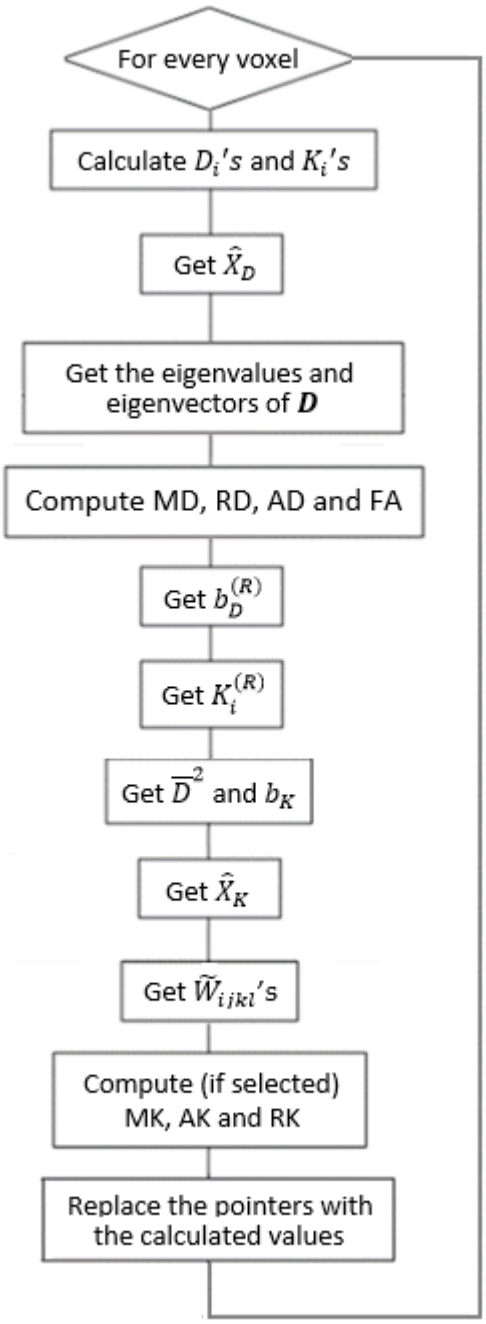

Figure 3.12: Flowchart for the *DKIcomputation* function

3.2.1 Computation of D's eigenvalues and eigenvectors

The eigenvalues and eigenvectors of the tensor are obtained using the function *dsyev*, another LAPACK routine. This function computes all eigenvalues and, optionally, the eigenvectors of a real symmetric matrix, in this case $\boldsymbol{D}$.

One of the necessary inputs to this function is a double precision array A containing the elements of the tensor. The population of A requires knowledge of how the tensor is organized which is dictated by equation (47). Following this equation and since the tensor is symmetric, one may illustrate, as in figure 3.13, the organization of the tensor elements in terms of their index in the X_D array.

0	3	4
3	1	5
4	5	2

Figure 3.13: The index of the diffusion tensor elements in the $\boldsymbol{X_D}$ array.

Therefore, the array A may be populated as seen in figure 3.14.

```
A[0] = Xd[0];
A[1] = Xd[3];
A[2] = Xd[4];
A[3] = Xd[3];
A[4] = Xd[1];
A[5] = Xd[5];
A[6] = Xd[4];
A[7] = Xd[5];
A[8] = Xd[2];
```

Figure 3.14: The population of A in the order described in figure 3.13.

Another important feature of the *dsyev* routine is that, after computation, the eigenvectors are stored in array A. The order in which the eigenvectors are stored was tested by auxiliary functions and it was determined that the eigenvectors are set in an order such that their corresponding eigenvalues form a descending pattern. This explains the code in figure 3.15, which is basically an inversion of the order imposed by the *dsyev* function.

```
for(int i = 0; i < n; i++)
{
    for(int h = 0; h < n ; h++)
    {
        eigenvectors[i][h] = a[3*abs(i-2)+h];
    }
}
```

Figure 3.15: The orderly population of the eigenvector's array.

The reason behind this inversion is that, as the eigenvalues of the diffusion tensor are considered to be in a descending order indicated in equation (54), their respective eigenvectors, by being in the same order, facilitate further calculations.

After computation, should a eigenvalue be negative, its value is instead set to their absolute value following the *ABS* algebraic method described in section 2.8. After this correction, the eigenvalues are ordered again following equation (54) and so are the corresponding eigenvectors.

3.2.2 DTI and DKI metrics computation

At this point, all the standard DTI metrics are calculated according to equations (55, 56, 57 and 58). These calculations are very straightforward and even if the user did not select any of these metrics in the GUI, their increment to processing time is negligible.

After the computation of MD, RD, AD and FA, the program checks if any of the maps corresponding to MK, RK or AK were selected by the user, and, if the user chose at least one of those DKI metrics, the program carries on to the estimation on the kurtosis tensor $\boldsymbol{K}$.

Should that be the case, the D_i set is re-estimated according to equation (49), and this re-estimated set is used together with the $D_i^{(2)}$ set to also re-estimate the $K_i's$ by applying the rule in equation (50) followed by the restrictions in (51).

The determining of $\overline{D}$ and b_K is then carried out as both are essential to the computing of $\hat{X}_K$ which is done immediately after, following equation (52).

In order to calculate either MK, RK or AK, one must have every $\widetilde{W}_{ijkl}$, which, as mentioned before, are the elements of the kurtosis tensor in the reference frame that diagonalizes the diffusion tensor. The calculation of these factors follows equation (60) and, to do so, one must have all the elements of the kurtosis tensor. For this reason, a new 4D array was created and populated to serve as the kurtosis tensor.

So as to populate this array, one must take into consideration the fact that the kurtosis tensor has only 15 independent components as it is symmetric with respect to interchange of its indices. To tackle this problem a four-layered cycle was used to cover every index of the tensor.

Within each iteration, so as to determine which of the 15 independent components is the correct fit, a switch statement was implemented. This switch statement's control variable is the sum of the indexes at the current iteration.

An example of one case of this switch statement is provided by figure 3.16.

```
case 2:
{
    if(i != 2 && j != 2 && k != 2 && l != 2)
    {
        W[i][j][k][l] = Xk[9];
        break;
    }
    else
    {
        W[i][j][k][l] = Xk[4];
        break;
    }
}
```

Figure 3.16: One of the cases of the switch statement declared in figure 3.16. In this case, the sum of the indices is 2 which occurs for both $\boldsymbol{W_{0002}}$ and $\boldsymbol{W_{0011}}$, thus, these two cases are distinguished by a subsequent if statement.

In this case, $i + j + k + l = 2$, which is true for both W_{0002} and W_{0011} (the indices start in 0 instead of 1), thus, these cases must be differentiated. The choosing between W_{0002} and W_{0011}, which correspond to the *4-th* and *9-th* component of the $\hat{X}_K$ array respectively, is determined by analysing which set of i, j, k and l contains twos.

Having populated the *4-th* order kurtosis tensor, the program may use the formula in equation (60) to calculate all $\widetilde{W}_{ijkl}$'s.

At this point, the program assigns one function for each selected DKI quantity and their computation proceeds.

Functions to compute not only Carlon's elliptic integrals (R_F and R_D) defined in equations (63, 64) but also both F_1 and F_2 are mandatory for calculating the mean kurtosis. The implemented function for calculating R_F and R_D is adapted from the *"Numerical Recipes in C"* [27] and a numerical check was performed to check the function's output against the values in the article *"Numerical computation of real or complex elliptic integrals"* [28].

Additionally, functions were developed for calculating F_1 and F_2 as well as G_1 and G_2. These functions take into account the singularities that occur when $\lambda_2 = \lambda_3$ for F_2, G_1 and G_2 and when $\lambda_1 = \lambda_2 = \lambda_3$ for both F_1 and F_2.

The computation of the mean kurtosis follows the code in figure 3.17. The function has a set of *if* statements to cover all the possible singularities that may occur. For the simple case when $\lambda_1 = \lambda_2 = \lambda_3$, the function follows equation (81), and, when $\lambda_1 \neq \lambda_2 \neq \lambda_3$, it follows equations (61) and (62) calculating both F_1 and F_2 with the *getFs* function. The boolean variable in the *getFs* function controls which F is to compute.

The three remaining cases are when $\lambda_1 = \lambda_2 \neq \lambda_3$, $\lambda_1 = \lambda_3 \neq \lambda_2$ and $\lambda_1 \neq \lambda_2 = \lambda_3$. These cases present different singularities that are tackled by functions *getA1111* and *getA2233*. These functions follow equations (72) and (79) respectively. The boolean variable in the *getA1111* function is used to differentiate the singularity that occurs when $\lambda_1 = \lambda_2$ from the one when $\lambda_2 = \lambda_3$.

The case when $\lambda_1 = \lambda_2$ was implemented as, for this specific case, the integral in equation (72) may be reduced to a simpler form. However, the singularity caused by $\lambda_2 = \lambda_3$ was not treated for the integral gets far more complicated to compute. Nevertheless, the development of a function to compute the integral is proposed in the future work chapter.

After having calculated all the quantities chosen by the user, the pointers for each quantity are given the respective computed value. Finally, the *DKIcomputation* function frees all the memory allocated in the present iteration and then moves on to the next voxel where the whole set of operations described above is

repeated. Once all the voxels in one slice have been treated, another slice is analysed and this process repeats until no slices are left.

```
if (A1 == A2 && A2 == A3)
{
    F1W1111 = Wijkl[0] * 1/5;
    F1W2222 = Wijkl[1] * 1/5;
    F1W3333 = Wijkl[2] * 1/5;

    F2W1122 = Wijkl[5] * 6/15;
    F2W1133 = Wijkl[4] * 6/15;
    F2W2233 = Wijkl[3] * 6/15;
}
else if (A1 != A2 && A2 != A3)
{
    F1W1111 = getFs(A1, A2, A3, true) * Wijkl[0];
    F2W2233 = getFs(A1, A2, A3, false) * Wijkl[3];

    F1W2222 = getFs(A2, A1, A3, true) * Wijkl[1];
    F2W1133 = getFs(A2, A1, A3, false) * Wijkl[4];

    F1W3333 = getFs(A3, A2, A1, true) * Wijkl[2];
    F2W1122 = getFs(A3, A2, A1, false) * Wijkl[5];
}
else if (A1 == A2)
{
    F1W1111 = getA1111(A1, A1, A3, true) * Wijkl[0];
    F1W2222 = getA1111(A1, A1, A3, true) * Wijkl[1];
    F1W3333 = getFs(A3, A2, A1, true) * Wijkl[2];
    F2W1122 = 6 * getA2233(A3, A2, A1) * Wijkl[5];
    F2W1133 = getFs(A2, A1, A3, false) * Wijkl[4];
    F2W2233 = getFs(A1, A2, A3, false) * Wijkl[3];
}
else if (A1 == A3)
{
    F1W1111 = getA1111(A1, A2, A1, false) * Wijkl[0];
    F1W2222 = getFs(A2, A1, A3, true) * Wijkl[1];
    F1W3333 = getA1111(A1, A2, A1, false) * Wijkl[2];
    F2W1122 = getFs(A3, A2, A1, false) * Wijkl[5];
    F2W1133 = 6 * getA2233(A2, A1, A3) * Wijkl[5];
    F2W2233 = getFs(A1, A2, A3, false) * Wijkl[3];
}
else
{
    F1W1111 = getFs(A1, A2, A3, true) * Wijkl[0];
    F1W2222 = getA1111(A2, A1, A2, false) * Wijkl[1];
    F1W3333 = getA1111(A1, A1, A3, true) * Wijkl[2];
    F2W1122 = getFs(A3, A2, A1, false) * Wijkl[5];
    F2W1133 = getFs(A2, A1, A3, false) * Wijkl[4];
    F2W2233 = 6 * getA2233(A1, A2, A3) * Wijkl[3];
}


MK = F1W1111 + F1W2222 + F1W3333 + F2W2233 + F2W1133 + F2W1122;
```

Figure 3.17: Function for computing MK. Various *if* statements are implemented so as to deal with any eventual singularities in computing $\boldsymbol{F_1}$ and $\boldsymbol{F_2}$.

3.3 Installation

The installation of the developed plugin is very straightforward since this project was developed using the OsiriX Plugin Generator, which is downloaded along the source code for Osirix. This way, after building the project in Xcode, a new executable with extension *.osirixplugin* is created in the project folder (figure 3.18) that, when ran, uploads the plugin into OsiriX. This concludes the installation after which the plugin is fully functional.

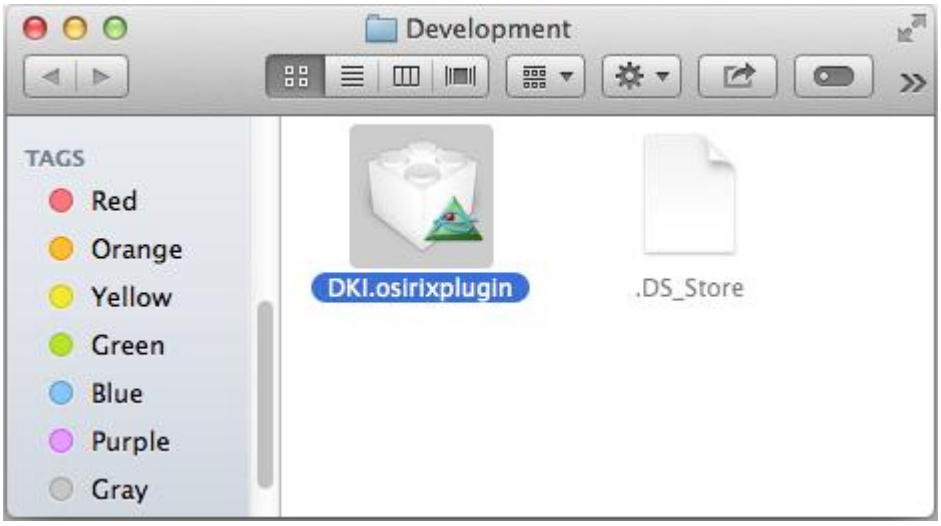

Figure 3.18: Plugin installation executable.

4

Results and Validation

4.1 Results

In order to test the developed plugin, one must perform MRI acquisitions which meet the requirements for DKI computation. However, before running the plugin on a DWI set, the two algebraic methods *ZERO* and *ABS* described in section 2.8.2, were compared by implementing them both and analysing the corrections they provide in the computed maps.

4.1.1 *ABS* and *ZERO* methods comparison

It is important to note that axial diffusion maps were not analysed in this process because the value for axial diffusion is simply the largest eigenvalue λ_1, that is only affected by these methods should all the eigenvalues be negative, which never happened for any DWI set analysed.

The comparison between these two methods was done by marking 50 voxels that violate the positive definiteness of $\boldsymbol{D}$ and correcting their inherent eigenvalues. Afterwards, the maps for mean and radial diffusion as well as fractional anisotropy were generated and the marked voxels were compared to their counterparts in the maps generated by the DKE software for the same DWI set using the CLLS-QP algorithm.

The results in table 4.1 show that, for radial diffusion, fractional anisotropy and mean kurtosis, the *ABS* method seems to produce better corrections than the *ZERO* method which only surpassed the former in the radial diffusion and radial kurtosis maps. As for axial kurtosis, the two methods produce very similar values.

	ZERO		ABS	
Map	**MSE**	**SD**	**MSE**	**SD**
MD	0.021798	0.133204	0.041533	0.184412
RD	0.038563	0.034884	0.023551	0.029371
FA	0.009424	0.013309	0.009011	0.012538
MK	20248.42	11945.24	240.9195	1092.671
RK	5.405671	2.823471	18682.15	92280.18
AK	0.099506	0.061458	0.099506	0.060845

Table 4.1: Mean squared error (MSE) and standard deviation (SD) of the corrected voxel values for the *ZERO* and *ABS* methods.

Notwithstanding the decent corrections that these methods provide for standard DTI metrics, the ensuing corrections on the kurtoses maps are not acceptable, for these algebraic methods, by slightly altering the wrongfully computed eigenvalues, induce some errors that may grow out of control in the subsequent complex calculations.

Even though these methods are not a valid option for tackling the non-positive definiteness problem, in the absence of a better option, the *ABS* method was implemented as it was still better than the *ZERO* method after all.

4.1.2 Unacceptable voxel values

While analysing the generated maps, it was observed that some (very few) voxels had extremely high values that fell in no way in the acceptable range observed in the maps generated by the DKE software. These voxels were therefore inspected in an attempt to unveil the reason behind the mentioned high values.

It was hypothesized that the non-positive definiteness of the diffusion tensor, as it has no physical meaning, was in the origin of such discrepant values.

This study was done by marking the voxels wherein the diffusion tensor was non-positive definite with point-type ROIs which were then saved and eventually imported to the generated maps. However, it was observed that the non-positive definite tensor estimations produced both acceptable and unacceptable values. Furthermore, it was also observed that the extremely high values where being calculated from positive definite tensor estimations as well as non-positive definite.

All in all, one may conclude that the extremely high values observed in the generated maps are not related to the positive definiteness of the diffusion tensor. These values may simply be a consequence of some tensor estimations which, although being positive definite, are not correct. These wrongful estimations are predicted due to the heuristic nature of the implemented algorithm.

To address the problem of the unacceptably high voxel value, a threshold should be imposed above which, all the values are neglected and set to a predefined value. The definition of such threshold is proposed in the future work section in chapter 5.

4.1.3 Generated maps for two sets of DWIs

It has been shown in chapter 2 that the diffusion tensor $\boldsymbol{D}$ has 6 degrees of freedom while the kurtosis tensor $\boldsymbol{K}$ has 15. As a result, in order to fully characterize these tensors, 21 parameters must be estimated. Therefore, any DKI computation entails having datasets with, at least 22 DWIs considering the unweighted image ($b = 0$). Furthermore, Jensen et al., 2010 state that there must be at least 3 distinct b-values and also, at least, 15 different gradient directions [6]. All the datasets processed by the developed plugin had b-values of 0, 1000 and 2000 $\mathrm{s/mm^2}$, which, assuming the validity of the adopted model (equation (21)), are the most efficient and advantageous choice [29].

Even though the minimum gradient directions is 15, when a dataset with 16 gradient directions was processed by the DKI plugin, the maps for mean, axial and radial kurtosis were very noisy. As for datasets with 33 gradient directions, the generated maps were cleaner and more enlightening. The reason behind this is that, the oversampling of the diffusion directions produce DKI estimates that are less influenced by motion artifacts which might be caused by underlying biological processes such as cardiac-induced pulsation in brain [6].

Prior to its validation, some DWI sets were processed by the developed plugin in order to obtain a qualitative and even semi-qualitative analysis of the generated maps.

The first set to be processed was obtained from the Center for Biomedical Imaging of the Medical Univeristy of South Carolina's website [32]. The chosen dataset was the *4D Nifti Image* as its DICOM was more complicated to upload as a single *4D* image into OsiriX.

The gradient directions for this DWI set were hardcoded into the plugin by a temporary auxiliary function, in order to overcome the absence of metadata tags with the same designation as in figure 3.4 (a function to enables the plugin to read the metadata regardless of the MRI vendor is proposed in the future developments section).

The image set was processed by both the DKE software and the developed DKI plugin. None of the options (special filtering, median filtering, co-registration and interpolation) offered by the DKE software were chosen, because none of them are included in the developed plugin. Also, the voxel threshold was set to zero for both estimators.

The generated maps are very similar as shown by figures 4.1, 4.2, 4.3, 4.4, 4.5, 4.6 and 4.7.

It is important to mention that, in the kurtosis maps computed by the developed plugin, some voxels appear black (their value is zero) whilst their counterparts in the DKE generated maps are do not appear so. The reason behind this

is that those voxels are violating either directional diffusivities, minimum directional kurtoses or maximum directional kurtoses [16].

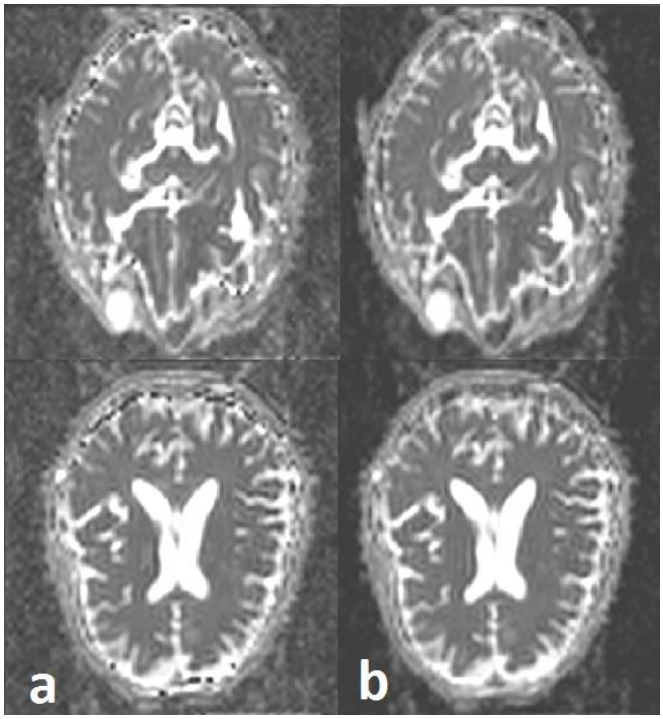

Figure 4.1: Mean diffusion maps computed by (a) DKI plugin, (b) DKE software, using the CLLS-H and CLLS-QP algorithms respectively.

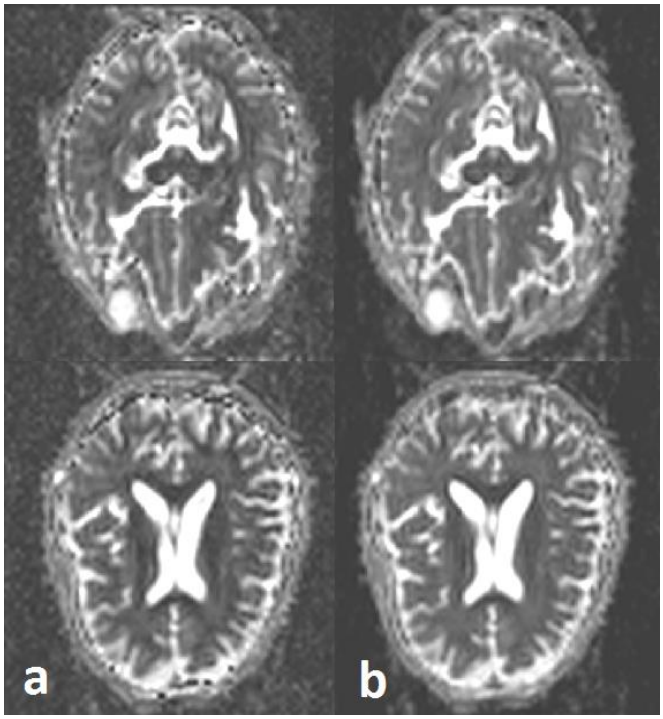

Figure 4.2: Radial diffusion maps computed by (a) DKI plugin, (b) DKE software, using the CLLS-H and CLLS-QP algorithms respectively.

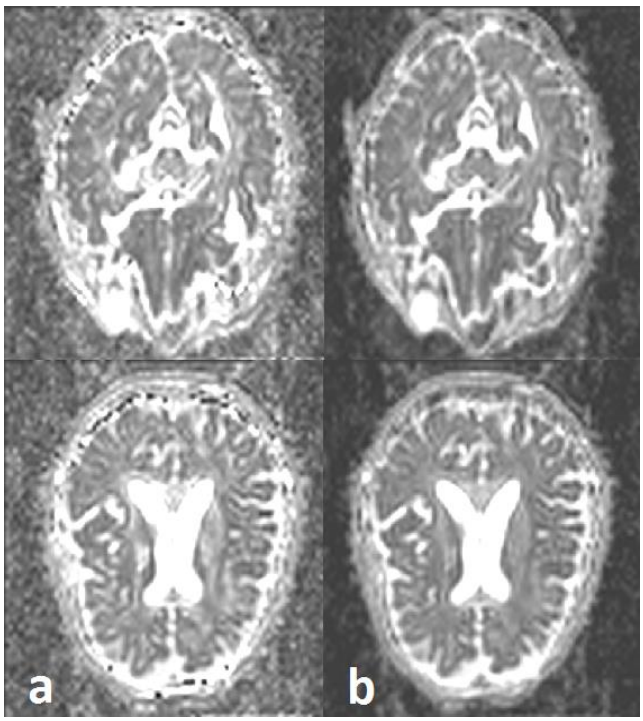

Figure 4.3: Axial diffusion maps computed by (a) DKI plugin, (b) DKE software, using the CLLS-H and CLLS-QP algorithms respectively.

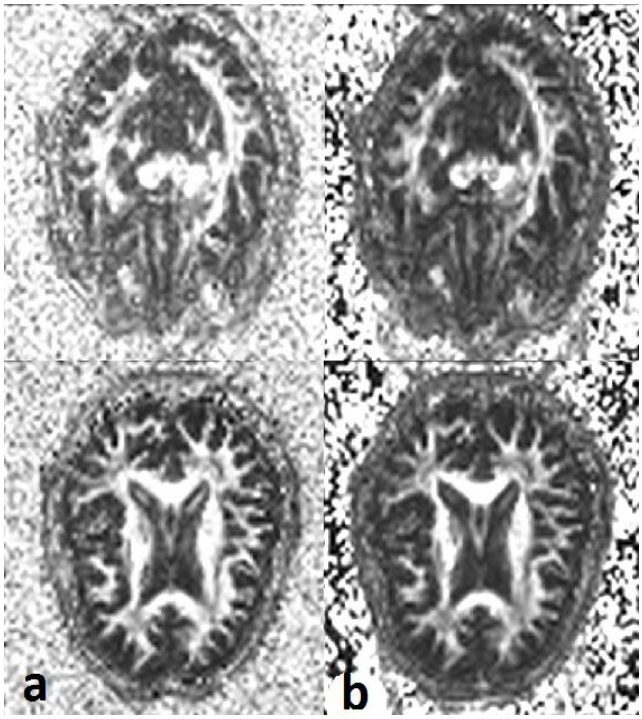

Figure 4.4: Fractional anisotropy maps computed by (a) DKI plugin, (b) DKE software, using the CLLS-H and CLLS-QP algorithms respectively.

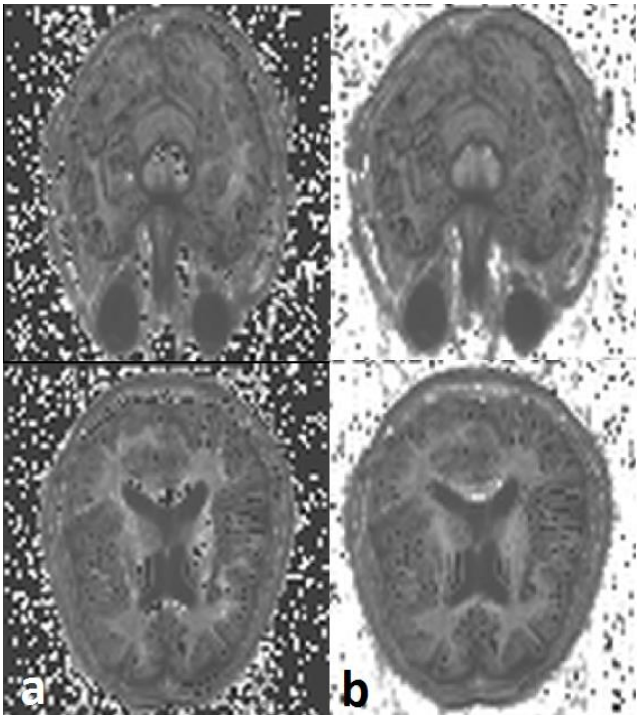

Figure 4.5: Mean kurtosis maps computed by (a) DKI plugin, (b) DKE software, using the CLLS-H and CLLS-QP algorithms respectively.

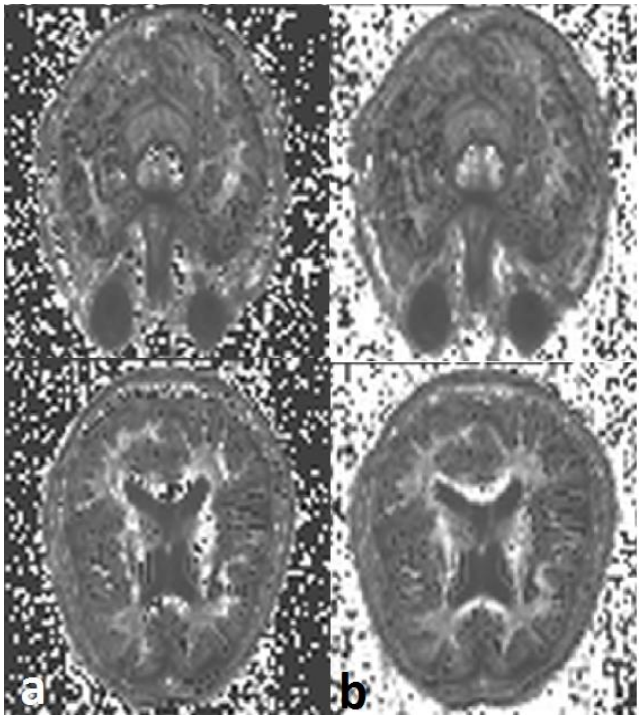

Figure 4.6: Radial kurtosis maps computed by (a) DKI plugin, (b) DKE software, using the CLLS-H and CLLS-QP algorithms respectively.

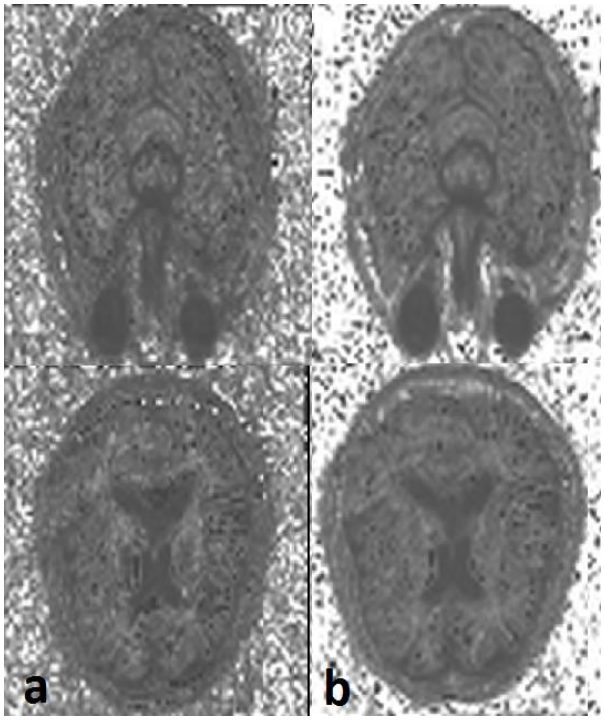

Figure 4.7: Axial kurtosis maps computed by (a) DKI plugin, (b) DKE software, using the CLLS-H and CLLS-QP algorithms respectively.

Despite being very similar, the maps have some differences that result from the different algorithms used. The maps for standard DTI metrics are almost identical between the two softwares. However, the slightest difference in the computed eigenvalues for $\boldsymbol{K}$ may snowball into bigger discrepancies in kurtosis metrics due to the complex calculations involved.

Because of this, the validation of the developed plugin will not consist on map comparison but on a process described in section 4.2.

Nevertheless, a simple comparison between the maps was conducted by drawing 50 ROIs (areas ranging from 20 to 60 mm^2) around for two different slices of the brain. The ROIs were drawn in the maps generated by the developed plugin as well as those generated by the DKE software, and their mean voxel value was compared. Two large ROIs (figure 4.8) were also drawn in these slices in order to provide an average value for the different metrics so that the mean squared error between the values in the smaller ROIs (figure 4.9) is meaningful.

The results shown in table 4.2 are satisfactory even though, once again, this analysis is only semi-quantitative as there are natural differences between the maps that result from the different algorithms used.

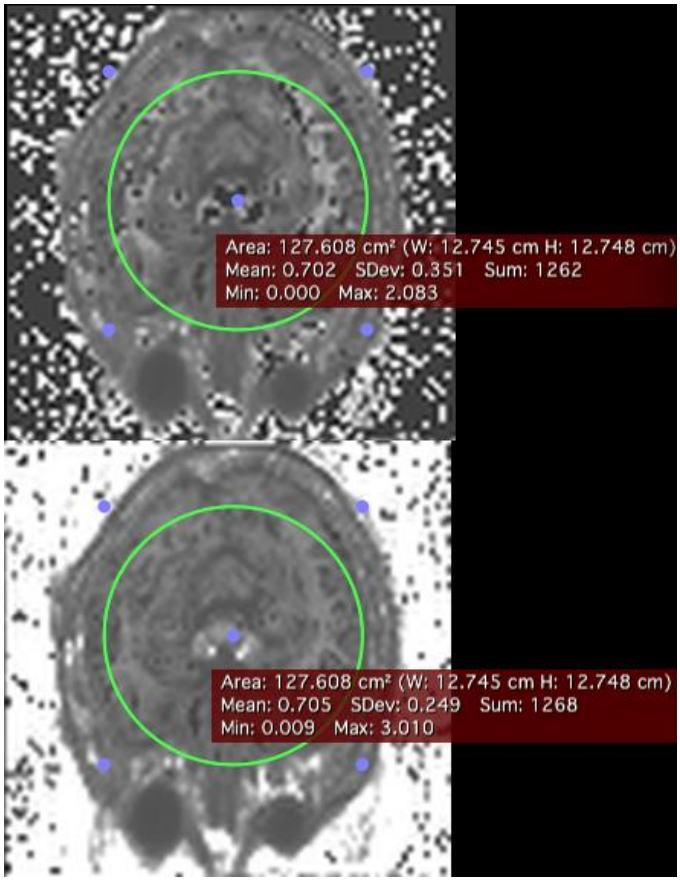

Figure 4.8: A large ROI containing the central part of the slice for the mean kurtosis maps.

Figure 4.9: Small ROI in mean diffusion maps.

	Average voxel value	MSE
MD	1.516	0.005
RD	1.272	0.011
AD	2.146	0.124
FA	0.312	0.006
MK	0.796	0.028
RK	9.821	0.297
AK	0.688	0.033

Table 4.2: The mean squared error between the ROIs drawn in the maps generated by the DKE software and the developed plugin. The average voxel value for each metric provides better perspective on the presented errors.

Another image set was processed by the developed plugin. This second DWI set was acquired in Hospital Santa Maria. The DKI acquisition was performed by a 3T Philips Achieva system with a 8-channel head coil. The DWI's were acquired, along 33 different gradient directions with b-values ($b_1 = 1000$ s/mm^2 and $b_2 = 2000$ s/mm^2), with a spin-echo sequence. Scan parameters were TR = 8223 ms, TE = 58, matrix = 96 × 96, FOV = 231 × 231 mm^2, 38 slices, slice thickness = 2.75 with no gap. The total acquisition time was of 9:11 minutes.

By using a Philips system, the gradient directions were not hardcoded into the plugin as it was possible to get them directly from the image's metadata using the code in figure 3.4.

The following figures show the generated maps for standard DTI metrics (figure 4.10) and DKI metrics (figure 4.11) for 3 slices of the brain.

The generated maps were presented to an MRI expert and a Doctor who, after a brief qualitative analysis, agreed were very satisfactory.

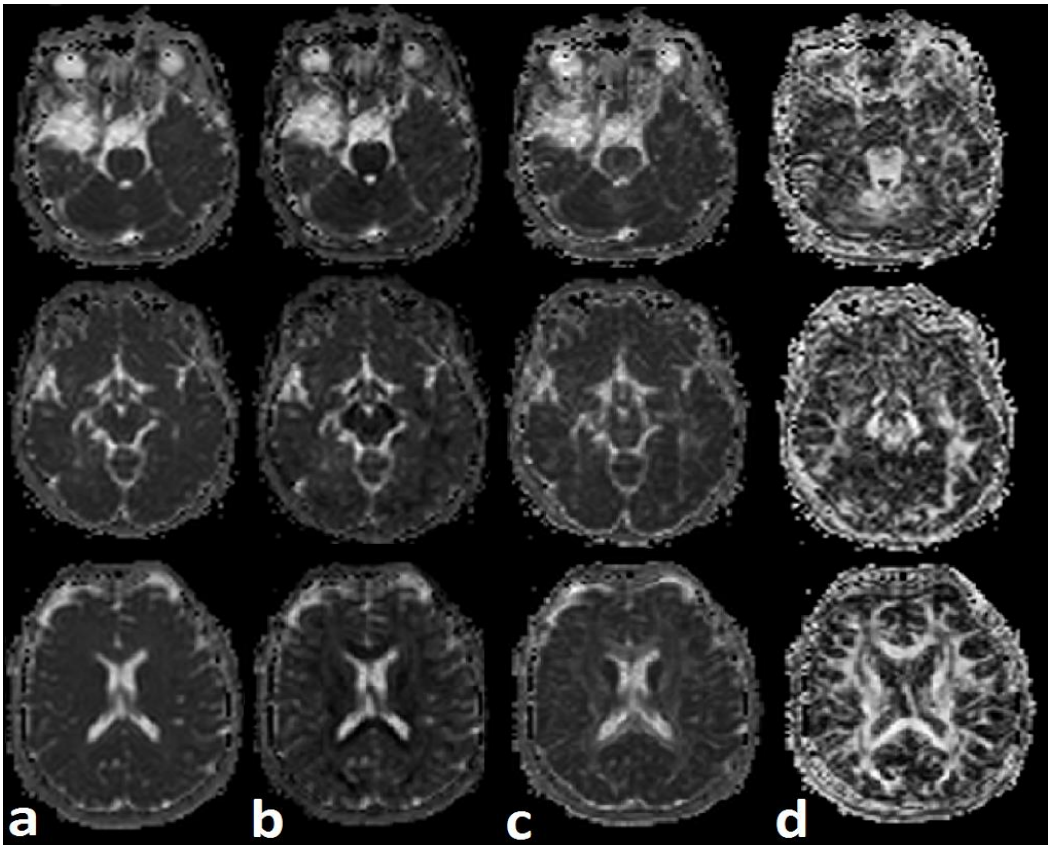

Figure 4.10: Maps generated by the plugin, (a) Mean diffusion; (b) Radial diffusion; (c) Axial diffusion; (d) Fractional Anisotropy.

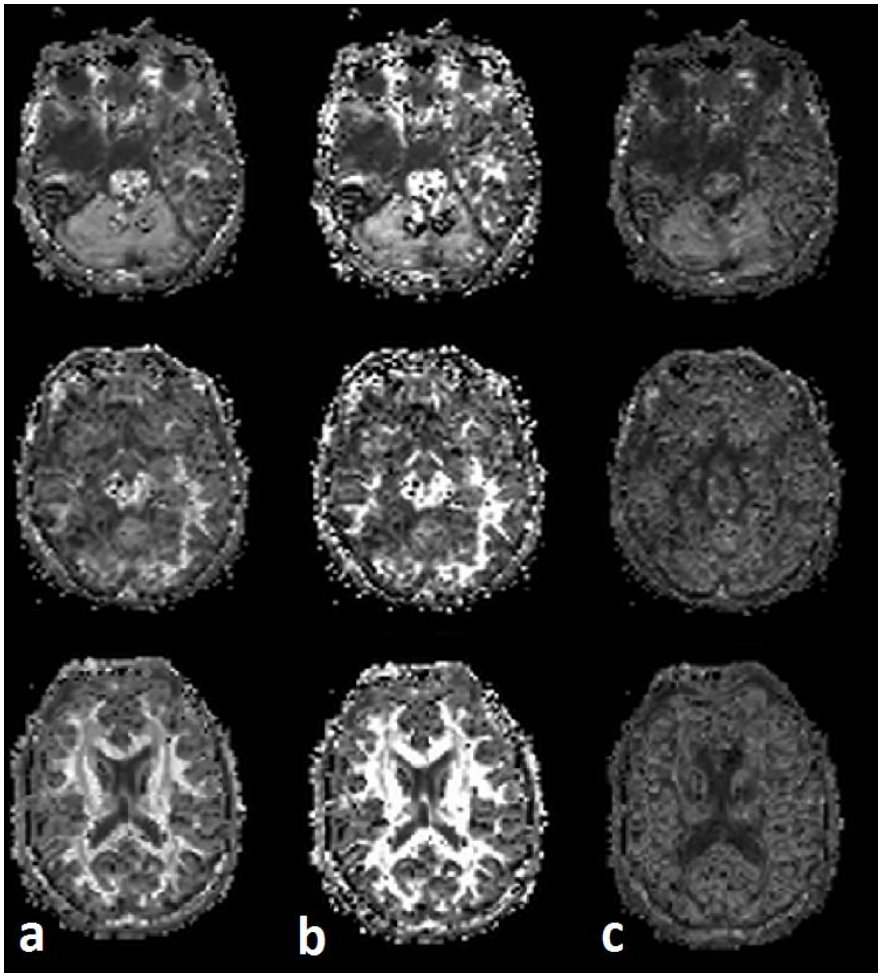

Figure 4.11: Maps generated by the plugin, (a) Mean kurtosis; (b) Radial kurtosis; (c) Axial kurtosis.

4.2 Validation

The validation of the presented plugin will follow the methods proposed by Fieremans et al., 2012 wherein a simple isotropic phantom composed of dairy cream is processed and subsequently analysed. The inherent methods proposed in those works were shown to provide nonzero kurtosis values that can be quantified and therefore used to verify the validity of the post-processing methods [30].

The phantom used for validation consisted of a 2.6 L cylindrical plastic vial filled with cream with a fat content of 35%, which is very similar to the 36% used by Fieremans et al., 2012. The cream underwent the following heat treatment before its storage in the phantom recipient:

- submersion in water bath at 80ºC;
- after reaching 60ºC the cream was kept an extra 10 minutes in the water bath;
- reaching a temperature of 74ºC the cream is stored at room temperature (18.5ºC) and left cooling overnight.

The following day, a MRI scan was performed on the phantom on a 3T Philips Achieva system with a 8-channel head coil. The DWI's were acquired, along 33 different gradient directions with b-values ($b_0 = 0\ \mathrm{s/mm^2}$, $b_1 = 1000\ \mathrm{s/mm^2}$ and $b_2 = 2000\ \mathrm{s/mm^2}$), with a spin-echo sequence. Scan parameters were TR = 2900 ms, TE = 105 ms, matrix = 96 × 96, FOV = 231 × 231 mm^2, 8 slices, slice thickness = 10 mm with no gap.

The heating treatment leads to an increase in the T_2-relaxation time of the fat protons. As a consequence, the resonance frequencies of fat and water are different which creates a visible separation between the two components. This separation between fat, water and cream (which is essentially water and fat) may be observed in the diffusion weighted images for ($b = 0$) shown in figure 4.12, similarly to the images acquired in [30].

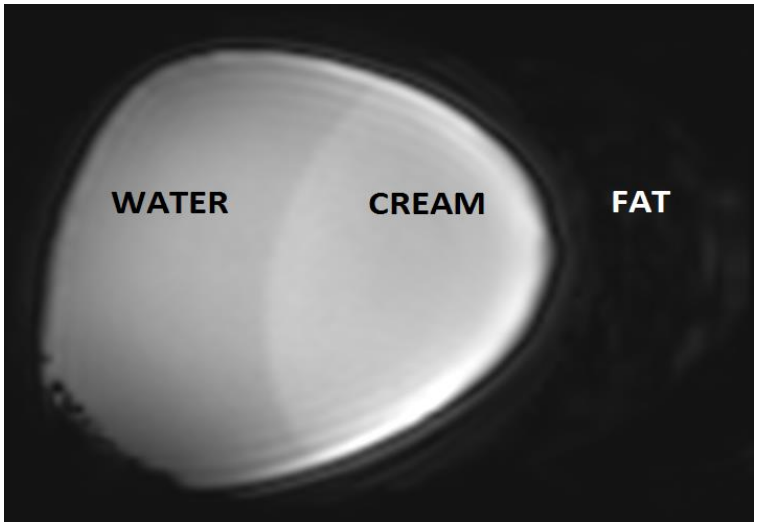

Figure 4.12: An example of an acquired image for ($\boldsymbol{b} = \mathbf{0}$). As predicted, the separation of the signal from fat, cream and water is visible due to the previous heating treatment.

After identifying the region corresponding to water, fat and both fat and water, regions of interest (ROIs) were drawn in those regions and, after computing the maps for both mean diffusion and mean kurtosis, every ROI was imported and the mean values for MD and MK were analysed and compared to the results obtained in [30]. The computed maps for MK are shown in figure 4.13.

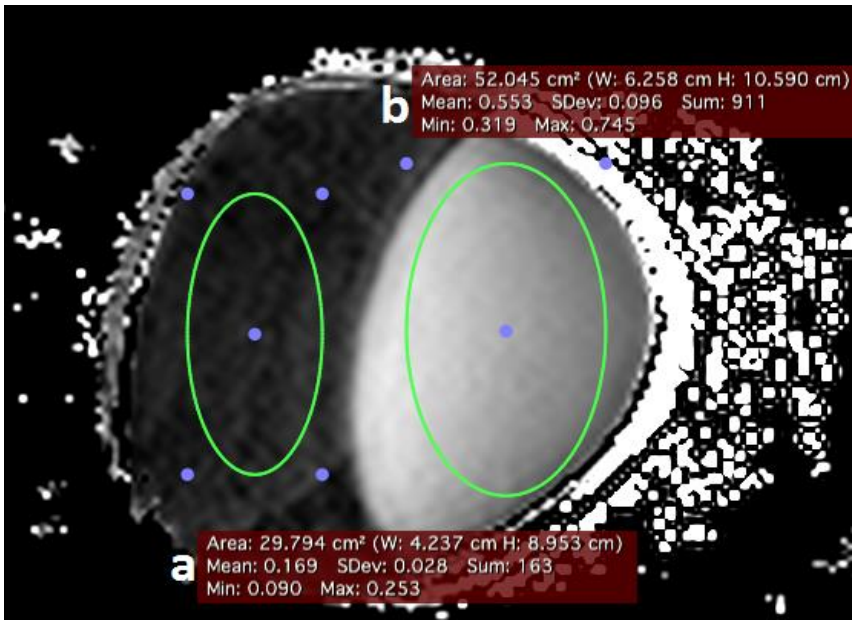

Figure 4.13: The mean kurtosis maps for the phantom and two ROIs, (a) water region and (b) cream region.

Even though the mean kurtosis values for water were around 0.17 which is close to the expected 0.15, the computed mean kurtosis values for the cream were around 0.55, which is not near the expected 1.2 presented in the literature. Although there are inumerous variables that may explain this discrepancy, the one that might have the strongest influence on the computed results is the size of the container, as it was very tightly fit into the head coil. This may have caused the bad signal acquisition in the fat region as seen in figure 4.12 and its subsequent computation in figure 4.13. Consequently, as the cream region is composed of both water and fat, the mean kurtosis values for the cream are wrongfully estimated. Another possible reason for this misestimation is that, as the separation of the water and fat signals is caused by a resonance process that produces a chemical shift which only happens for a specific bandwidth, the chosen bandwidth in the present acquisition might not be appropriate for the resonance phenomenon to occur.

The validation process was repeated with a 330 mL container and a bandwidth of 2345 Hz/pixel.

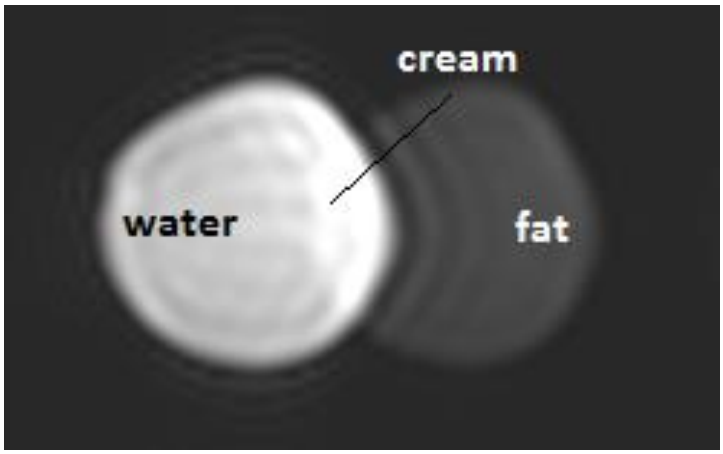

Figure 4.14: The image for ($\boldsymbol{b} = \mathbf{0}$) in the second validation attempt.

The separation of the signal from fat, cream and water in figure 4.14 is clearer than the previous validation attempt probably due to the more appropriate vial size and bandwidth applied. Nevertheless, there is still some acquisition artifacts (Nyquist ghost) that might influence the processing of the maps.

The computed maps for mean diffusion and mean kurtosis are shown in figures 4.15 and 4.16 respectively.

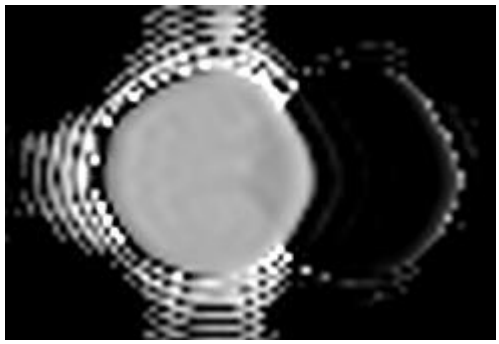

Figure 4.15: Computed MD map in the second validation attempt.

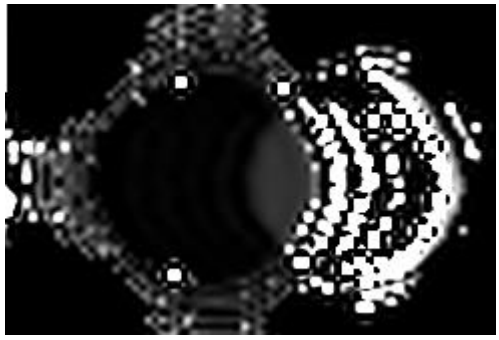

Figure 4.16: Computed MK map in the second validation attempt.

Despite the clearer separation of the 3 components, the computed maps are not as clean as those shown in the literature [30].

The analysis of the computed maps was done by drawing 50 ROIS with areas ranging from 20 to 30 mm^2. The results, presented in table 4.3, show that, for the mean diffusion values for both water and fat, fall into the expected range, as for the cream region, the values are not very satisfactory. Regarding the mean kurtosis maps, only the water region seems to be correctly processed as the values in the cream region are still below the expected range.

Even though the second validation attempt did not produce the expected values, this does not mean that the developed tool is not accurate as there are

several variables that were not be fully explored in the conducted validation processes. For instance, the dairy cream used for the phantom had a fat content of 35%, which is not exactly the same as the 36% used in the literature. Moreover, not only might the remaining constituents of the cream be different, but also, the acquisition was done using a different MRI system. All of this may explain the discrepancies observed.

		EXPECTED	AVERAGE	SD
MD	WATER	1.35 ± 0.02	1.37456	0.016606
	CREAM	1.08 ± 0.02	1.26044	0.181125
	FAT	0.01 ± 0.02	0.00944	0.007563
MK	WATER	0.15 ± 0.07	0.05064	0.012227
	CREAM	1.18 ± 0.04	0.79222	0.13867

Table 4.3: Average value of the voxels within the defined ROIs for the generated MD and MK maps.

5

Conclusion

5.1 Conclusion

The developed plugin for DKI computation fulfilled the proposed objectives as it was shown to be able to generate reliable maps for both the standard DTI metrics and for the DKI metrics as well. Its usage is very straightforward as the graphical interface is very simple and user friendly. The software was validated using methods described in the literature and, following the proposed validation and some further minor adjustments and improvements (discussed in the next section), it shall be fully functional and ready for implementation.

All in all, the plugin, by being integrated in the OsiriX framework, is a fast and simple tool that may be easily installed and used by medical personnel for gathering information about neural pathologies that are made evident only by DKI processing.

5.2 Future work

Although the developed plugin is very complete and shows accurate estimates for all the intended metrics, some improvements are in order. These improvements are:

- Develop a function to numerically compute the integral in equation (70) so as to eliminate the singularity that arises in equation (60) when $\lambda_2 = \lambda_3$.
- Implementation of an alternative method for dealing with bad tensor estimations which result in negative eigenvalues (described in chapter 2).
- Set a plausible threshold for voxel values in order to eliminate the very high voxels present in the computed maps.
- Develop a function to access the metadata and retrieve the required information regardless of the MRI vendor.
- Implement a multithreading architecture in order to achieve faster processing times.
- Implement the quadratic programming version of the constrained linear least squares algorithms for an exact estimation of the diffusion and kurtosis tensors to serve as a ground truth.

Bibliography

[1] J. H. Jensen, J. a. Helpern, A. Ramani, H. Lu, and K. Kaczynski, "Diffusional kurtosis imaging: The quantification of non-Gaussian water diffusion by means of magnetic resonance imaging," *Magn. Reson. Med.*, vol. 53, no. 6, pp. 1432–1440, Jun. 2005.

[2] P. J. Basser, J. Mattiello, and D. Lebihan, "MR Diffusion Tensor Spectroscopy and Imaging," vol. 66, no. January, 1994.

[3] J. E. Tanner, "Restricted Self-Diffusion of Protons in Colloidal Systems by the Pulsed-Gradient, Spin-Echo Method," *J. Chem. Phys.*, vol. 49, no. 4, p. 1768, 1968.

[4] T. Niendorf, R. M. Dijkhuizen, D. G. Norris, M. van Lookeren Campagne, and K. Nicolay, "Biexponential diffusion attenuation in various states of brain tissue: implications for diffusion-weighted imaging.," *Magn. Reson. Med.*, vol. 36, no. 6, pp. 847–57, Dec. 1996.

[5] J. Zhuo, "Diffusion Kurtosis Magnetic Resonance Imaging and its Application to Traumatic Brain Injury," University of Maryland, 2011.

[6] J. H. Jensen and J. a Helpern, "MRI quantification of non-Gaussian water diffusion by kurtosis analysis.," *NMR Biomed.*, vol. 23, no. 7, pp. 698–710, Aug. 2010.

[7] J. Kärger, "NMR SELF-DIFFUSION STUDIES IN HETEROGENEOUS SYSTEMS," *Adv. Colloid Interface Sci.*, vol. 23, pp. 129–148, 1985.

[8] J. J. H. A. Dmitriy A. Yablonskiy, G. Larry Bretthorst, "Statistical Model for Diffusion Attenuated MR Signal," *Magn. Reson. Med.*, vol. 50, no. 4, pp. 664–669, 2007.

[9] C. Liu, R. Bammer, B. Acar, and M. E. Moseley, "Characterizing non-Gaussian diffusion by using generalized diffusion tensors.," *Magn. Reson. Med.*, vol. 51, no. 5, pp. 924–37, May 2004.

[10] a L. Sukstanskii, D. a Yablonskiy, and J. J. H. Ackerman, "Effects of permeable boundaries on the diffusion-attenuated MR signal: insights from a one-dimensional model.," *J. Magn. Reson.*, vol. 170, no. 1, pp. 56–66, Sep. 2004.

[11] V. J. Wedeen, P. Hagmann, W.-Y. I. Tseng, T. G. Reese, and R. M. Weisskoff, "Mapping complex tissue architecture with diffusion spectrum magnetic resonance imaging.," *Magn. Reson. Med.*, vol. 54, no. 6, pp. 1377–86, Dec. 2005.

[12] E. S. Hui, E. Fieremans, J. H. Jensen, A. Tabesh, W. Feng, L. Bonilha, M. V. Spampinato, R. Adams, and J. a. Helpern, "Stroke assessment with diffusional kurtosis imaging," *Stroke*, vol. 43, no. 11, pp. 2968–2973, Nov. 2012.

[13] M. F. Falangola, C. Branch, J. H. Jensen, C. Hu, L. Xuan, K. Duff, R. Nixon, and J. A. Helpern, "Assessment of Brain Microstructure in a Transgenic Mouse Model of β -Amyloid Deposition," in *Proc. Intl. Soc. Mag. Reson. Med. 15*, 2007, vol. 15, p. 2007.

[14] J. A. Helpern, V. Adisetiyo, M. F. Falangola, C. Hu, A. Di Martino, K. Williams, X. Francisco, and J. H. Jensen, "Preliminary Evidence of Altered Gray and White Matter Microstructural Development in the Frontal Lobe of Adolescents With Attention-Deficit Hyperactivity Disorder: A Diffusional Kurtosis Imaging Study," vol. 33, no. 1, pp. 17–23, 2012.

[15] A. Ramani, J. H. Jensen, K. U. Szulc, O. Ali, C. Hu, H. Lu, J. D. Brodie, and J. A. Helpern, "Assessment of abnormalities in the cerebral microstructure of schizophrenia patients: a diffusional kurtosis imaging study," in *Proc. Intl. Soc. Mag. Reson. Med. 15*, 2007, vol. 15.

[16] A. Tabesh, J. H. Jensen, B. a. Ardekani, and J. a. Helpern, "Estimation of tensors and tensor-derived measures in diffusional kurtosis imaging," *Magn. Reson. Med.*, vol. 65, no. 3, pp. 823–836, Mar. 2011.

[17] T. Birnstiel, "Bayesian Estimation of the Diffusion Tensor from diffusion weighted MRI data," Faculty of the University at Albany, State University of New York, 2007.

[18] L. T. DeCarlo, "On the meaning and use of kurtosis.," *Psychol. Methods*, vol. 2, no. 3, pp. 292–307, 1997.

[19] M. Lazar, J. H. Jensen, L. Xuan, and J. A. Helpern, "Estimation of the orientation distribution function from diffusional kurtosis imaging.," *Magn. Reson. Med.*, vol. 60, no. 4, pp. 774–81, Oct. 2008.

[20] P. J. Basser and D. K. Jones, "Diffusion-tensor MRI: theory, experimental design and data analysis - a technical review.," *NMR Biomed.*, vol. 15, no. 7–8, pp. 456–67, 2002.

[21] Y. Liu, L. Chen, and Y. Yu, "Diffusion kurtosis imaging based on adaptive spherical integral," *IEEE Signal Process. Lett.*, vol. 18, no. 4, pp. 243–246, 2011.

[22] D. Le Bihan, J. F. Mangin, C. Poupon, C. a Clark, S. Pappata, N. Molko, and H. Chabriat, "Diffusion tensor imaging: concepts and applications.," *J. Magn. Reson. Imaging*, vol. 13, no. 4, pp. 534–46, Apr. 2001.

[23] J. A. H. Jens H. Jensen, Maria F. Falangolaa, Caixia Hua, Ali Tabesh, Otto Rapalino, Calvin Lo, "Preliminary observations of increased diffusional kurtosis in human brain following recent cerebral infarction," *NMR Biomed.*, vol. 24, no. 5, pp. 452–457, 2011.

[24] M. M. Cheung, E. S. Hui, K. C. Chan, J. a. Helpern, L. Qi, and E. X. Wu, "Does diffusion kurtosis imaging lead to better neural tissue characterization? A rodent brain maturation study," *Neuroimage*, vol. 45, no. 2, pp. 386–392, Apr. 2009.

[25] B. C. Carlson, "Computing elliptic integrals by duplication," *Numer. Math.*, vol. 33, no. 1, pp. 1–16, 1979.

[26] C. G. Koay, J. D. Carew, A. L. Alexander, P. J. Basser, and M. E. Meyerand, "Investigation of anomalous estimates of tensor-derived quantities in diffusion tensor imaging," *Magn. Reson. Med.*, vol. 55, no. 4, pp. 930–936, Apr. 2006.

[27] E. Ziegel, W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, vol. 29, no. 4. Cambridge University Press, 1987, p. 501.

[28] B. C. Carlson, "Numerical computation of real or complex elliptic integrals," Sep. 1994.

[29] J. Jensen, C. Hu, and J. Helpern, "Rapid data acquisition and post-processing for diffusional kurtosis imaging," in *Proc. Intl. Soc. Mag. Reson. Med. 17*, 2009, vol. 17.

[30] E. Fieremans, A. Pires, and J. Jensen, "A Simple Isotropic Phantom for Diffusional Kurtosis Imaging," *Magn. Reson. Med.*, vol. 68, pp. 537–542, 2012.