Quantitative Magnetic Resonance Diffusion Imaging of the Human Brain

Paul Anthony Armitage

PhD
University of Edinburgh
1999

Abstract

Physiological changes within tissues are often accompanied by a movement of water between different cellular environments and an eventual breakdown of cell membranes. Magnetic resonance diffusion imaging is a relatively new medical imaging technique, capable of observing these water diffusion changes in vivo on a quantitative basis. The ability of diffusion imaging methods to provide a quantitative evaluation is extremely useful, for example in observing the effectiveness of various drug therapies at reversing pathophysiological processes. However, there are many potential difficulties to overcome if reliable diffusion measurements are to be obtained in vivo. This work studies some of these problems and attempts to overcome them to provide a reliable quantitative analysis that can be used in the current clinical setting.

In conventional experiments, diffusion was treated as a scalar quantity, based on models assuming spherical or isotropic diffusion. In most living tissues, this is an oversimplification as diffusion is not isotropic and must be treated as a tensor quantity. As such, the likely errors arising from performing diffusion measurements assuming spherical or cylindrical symmetry were investigated using computer simulations. Large and unpredictable errors were found to result if diffusion was not treated as a tensor quantity and models assuming spherical or cylindrical symmetry were used.

Diffusion measurements are also highly susceptible to the effects of experimental noise. Previously suggested measurements as to the extent of noise corruption in diffusion experiments are found to be extremely dependent on the experimental parameters used and so are difficult to apply in a general situation. To overcome this, a measurement of noise was found that gives an analysis that is largely independent of the experimental conditions. Achieving an accurate quantitative analysis has been shown to require a careful balance between obtaining a high enough degree of diffusion weighting, while still maintaining sufficient signal-to-noise ratio. A theoretical method was developed for producing reliable diffusion measurements by optimising the diffusion weighting *b*-value and the number of acquisitions obtained. Both in vitro and in vivo data were found to give reliable quantitative data from an acquisition scheme based on the theoretical method.

Many different ways for displaying diffusion data have been proposed. An analysis of the different levels of contrast and sensitivity arising from various diffusion anisotropy indices was also undertaken, resulting in the development of a method for displaying diffusion data with improved contrast. During the course of this work quantitative diffusion imaging has been performed in a clinical setting on over 100 acute stroke patients and 16 head injured patients.

Contents

1

Introduction

1.1 Background

Magnetic resonance (MR) diffusion imaging is a non-invasive technique for measuring the mobility of water. Potentially, many clinical disorders can be highlighted using MR diffusion imaging, where the biophysical environment is altered in the pathological state. The most notable of these examples is in the process that accompanies ischaemic brain injury, particularly in stroke. It is in this application that this project focuses, with a view to determining a methodology for producing a reliable quantitative assessment of water diffusion changes resulting from ischaemic damage. The work was funded by the Medical Research Council of Great Britain and was carried out in the Department of Clinical Neurosciences of the University of Edinburgh, Western General Hospital, Edinburgh.

Prior to the author joining the group in September 1996, some preliminary work had been performed by Dr Ian Marshall (Senior Lecturer, Dept. of Medical Physics, Edinburgh). A sequence capable of diffusion-weighted imaging (DWI) had been implemented on the clinical magnetic resonance imaging (MRI) scanner (Siemens, Erlangen, Germany), and a small pilot study of patients was scanned under the direction of Dr Joanna Wardlaw (Consultant Research Neuroradiologist, Dept. of Clinical Neurosciences, Edinburgh). However, at this stage the infrastructure for performing a quantitative analysis of this data had not been implemented. As such, the early goals of this project were to develop a simple quantitative analysis, and to

evaluate the feasibility of performing this analysis on stroke patients. This was a challenging prospect due to the sensitivity of the imaging technique to motion and the relatively uncooperative nature of many stroke patients. Thus, a main focus for the project was to investigate the effects of motion and its associated artefacts and investigate methods to overcome these.

A dedicated research scanner (Elscint, Haifa, Israel) was installed in January 1998, with the advent of the SHEFC Brain Imaging Research Centre for Scotland, resulting in a shift of the project focus. The new scanner had much improved hardware performance allowing for faster imaging, which largely solved the problem of motion artefacts. Therefore, the project became more concerned with achieving a fully quantitative analysis of diffusion with the implementation and optimisation of a diffusion tensor imaging (DTI) acquisition and post-processing scheme that would be suitable for imaging of stroke patients.

1.2 Scope of the text

During the rest of this introduction, a brief history of the development of MR and diffusion imaging techniques will be presented. The reasons for developing clinical MR diffusion imaging, compared to other imaging modalities will be discussed, providing an insight into its benefit and potential applications.

In Chapter 2 the basic principles of magnetic resonance imaging are presented. This is discussed in terms of the theory of the nuclear magnetic resonance (NMR) phenomena and how this can be exploited to produce images of structural and physiological properties. This theory is important for understanding many of the imaging strategies and their potential problems, that are discussed in subsequent chapters. Chapter 3 then goes on to present the physical methods used to obtain diffusion images from NMR principles along with the practical implications and applications of diffusion imaging. Image artefacts that are largely specific to diffusion sequences are also discussed. A critical feature of diffusion imaging is its ability to obtain quantitative physiological data in vivo, and the methodology for producing these quantitative measurements is presented. Finally, the potential chemical and biological applications for diffusion imaging are discussed.

Chapter 4 presents the methods used during the course of this study for acquiring and displaying diffusion data. The technical aspects of the two MRI scanners used during the project are described, namely the Siemens 63SP Magnetom 1.5T and the Elscint Prestige 1.9T. Due to the limited access available for development work on the scanners (a maximum of 2-3 hours a week during 'working' hours), a large proportion of the project has been concerned with the development and evaluation of post-processing techniques. Much of the software needed to do this has been developed 'in house' and is described in detail in Chapter 4.

Results from a preliminary study into the feasibility of performing diffusion imaging in our clinical environment are presented in Chapter 5. Initial quantitative measurements, using the orthogonal imaging technique, are given from a series of 12 stroke patients. The problems associated with obtaining accurate data are highlighted and an assessment made of the likely reliability of the presented results. The study also observes the effects of noise and image alignment on diffusion measurements.

The following two chapters are concerned with developing and investigating methods for improving the reliability of typical diffusion data as presented in the preliminary study. Chapter 6 takes a theoretical look at the different methods for sampling diffusion, namely the orthogonal, tetrahedral and tensor techniques. This chapter assesses the likely errors that will arise if the whole diffusion tensor is not sampled, i.e. if models assuming spherical or cylindrical symmetry are used. Chapter 7 addresses the problem of obtaining noise independent diffusion anisotropy measurements in the human brain, by optimising acquisition parameters, such as the degree of diffusion weighting and the number of acquisitions. A novel approach is taken that minimises the overall imaging time, while still ensuring accurate diffusion anisotropy measurements. This optimisation scheme is developed in theory and demonstrated by simulation, and with in vitro and in vivo data.

Whilst Chapters 6 and 7 deal with the acquisition aspects of diffusion imaging and with methods for improving the accuracy of the resulting measurements, Chapter 8 compares the techniques developed for displaying diffusion data. A mathematical representation for describing the sensitivity of diffusion anisotropy indices (DAIs) was developed, and used to predict their contrast and sensitivity to

noise. This lead to the development of a new DAI which gave improved contrast for human brain imaging.

Lastly, Chapter 9 presents in vivo results from the various clinical studies that have been performed during the course of this project. These studies have been performed with a view to obtaining information on the biophysical processes that take place, particularly in ischaemic brain injury. Most of the results presented are for patients suffering from stroke, with over 100 being scanned using diffusion imaging, although other examples are given from a smaller study of acute closed head injury.

1.3 History of nuclear magnetic resonance imaging

1.3.1 Nuclear magnetic resonance

The NMR effect was first observed in 1945, independently, by two different groups, that of Bloch et al. (1) and Purcell et al. (2). In 1952 they were to share the Nobel Prize for physics in recognition of this pioneering work. They observed that when water molecules were irradiated with radio frequency (RF) electromagnetic radiation in the presence of a strong static magnetic field, certain 'resonant' frequencies were selectively absorbed. In 1949, Proctor and Yu (3), and Dickinson (4), independently observed that nuclei sited in different chemical environments resonated with slightly different frequencies. This was the first observation of the chemical shift that was to have wide ranging implications for the field of analytical spectroscopy. In 1950, Hahn (5) discovered the spin echo which forms the basis of many imaging and spectroscopy NMR sequences.

These discoveries, along with the development of Fourier analysis techniques, lead to the first commercial NMR spectrometer being introduced in 1953. Since then NMR techniques have been superseding infra-red spectroscopy as the method of choice for characterising the composition and structure of molecules. The characteristic frequencies present in an NMR spectrum provide a fingerprint with which to identify the molecule. This form of high resolution NMR spectroscopy requires extremely small magnetic field inhomogeneities (typically 1 part in 10^9) across the sample, so that the small chemical shift effects which enable molecular

identification are not masked by magnetic inhomogeneity induced line broadening. However, it would be impractical to perform this kind of high resolution NMR in a biological setting as achieving a uniform magnetic field across a large sample, such as the human brain, is extremely difficult to achieve. The high sensitivity and abundance of protons in biological systems, has resulted in most biological applications focusing on the imaging of protons. Various guises of protons are found in biological systems, those contributing to the NMR signal are predominately associated with water, but there is also a significant contribution from protons of mobile lipids. Therefore, NMR techniques have largely been refined to provide measurements of the total proton spin density at a given position.

The high abundance and sensitivity of protons in water hampered in vivo proton spectroscopy in biological systems for many years. The strong water signal masked out the small resonance's from the metabolites of interest and as such ^{31}P spectroscopy became the principal choice for in vivo work. By the early 1980s, water suppression techniques had been developed that enabled in vivo proton spectroscopy to become a realistic proposition and to date many studies have been performed to observe metabolic concentrations in vivo.

1.3.2 Nuclear magnetic resonance imaging

The major development in NMR imaging came with the realisation that, with the resonant frequency being proportional to the magnetic field strength, the application of a magnetic field gradient would give rise to spins resonating with different frequencies depending on their spatial position in the field. This lead to the first NMR images being produced in 1973 by the groups of Lauterbur (6) in New York and Mansfield (7) in Nottingham. The spatial information can be mapped using several different techniques which use the intensity and phase information to encode the position of the spin as a frequency or cumulative phase. There are many different ways to encode the data, however the most established techniques are those using Fourier imaging or some form of projection reconstruction.

High resolution NMR spectrometers generally work with sample sizes of around 5 - 10 mm and so it is relatively easy to achieve acceptably small magnetic

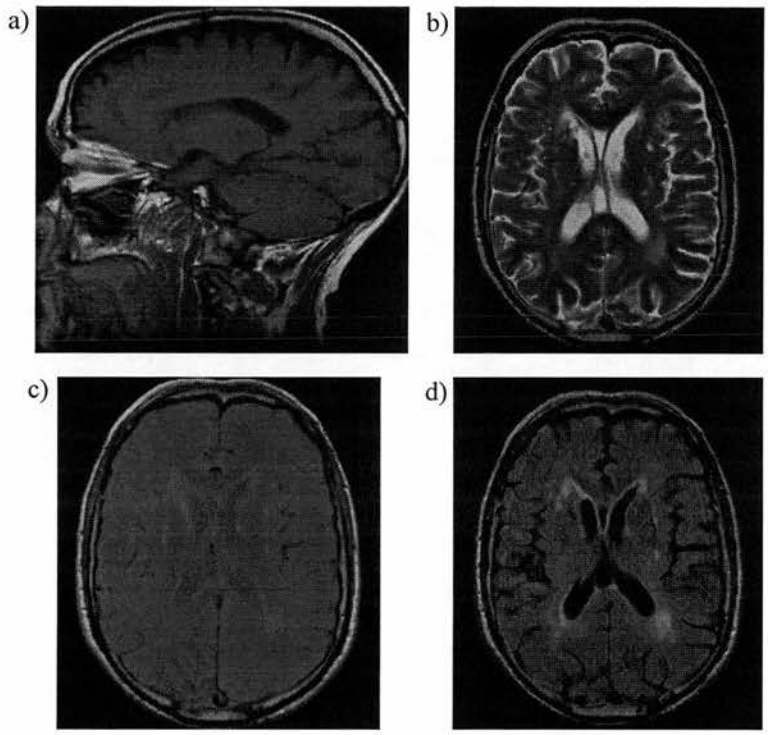


Figure 1.1 Examples of the different types of image contrast possible using MRI techniques showing; a) T_1-weighted sagittal image; b) T_2-weighted axial image; c) Proton density axial image; and d) inversion recovery CSF-suppressed FLAIR image.

field inhomogeneities. In order to achieve high resolution images of human subjects small field inhomogeneities must be obtained over a much larger volume. By 1976 acceptable images were obtained in human fingers, and magnets capable of imaging cross-sections of 1 - 10 cm appeared soon after. Whole body imaging was first achieved in 1977 by Damadian (8) using a FONAR technique and shortly afterwards by Mansfield et al. (9) who used a line scanning method. Since then many whole body systems have been developed, both commercially and in academia with the first commercially available clinical systems being introduced in the USA in 1983. Static magnetic field strengths have been steadily increasing through the use of superconducting magnets, with whole body systems having typical strengths of 1.5T. This compares to 0.04 - 0.2T for the resistive magnets used in the early days of whole body imaging. The magnetic field homogeneity has also improved greatly across the wider bore magnets.

The technique on which many imaging sequences base their acquisition is the spin-warp phase encoding method developed by Edelstein et al. in 1980 (10). This technique uses frequency and phase encoding and a 2D fast Fourier transform (FFT) reconstruction (11). Some typical MR images are shown in Fig. 1.1, demonstrating the different contrast that can be achieved by exploiting the fundamental NMR relaxation parameters T_1 and T_2. One of the most significant contributions to modern day NMR is that of echo planar imaging (EPI), first conceived in 1977 by Mansfield (12), however, the procedure was slow to be adopted due the high demands this technique placed on imaging hardware. Echo planar imaging is a single-shot technique, acquiring the whole image in a single excitation, allowing for very rapid imaging (1000 times faster than typical spin-warp techniques) and hence the freezing out of motion artefacts. By 1988 the group in Nottingham had begun to produce EPI images of comparable quality to 'conventional' images (13) and proved that EPI was a viable technique for in vivo imaging. The development of the technology behind EPI forms the basis for many current imaging sequences such as fast spin echo (FSE) and functional techniques using BOLD (blood oxygen level dependent) imaging.

1.3.3 Magnetic resonance diffusion imaging

Magnetic resonance techniques are inherently sensitive to diffusion as was realised by Carr and Purcell in 1954, who made the first diffusion measurements using NMR (14). The Bloch equations (1), which describe the NMR free precession were modified by Torrey in 1956 to include terms accounting for the effects of diffusion (15). In 1965 Stejskal and Tanner developed a method for measuring diffusion coefficients which forms the basis for most of the diffusion acquisition strategies used today (16). This was based on the spin-echo technique of Hahn (5) and utilised a pulsed field gradient. This methodology formed the basis for measurements of self diffusion coefficients in chemistry, replacing previously used radioactive tracer techniques.

During the early 1970s diffusion measurements were being obtained from tissue samples using NMR spectroscopy, but it was not until 1985 that spatial diffusion images were first obtained (17,18). A year later the first in vivo diffusion images were obtained by Le Bihan et al. (19) along with the concept of intravoxel

incoherent motion (IVIM). Shortly afterwards, diffusion images were acquired using EPI techniques, allowing for a considerable reduction of motion related artefacts (20). Since then diffusion imaging has found a variety of clinical applications, particularly in neurological disorders such as stroke (21), multiple sclerosis (22) and tumours (23). Basser et al. introduced the concept of diffusion tensor imaging in 1994 (24,25), proposing a methodology for sampling the entire diffusion tensor, and thus being able to characterise completely the molecular diffusion environment. This lead to many further possibilities for using diffusion imaging to perform fibre tract mapping and so to gain a further insight into white matter structural disorders (26).

1.4 Why should MR diffusion imaging be performed?

Conventional brain imaging techniques have concentrated on obtaining anatomical information to map the underlying structure present in an area of interest. Such information is obtained, routinely, using x-ray computed tomography (CT) (28) and to a lesser extent T_1 or T_2 weighted MRI (29). Excellent visualisation of brain anatomy is provided by these modalities. However, their ability to provide early visualisation of events such as ischaemia may be limited due to their reliance on structural changes, which may occur significantly after the accompanying pathophysiological processes. The ability to observe physiological processes such as water diffusion, blood perfusion and blood flow would potentially enable such events to be observed before permanent structural changes have taken place.

A comparative study of early visible infarction using MRI and CT demonstrated that 82% of the study group had an MRI visible infarct and only 58% were CT visible after 24 hours (30). The suggestion that MRI may be more sensitive to the early stages of acute brain ischaemia than CT may be indicative of the different processes involved in structural CT and MRI scans. CT relies on x-ray scattering and absorption processes to provide structural information, whereas MRI relies on the processes of T_1 spin-lattice and T_2 spin-spin relaxation. The hyperintensity observed on T_2 weighted MRI due to ischaemia is believed to be caused by the swelling associated with cerebral oedema. During the early stages of ischaemia, oedema may not be very extensive and so changes due to T_2 may not be seen. Thus, physiological

imaging techniques such as diffusion-weighted MRI, perfusion MRI and MR angiography provide the potential to observe the processes preceding oedema and infarction, allowing an early diagnosis, which could be essential in the administration of early neuroprotective reperfusion therapies.

1.5 Alternative and complementary imaging strategies

Magnetic resonance is not the only modality capable of performing physiological imaging. Other techniques such as single photon emission computed tomography (SPECT) (31), positron emission tomography (PET) (32), and transcranial Doppler ultrasound (TCD) (33) can all provide an analysis of physiological parameters. MR techniques may not be the method of choice for any given physiological measurement, but with the development of faster imaging sequences, all of the complementary structural and physiological information needed could be obtained in one sitting, lessening the need for time consuming transfer between imaging modalities. The relative merits of the different techniques for performing physiological imaging are discussed in this section.

1.5.1 Blood flow

Measurement of blood flow is critical for identifying aneurysms and vessel occlusions which can lead to reduced blood supply to the brain, potentially resulting in the onset of ischaemia. Blood flow in vessels can be measured using TCD (33) which can be used to identify turbulent flow associated with partial vessel occlusion. TCD equipment is relatively portable and has a significant cost advantage over MR, currently being cheaper by a factor of about ten.

A variety of MR angiography techniques are available to observe blood flow abnormalities. The most common method involves the injection of a bolus of paramagnetic contrast agent into the blood stream. If images are taken quickly then the contrast agent will still be present in the major blood vessels of the brain, highlighting their structure. This technique enables larger occlusions to be observed, but measurements of blood flow are not easily obtained. The second method uses either arterial spin tagging or time of flight measurements (TOF), and the resulting

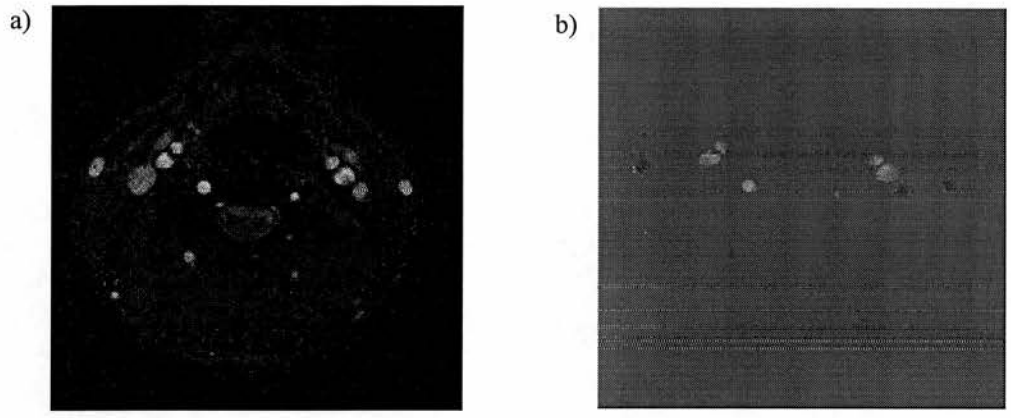


Figure 1.2 Illustration of a) magnitude and b) reconstructed velocity data obtained using PCA. The velocity is encoded such that dark pixels represent velocities into the image plane and light pixels represent velocities out of the plane. (Images appear courtesy of Uwe Koehler).

images show blood as either bright or dark regions depending on the exact labelling mechanism used (34). There are a few different varieties of this type of sequence, such as those which magnetically label spins in one plane and image them in another, resulting in the spins that were magnetically labelled and have 'flowed' into the imaging plane being observed. This technique enables an estimation of the blood flow, but requires a knowledge of the vessel geometry. The third technique is phase-contrast angiography (PCA), which allows a full quantification of velocities in vessels (35,36). This technique utilises an imaging sequence that is highly sensitised to blood flow along the direction of an applied gradient. The mechanism is very similar to the diffusion techniques described later, but flow results in a coherent phase shift of the transverse magnetisation, which can be measured and is proportional to the flow velocity. The ability to measure flow along a gradient direction enables measurements to be obtained in three orthogonal directions, enabling a complete description of the velocity vector. Figure 1.2 illustrates a magnitude PCA image and a reconstructed velocity image obtained from a normal volunteer.

1.5.2 Perfusion

Perfusion describes the density of microvasculature in a tissue (37) and is conventionally measured in cc $(100g)^{-1}$ min^{-1}. If blood flow to the brain is reduced by

some means, such as an occluded artery, then it is likely that a perfusion abnormality will occur in some region of the brain. Highlighting these regions could be critical in identifying areas that are likely to go on to possible infarction if the blood supply is not resumed. Perfusion measurements can be made using SPECT and PET, which provide information on the regional cerebral blood volume (rCBV) and regional cerebral blood flow (rCBF). SPECT uses a gamma camera to detect gamma emitting radio-pharmaceuticals injected into the blood stream. The gamma camera is then rotated around the patient to obtain a series of planar images which can be reconstructed to obtain a tomographic image using back projection. The resolution of the gamma camera is relatively poor, typically giving a pixel size of 12mm at full width half maximum. Radio-nuclides such as Technitium-99m and Iodine-123 are usually used and after administration are distributed in proportion to the blood flow, giving a good indication of the rCBV. The limitations on administering radio-pharmaceuticals means that relatively small quantities are administered making measurements of rCBV from small anatomical regions difficult, requiring specific targeting of the region to achieve satisfactory sensitivity. Therefore, the ability to obtain perfusion measurements using non-ionising radiation would clearly be desirable.

MR perfusion measurements have been developed using several techniques. The first involves the use of an exogenous tracer, whereby rapid MRI techniques resolve the tissue transit of intravenously administered contrast agents such as gadolinium diethylenetriamine pentaacetic acid, Gd-DTPA (38,39). These contrast agents can exploit either T_1 or T_2 contrast, but the resulting signal is, in general, not linear with the concentration of contrast present. Considerable research has therefore focused on developing a better understanding of the dose-relaxation rates of these contrast mechanisms (40). The second approach to measurement of perfusion, magnetically labels blood to act as an endogenous tracer, resulting in a change of T_1 signal dependent on the rCBF (41,42). Incoming blood T_1 weighted signal undergoes inversion or saturation, as in time-of-flight steady flow measurements. The spin inversion pulses are applied and omitted alternately and changes in T_1 signal between the two sets of images are then observed and related to the rCBF. The technique is demanding on both hardware and processing because the signal changes are very

small, typically less than 5%. The third method uses the principle of IVIM as in diffusion imaging, where the microvasculature present in a tissue forms a random network over the scale of a macroscopic voxel and can be considered to be a pseudo-diffusion process (19). A further method for providing the necessary contrast to show perfusion abnormalities is to utilise blood oxygen level dependent, BOLD, contrast. This technique relies on the magnetic susceptibility associated with different degrees of oxygen saturation of deoxy-haemoglobin in red blood cells (43). A T_2^* imaging sequence is highly sensitive to these changes in susceptibility and shows a signal reduction when the blood oxygenation level is reduced in hypo-perfused tissue. These signal reductions have been observed in ischaemia and hypoxia (44), but are more commonly used as the basis of functional MRI (fMRI) studies.

1.5.3 Metabolic imaging

Metabolic imaging can be performed with PET, SPECT and MR spectroscopy studies. PET uses positron emitting isotopes that are created in a particle accelerator. The resulting radio-isotopes usually have half-lives that are considerably shorter than SPECT radio-nuclides. Therefore, activities can be increased by using PET, improving sensitivity and allowing the acquisition of repeated measurements over a reasonable time scale. However, the short half lives of PET radio-nuclides means that an on-site particle accelerator is required along with chemical synthesis facilities. The requirement for such facilities along with the relatively high cost of PET scanners has limited the application of this technique. Nevertheless, cerebral glucose metabolism has been studied using the nuclide ^{14}C-deoxyglucose (45) and the isotope ^{15}O has been used to observe cerebral oxygen uptake enabling the study of cerebral metabolism and tissue oxygenation (46). Recent work has involved the development of PET and SPECT tracers for CNS receptors such as serotonin and dopamine, which are believed to be involved in the processes resulting in nervous disorders such as Alzheimer's and Huntingdon's diseases.

With the improvement of magnetic field homogeneities and the development of water suppression techniques (47,48), ^{1}H proton MR spectroscopy has been used in vivo to study the concentrations of different metabolites in an imaging volume, typically of size 1 cm^3 or more. An NMR spectrum is obtained, where resonant peaks occur corresponding to the characteristic frequencies of the metabolites present. The technique suffers from relatively poor spatial resolution because the low concentrations of metabolites (mM) require a reasonably large imaging volume in order to achieve sufficient signal-to-noise ratio (SNR). The area under the peaks can be integrated to obtain estimates of the relative concentration of the different metabolites present in the imaging volume. Recently, spectroscopic imaging techniques have been developed, whereby 2D phase encoding is used to produce an array of spectra from a planar grid of voxels (49,50). These techniques typically acquire spectra from a coarse grid of 16×16 voxels over a single imaging slice, allowing the mapping of metabolite distribution over a cross-section of the brain.

The main metabolites of interest using conventional long echo time ^{1}H spectroscopy of the brain are N-acetylaspartate (NAA), choline, creatine and lactate, although other resonances, such as glutamate and glutamine can be observed using short echo time techniques. NAA is an amino acid associated with healthy neurones and is believed to be reduced in many diseased states, such as stroke, Alzheimer's disease, epilepsy and multiple sclerosis. Choline is related to phospholipid metabolism, an increase of which may be associated with the breakdown of cell membranes in decaying tissue. Creatine is found in neurones and glial cells and is involved in the process of phosphate transfer. The creatine concentration is believed to remain roughly constant in the human brain, even in diseased states and is often used as an internal reference for the quantification of other metabolite concentrations. Lactate is a by-product of anaerobic glycolysis and is present in very small concentrations in normal brain. However, its levels can become significantly elevated in states of hypo-perfusion, where there is an imbalance between glucose uptake and blood flow clearance. Thus, increased lactate concentration may be an early indicator of a perfusion deficit and potential onset of ischaemia. However, the lactate resonance is often obscured by the broad lipid resonance and is difficult to measure because its lineshape varies with echo time, due to spin-spin coupling

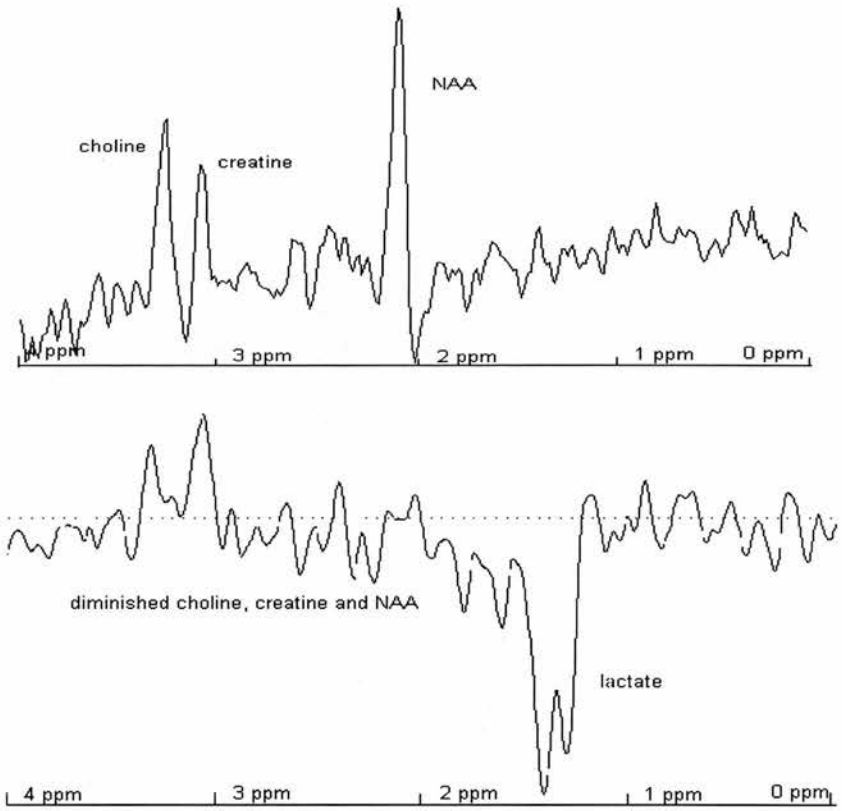


Figure 1.3 NMR spectra acquired from a) a healthy subject and b) a stroke patient, illustrating the reduction of the NAA, choline and creatine peaks and the presence of a lactate peak in the stroke patient. (Spectra appear courtesy of Dr Jim Wild).

between different proton subgroups resulting in J-modulation. Therefore, lactate editing techniques are often required to observe the resonance (51). Figure 1.3 shows NMR spectra obtained from a healthy subject and a stroke patient, clearly illustrating the reduced NAA and lactate peaks observed in the stroke patient.

1.5.4 Summary of physiological imaging techniques

The physiological and metabolic imaging strategies outlined above, along with diffusion imaging, provide complementary information for accurate diagnosis of pathological conditions and for identifying regions of potentially salvageable tissue. In the case of ischaemia, using blood flow measurements to identify aneurysms and vessel occlusions as the root cause of reduced blood supply to the brain allows methods to reinstate normal blood flow to be instigated. Imaging possible perfusion

deficits enables regions of 'at risk' tissue to be identified and reperfusion steps can be taken. Diffusion imaging demonstrates early changes of tissue microstructure which may be indicative of subsequent infarction, however, evidence suggests that in the early stages diffusion abnormalities may be reversed if reperfusion of the region occurs quickly. If reperfusion has been unsuccessful and the tissue progresses towards infarction then the abnormalities can be identified using conventional structural imaging techniques. Metabolic imaging provides complementary information with evidence of increased lactate supporting the presence of a perfusion deficit, while reduced NAA and elevated choline may be indicative of cell destruction.

MR techniques have the potential to provide information on structural, flow, perfusion, diffusion and metabolic changes in one examination, particularly with the development of improved imaging sequences and post-processing techniques, resulting in faster and improved imaging and diagnosis. The development of physiological imaging strategies is critical for early identification of ischaemia, aiding the potential administration of reperfusing or neuro-protective drugs. While several different modalities are available for imaging many of the physiological and metabolic parameters, to date only MR techniques have been utilised in a clinical environment for the measurement of diffusion processes.

2

Fundamental Principles of Magnetic Resonance Imaging

2.1 Introduction

This chapter describes the fundamental theory behind NMR and MR imaging. The classical and quantum mechanical descriptions of the NMR effect are presented in §2.2 and it is demonstrated how the classical model can often be used to give a clearer visual representation of the quantum mechanical NMR effect. The NMR relaxation mechanisms, T_1 and T_2, are presented in §2.3 and their importance in the production of image contrast is discussed. Methods for measuring the time varying NMR signal are discussed and Fourier methods for obtaining the frequency components present in the time domain signal (the NMR spectrum) are shown. The use of magnetic field gradients to provide selective excitation and spatial localisation of the NMR signal in a sample and how this is combined with Fourier methods to obtain an NMR image is given in §2.4. Finally, in §2.5 practical examples are presented to demonstrate how magnetic field gradients can be combined to produce a pulse sequence capable of acquiring an NMR image. This is explained by using the concept of reciprocal k-space. The examples given are based on the two pulse sequences used throughout this study; the 2DFT spin-echo and the EPI spin-echo sequence.

2.2 The NMR effect

Atomic nuclei consist of protons and neutrons, each of which have an intrinsic spin quantum number of ½. The total spin of the nucleus is obtained by taking the vector sum of the component nucleon spins, resulting in the existence of many nuclei that have non-zero net spin. The hydrogen nucleus, consisting of a single proton and hence non-zero spin, is of primary interest in medical NMR due to its abundance in the human body. Associated with this spin is a small magnetic moment and the application of a large magnetic field, $\boldsymbol{B}_0$, results in the nuclei interacting with the field and precessing around it. Some of the spins precess with their axes in the same direction as the magnetic field and others align against the field.

If radiofrequency (RF) radiation is applied at a frequency determined by $\boldsymbol{B}_0$, then the protons can flip from being aligned with the magnetic field to being against it and vice-versa. In the equilibrium situation with no RF radiation applied, slightly more spins will align themselves along the field direction than against it, because the former is the lower energy state. This results in a small net magnetisation along the direction of $\boldsymbol{B}_0$. When RF radiation is applied to the nuclei, more nuclei absorb energy and transfer into the higher energy state than lose energy and fall into the lower energy state, resulting in a net absorption of RF radiation. This absorption of energy can be measured and the signal intensity observed depends on the number of nuclei present and its form depends on the chemical environment in which the nuclei are situated.

Following the application of RF radiation, the nuclei relax back to their equilibrium state by various mechanisms. These relaxation rates are typically of the order of tens to hundreds of milliseconds and can be measured by observing the signal changes induced during the relaxation. The relaxation is governed by the physical and chemical environment in which the nuclei are situated and can be used to give information on the dynamics and structure of the sample. Two basic methods have been developed for the measurement of NMR signals, utilising either continuous wave or pulse techniques. The former immerses the sample in a high magnetic field that can be swept about a central value and the sample is irradiated

with RF radiation such that the NMR spectrum is obtained continuously. The second technique involves the application of short bursts of RF radiation and then the transient behaviour of the NMR signal is observed while the sample is immersed in a high magnetic field. The latter technique is much more versatile and enables a faster acquisition of the NMR spectrum, resulting in most modern day strategies being based on pulsed NMR techniques.

2.3 Nuclear magnetism and spin excitation

2.3.1 Classical description

The proton has a positive charge and an intrinsic spin angular momentum that combine to give it a magnetic dipole moment. A proton with a magnetic moment μ will experience a torque when a magnetic field $\boldsymbol{B}_0$ is applied at an angle θ, as illustrated in Fig. 2.1. The proton precesses about the applied field in a direction perpendicular to the plane of μ and $\boldsymbol{B}_0$ such that the rate of change of angular momentum $\boldsymbol{J}$ is given by

$$\mu \wedge \boldsymbol{B}_0 = \frac{d\boldsymbol{J}}{dt} \qquad [2.1]$$

where $\wedge$ indicates the vector cross product. The precessional frequency ω_0 can be derived when the precession illustrated in Fig. 2.1 is observed with small changes in

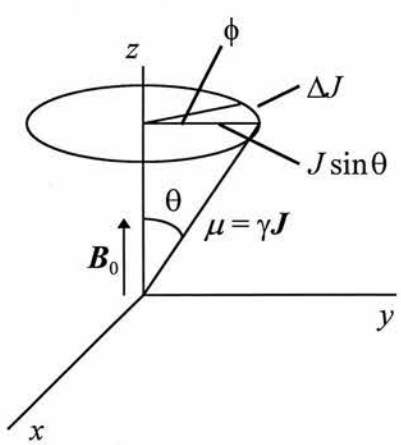


Figure 2.1 Classical representation of the precession of a magnetic moment μ under the influence of a strong magnetic field gradient $\boldsymbol{B}_0$ applied along z.

the angular momentum ΔJ. In time Δt, $\boldsymbol{J}$ precesses through an angle ϕ, such that

$$\phi = \omega_0 \Delta t \qquad [2.2]$$

and ΔJ is then given by

$$\Delta J = J \sin\theta . \phi = J \sin\theta \omega_0 \Delta t \qquad [2.3]$$

Rearranging Eq. 2.3 and taking the limit of small ΔJ and Δt gives the differential form

$$\frac{dJ}{dt} = J \sin\theta \omega_0 \qquad [2.4]$$

Substituting from Eq. 2.1 and expanding the cross product term gives

$$\mu \wedge \boldsymbol{B}_0 = \mu B_0 \sin\theta = J \sin\theta \omega_0 \qquad [2.5]$$

and so

$$\omega_0 = -\frac{\mu \boldsymbol{B}_0}{J} = -\gamma \boldsymbol{B}_0 \qquad [2.6]$$

Equation 2.6 is the Larmor equation which shows how the angular velocity ω_0 is proportional to the magnetic field strength $\boldsymbol{B}_0$, related by the gyromagnetic ratio γ, which is the ratio of the magnetic moment to the angular momentum.

2.3.2 Quantum mechanical description

Applying the theory of quantum mechanics to the spin system, whereby a proton with spin $\boldsymbol{I}$ and magnetic moment μ is immersed in a magnetic field $\boldsymbol{B}_0$, only certain allowed quantum states or axes of rotation are available to the total nuclear angular momentum $\boldsymbol{J}$. The length of the angular momentum vector $\boldsymbol{J}$ is $\hbar\left[I(I+1)\right]^{1/2}$, however, the solutions of the eigenvalue equation representing the nuclear angular momentum wavefunction (i.e. the measurable components of $\boldsymbol{J}$) are given by $m_I \hbar$. The magnetic quantum number m_I can take $(2I + 1)$ values such that $m_I = I, (I - 1), \ldots, -(I - 1), -I$. For a single unpaired nucleon, such as in ^{1}H, ^{13}C, ^{19}F, ^{31}P etc., which has spin $I = ½$, only two values are possible for m_I, + ½ and - ½, corresponding to the parallel and anti-parallel orientations (Fig. 2.2). Nuclei having two unpaired nucleons, such as ^{2}H have spin $I = 1$ and so $m_I = -1, 0, +1$ and then $\boldsymbol{J}$ can orientate itself in three different directions relative to the applied magnetic field

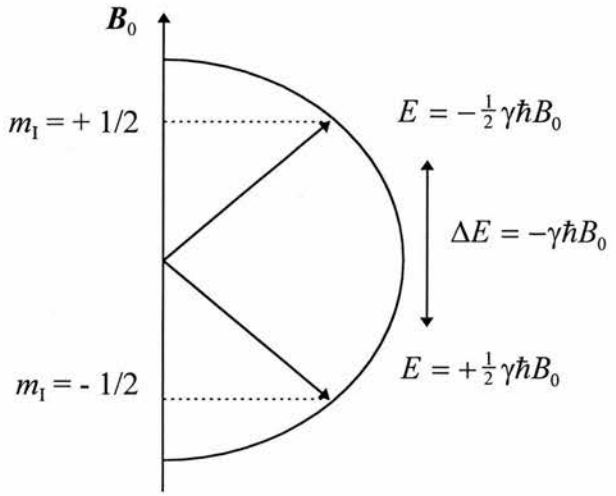


Figure 2.2 Quantisation of energy levels illustrated for the quantum mechanical description of a spin ½ nucleus oriented in a magnetic field $\boldsymbol{B}_0$.

$\boldsymbol{B}_0$. Proton NMR is used throughout this study, where m_I can only take two values and so is the case that is discussed in subsequent sections.

The energy of the quantum states is given by

$$E = -\mu.\boldsymbol{B} = -\gamma\hbar m_I B_0 \qquad [2.7]$$

where $m_I = \pm$ ½ for protons. The relative population of the two states is given by the Boltzmann distribution, such that

$$\frac{n_\uparrow}{n_\downarrow} = \exp\left(\frac{\Delta E}{kT_s}\right) \qquad [2.8]$$

where $n_\uparrow$ and $n_\downarrow$ represent the number of spins in the parallel and antiparallel states, respectively, k is Boltzmann constant, T_s is the temperature of the spin system and ΔE is the energy difference between the two states. At room temperature and typical magnetic field strengths ($\approx$ 1T), Eq. 2.8 demonstrates how slightly more protons exist in the lower energy state ($n_\uparrow$). Transitions between the two states can occur if m_I changes by ± 1, obeying the quantum mechanical selection rule, as illustrated in Fig. 2.2. Thus, the energy difference between the two states is given by $\Delta E = -\gamma\hbar B_0$, and for a transition to occur from the lower energy state to the higher state a quantum of energy must be absorbed, given by

$$\hbar\omega_0 = -\gamma\hbar\boldsymbol{B}_0 \quad \text{or} \quad \omega_0 = -\gamma\boldsymbol{B}_0 \qquad [2.9]$$

This is identical to Eq. 2.6 obtained using classical principles and demonstrates how the classical and quantum mechanical models can be used interchangeably to give a

clear physical understanding of the NMR process. However, the quantum mechanical approach provides the additional idea that the angular momentum $\boldsymbol{J}$ becomes quantised into units of $I\hbar$.

2.3.3 RF excitation of a spin ensemble

The behaviour of an individual proton immersed in a magnetic field has been discussed in §2.3.1 and §2.3.2. In reality, there are many such protons in a sample and their combined behaviour must be considered. As was shown earlier, the protons distribute themselves between the energy levels as determined by the Boltzmann distribution in Eq. 2.8. At room temperature, in a magnetic field of around 1T, $n_{\uparrow} / n_{\downarrow} \approx 1 + 4 \times 10^{-6}$, resulting in a small net magnetisation $\boldsymbol{M}_0$ in the direction of the applied static magnetic field $\boldsymbol{B}_0$. The strength of $\boldsymbol{M}_0$ will increase as $\boldsymbol{B}_0$ increases, as can be seen from Eq. 2.8 (as ΔE increases with $\boldsymbol{B}_0$ leading to an increase in $n_{\uparrow} / n_{\downarrow}$), resulting in a general improvement in SNR for higher magnetic field strengths. The net magnetisation $\boldsymbol{M}_0$ is considered to rotate at the Larmor frequency ω_0, as is illustrated in Fig. 2.3.

In a traditional continuous wave experiment, illustrated in Fig. 2.4, the sample is immersed in a static magnetic field $\boldsymbol{B}_0$ and irradiated with RF radiation at

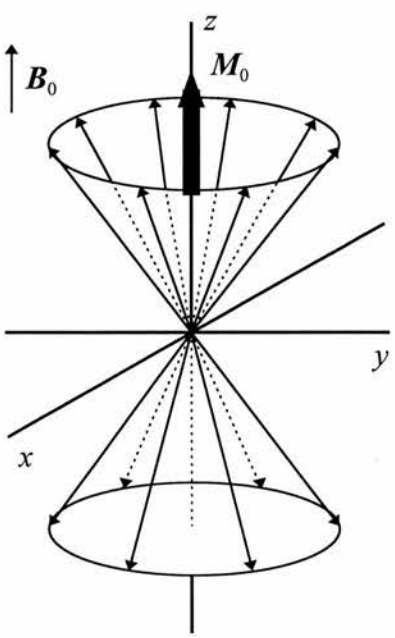


Figure 2.3 Illustration of the net magnetisation $\boldsymbol{M}_0$ resulting from an ensemble of protons precessing in a magnetic field $\boldsymbol{B}_0$. Notice how slightly more protons align themselves parallel to $\boldsymbol{B}_0$.

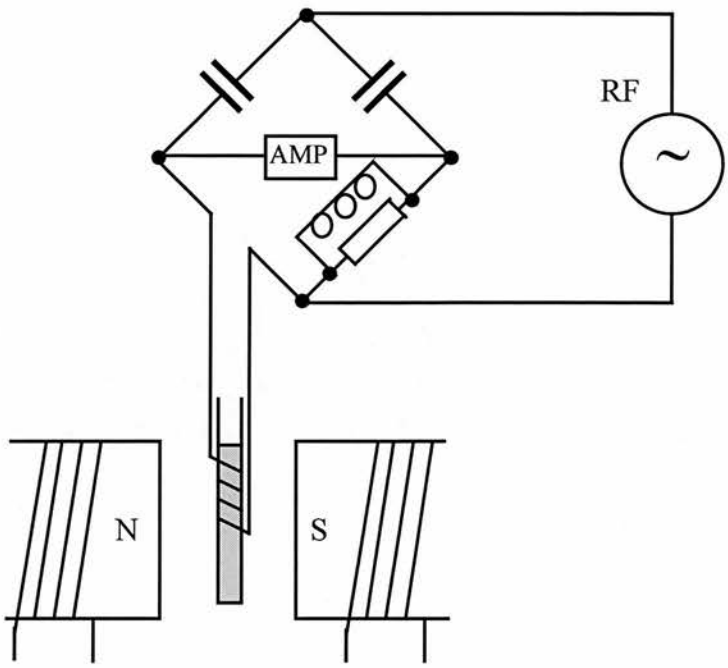


Figure 2.4 Schematic illustration of a continuous wave NMR experiment

frequency ω_0 perpendicular to the direction of $\boldsymbol{B}_0$. The $\boldsymbol{B}_0$ field can then be swept from below to above the resonant condition of $B_0 = \omega_0 / \gamma$, and at resonance the sample will absorb RF radiation, which can be detected from the imbalance it causes in the bridge circuit. The sample absorbs radiation because there are more protons in the lower energy state ($m_I = + ½$) and the transition rates are equal for both protons being stimulated to absorb energy from $m_I = + ½$ to $m_I = - ½$ and for those emitting energy from $m_I = - ½$ to $m_I = + ½$, thereby resulting in a net absorption. A Lorentzian shaped absorption curve of finite width is obtained from a water sample. This line broadening is due to magnetic field inhomogeneities which result in spins at different positions resonating at slightly different frequencies and also due to T_2 spin-spin interactions (see §2.4.3) which cause a dephasing of the spins rotation about $\boldsymbol{B}_0$. For a more complicated chemical structure there may be several different chemical groups, each with different resonant frequencies, resulting in a spectrum that exhibits a fine structure which can be used to characterise the chemical groups present in the sample. This phenomenon where different chemical groups have different resonant frequencies is known as the chemical shift.

The effect of the RF radiation on the net magnetisation $\boldsymbol{M}_0$ is highly complicated and it is instructive to introduce a rotating frame of reference, where the

frame rotates about the z-axis at frequency ω_0. Equation 2.1 can be easily generalised to describe the ensemble of protons that results in a net magnetisation along $\boldsymbol{B}_0$, such that

$$\frac{d\boldsymbol{M}_0}{dt} = \gamma \boldsymbol{M}_0 \wedge \boldsymbol{B}_0 \qquad [2.10]$$

Relating the stationary frame of reference to the rotating frame

$$\left(\frac{d\boldsymbol{M}_0}{dt}\right)_{stat} = \left(\frac{d\boldsymbol{M}_0}{dt}\right)_{rot} + \omega \wedge \boldsymbol{M}_0 \qquad [2.11]$$

and substituting Eq. 2.10 into Eq. 2.11

$$\left(\frac{d\boldsymbol{M}_0}{dt}\right)_{rot} = \gamma \boldsymbol{M}_0 \wedge \left(\boldsymbol{B}_0 + \frac{\omega}{\gamma}\right) = \gamma \boldsymbol{M}_0 \wedge \boldsymbol{B}_{eff} \qquad [2.12]$$

giving the Bloch equations which describe the magnetisation evolution. Thus, in the rotating frame of reference, $\boldsymbol{M}_0$ precesses about the effective magnetic field $\boldsymbol{B}_{\text{eff}}$. In the case when $\omega = \omega_0 = -\gamma B_0$ then $\boldsymbol{B}_{\text{eff}} = 0$ and $\boldsymbol{M}_0$ is invariant in the rotating frame of reference, as is expected as this is the condition of Larmor precession.

The addition of a field $\boldsymbol{B}_1$ applied at the resonant frequency ω_0, i.e. also rotating at frequency ω_0 in the stationary frame, results in $\boldsymbol{B}_{\text{eff}} = \boldsymbol{B}_1$ at resonance, as the rotating frame rotates with frequency $\omega = \omega_0 = -\gamma B_0$. Equation 2.12 demonstrates how the net magnetisation will rotate about the additional field $\boldsymbol{B}_1$. If $\boldsymbol{B}_1$ is applied along the x' axis (the x axis in the rotating frame), then $\boldsymbol{M}_0$ will rotate about the x' axis as is illustrated in Fig. 2.5. The frequency of rotation will be given by

$$\omega_1 = -\gamma B_1 \qquad [2.13]$$

and the angle of rotation θ can be calculated from

$$\theta = \gamma B_1 t_p \qquad [2.14]$$

where t_p is the time for which the field $\boldsymbol{B}_1$ is applied. This concept is fundamental to the technique of pulsed NMR, where a burst of RF radiation is applied to rotate $\boldsymbol{M}_0$ into the x'-y' plane. If a RF pulse rotates $\boldsymbol{M}_0$ from the equilibrium alignment along z' into alignment along y', then the angle θ is 90° and the pulse is known as a 90° pulse. Similarly a pulse that rotates $\boldsymbol{M}_0$ from z' to $-z'$ is a 180° pulse. A component of $\boldsymbol{M}_0$ along y' will then induce a current in a coil placed along the y-axis, which can be detected by a system tuned to the Larmor frequency (i.e. effectively measuring along y'). The resulting transient signal behaviour is then measured as the magnetisation

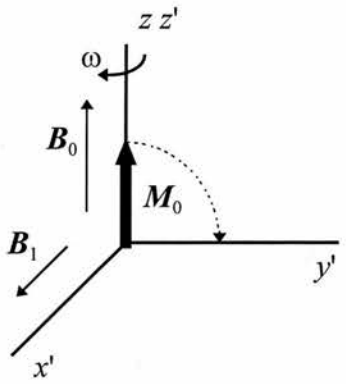


Figure 2.5 Diagram illustrating how the net magnetisation $\boldsymbol{M}_0$ is rotated about a magnetic field $\boldsymbol{B}_1$ applied along the x' axis in a frame of reference rotating at resonance such that $\omega = \omega_0 = -\gamma \boldsymbol{B}_0$.

returns to its equilibrium state, and the resulting signal is called the free induction decay (FID). Fourier transformation of the FID then gives the NMR absorption spectrum as in the case of the continuous wave experiment. The transient relaxation properties as the spins return to their equilibrium state is the subject of the next section.

2.4 Relaxation mechanisms

2.4.1 Spin-lattice T_1 relaxation

Absorption of RF energy causes a heating effect which disturbs the equilibrium Boltzmann distribution by increasing the spin system temperature T_s, resulting in an increase in the number of spins in the higher energy state $n_\downarrow$ as seen from Eq. 2.8. Spin-lattice relaxation is the process by which the spins reorder themselves to the equilibrium Boltzmann distribution following the absorption of RF energy at the Larmor frequency. The rate at which the net equilibrium magnetisation $\boldsymbol{M}_0$ returns to its initial alignment along $\boldsymbol{B}_0$ is characterised by the spin-lattice relaxation time T_1. The relaxation is brought about by interactions between magnetic fields fluctuating at the correct frequency in the x-y plane, which can result in an exchange of energy between the spin and lattice systems. In water, random molecular motion results in dipolar fields from the constituent protons fluctuating with time. This process can induce transitions in neighbouring protons, if the fluctuations occur at the correct frequency. This is the well known dipole-dipole

interaction which has a dependency on r^{-3}, where r is the distance between the protons. The interaction principally results from dipole fluctuations with frequency components occurring at ω_0 and $2\omega_0$. Other principal mechanisms that result in energy transfer between the spin and lattice systems include electric quadrupole, spin rotation and paramagnetic centre interactions. Paramagnetic molecules produce particularly strong T_1 relaxation because their magnetic moments are similar to that of an electron, which is some 10^3 times greater than the nuclear magnetic moment, resulting in the production of very strong fluctuating magnetic fields.

The rate of change of the magnetisation $\boldsymbol{M}_z$ parallel to $\boldsymbol{B}_0$ after excitation by an RF pulse is a first order process given by

$$\frac{dM_z}{dt} = -\frac{M_0 - M_z}{T_1} \qquad [2.15]$$

which results in the magnetisation recovery towards the equilibrium value $\boldsymbol{M}_0$ being exponential. The random motion of the molecules which causes the fluctuating fields is given by an activation energy expression and is characterised by the rotational correlation time τ_c, where

$$\tau_c = \tau_0 \exp\left(E_A / kT_L\right) \qquad [2.16]$$

and τ_0 is a time constant defining the motion, E_A is the activation energy, k is the Boltzmann constant and T_L is the lattice temperature. For a randomly fluctuating molecule, the frequency distribution of the motion is characterised by a spectral density function $J(\omega)$, such that

$$J(\omega) = \frac{\tau_c}{1 + \omega_0^2 \tau_c^2} \qquad [2.17]$$

Fluctuating molecules with a large spectral density around the Larmor frequency will clearly excite more transitions between spin states and therefore have a large effect on T_1. Also, T_1 will be affected by the magnitude of the magnetic field in the x-y plane B_{xy}, so

$$\frac{1}{T_1} \propto B_{xy}^2 \frac{\tau_c}{1 + \omega_0^2 \tau_c^2} \qquad [2.18]$$

It can be seen that when the molecules are fluctuating with a correlation time $\tau_c = 1 / \omega_0$, T_1 will take a minimum value because the relaxation processes are most efficient

at the Larmor frequency. This analysis provides a simple quantitative description of T_1 spin-lattice relaxation. In reality the biological situation is more complicated as spins exist in various bound and interacting states, but in general, T_1 relaxation remains an exponential process that is dependent upon the dipole-dipole interaction. A more detailed discussion of these processes can be found in texts such as Abragam (52), and Farrar and Becker (53).

2.4.2 Measurement of T_1 relaxation times

There are several different pulsed NMR methods capable of measuring T_1, but of these the 180° - τ - 90° inversion recovery sequence is relatively simple to implement. Firstly, a 180° RF pulse is applied that inverts $\boldsymbol{M}_0$ from along $+z$ to $-z$ such that $\boldsymbol{M}_z = -\boldsymbol{M}_0$. The magnetisation is then allowed to gradually recover along the $+z$ direction, governed by the spin-lattice relaxation time T_1. Subsequent application of a 90° RF pulse will tip the magnetisation aligned along the z axis into the x-y plane, inducing a current in the receiver coil with an amplitude $A(\tau)$ given by

$$A(\tau) = A(0)\left[1 - 2\exp\left(-\frac{\tau}{T_1}\right)\right] \qquad [2.19]$$

The experiment is illustrated in Fig. 2.6 and can be repeated for different values of τ such that the exponential in Eq. 2.19 is fitted to determine a value for T_1. It is necessary to wait for a period of at least 4 - $5T_1$ before repeating the experiment, in order that the magnetisation has returned to its equilibrium value $\boldsymbol{M}_0$ along z. Equation 2.19 demonstrates that the measured signal $A(\tau)$ will be zero when $\tau = \tau_{null} = T_1 \ln 2$. This feature of the inversion recovery sequence can be exploited to suppress unwanted signals in an imaging sequence. If a preparatory 180° pulse is applied with a delay of τ_{null} (determined by the T_1 of the substance to be suppressed) before the imaging acquisition pulses are applied, then the subsequent image will show zero signal for the suppressed substance (see Fig. 1.1d for a CSF suppressed example).

A second method for measuring T_1 relaxation is to apply a series of 90° - τ - 90° pulses; the progressive saturation technique. If the 90° pulses are applied with a repetition rate τ that is short compared to T_1, the magnetisation only partially recovers before the next 90° pulse and an equilibrium condition is established such

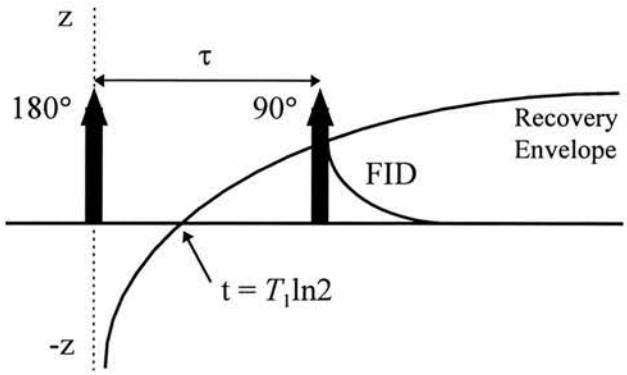


Figure 2.6 Illustration of the 180 - τ - 90 inversion recovery pulse sequence, demonstrating the recovery of signal to the equilibrium state.

that the z magnetisation, M_z, immediately proceeding a 90° readout pulse is

$$M_z = M_0\left[1-\exp\left(-\frac{\tau}{T_1}\right)\right] \qquad [2.20]$$

The experiment can be repeated with different repetition rates τ and as τ is reduced, the equilibrium magnetisation falls as the system is progressively saturated. The exponential in Eq. 2.20 can then be fitted enabling T_1 to be obtained. This method is reliant on the accuracy of the 90° pulses and can only be used for a limited range of T_1 values because the 90° pulses can only be applied at a maximum rate determined by the minimum acquisition time. Also, τ must be less than the spin-spin relaxation time. In many practical situations, partial saturation arises because measurements must be repeated with a repetition rate that is less that 4 - 5 T_1. In a biological system where tissues have many different T_1 values, differential saturation will occur resulting in contrast being displayed between the different tissues.

2.4.3 Spin-spin T_2 relaxation

After the RF excitation of protons into the transverse x-y plane, the spins interact with one another and precess at slightly different frequencies, losing their phase coherence. This process results in a signal decay, governed by a relaxation time constant T_2, as the net magnetisation in the x-y plane is destroyed. Each proton produces a small magnetic field at its neighbour, governed by the dipole-dipole interaction discussed in §2.4.1. Therefore, individual protons will exist in slightly

different magnetic environments resulting in a spread of static magnetic field values experienced by the different protons. This spread of magnetic fields will result in different protons precessing at different resonant frequencies depending on the field they experience, resulting in a broadening of the NMR absorption spectra. Thus, after the application of an RF pulse, components of the magnetisation in the *x*-*y* plane will not remain constant in a particular direction, but will reduce as the protons dephase, resulting in a reduction of signal over time, measured with a coil placed along *y*. A second T_2 process occurs due to spin-exchange mechanisms, where a proton oscillating with a Larmor frequency identical to a nearby proton induces a transition between spin states. This mutual exchange of energy results in one proton flipping from $n_\uparrow$ to $n_\downarrow$, while the other undergoes the opposite transition. Again, this process of spin-exchange results in a broadening of the absorption spectra.

Diffusion and reorientation of nuclei both contribute to T_2 by producing variations in the magnetic fields along and perpendicular to $\boldsymbol{B}_0$. The spin-spin relaxation term T_2 depends on magnetic field variations both in the *x*-*y* plane and along *z*, as both of these processes result in a broadening of the NMR absorption spectrum and a loss of phase coherence in the *x*-*y* plane. In contrast, T_1 relaxation only results from interactions occurring perpendicular to $\boldsymbol{B}_0$, as only these can induce transitions between the energy levels of the spin system. As such, all the processes that contribute to T_1 relaxation also contribute to T_2 relaxation, but T_2 relaxation is also dependent on fluctuations along $\boldsymbol{B}_0$, consequently T_2 relaxation times will always be shorter than T_1.

Again, the spin-spin interaction is assumed to be a first order process, resulting in an exponential decay of the component of magnetisation $\boldsymbol{M}_0$ in the *x*-*y* plane after application of an RF pulse, such that

$$\frac{dM_j}{dt} = -\frac{M_j}{T_2} \qquad \text{where } j = x', y' \qquad [2.21]$$

This equation, along with Eq. [2.15] describes the evolution of the magnetisation in the rotating frame after RF excitation - the Bloch equations with relaxation terms (1). A vector representation of the full Bloch equation representing the evolution of the magnetisation $\boldsymbol{M}$ is presented in Eq. 2.22,

$$\frac{d\boldsymbol{M}}{dt} = -\frac{M_{x'}}{T_2}\boldsymbol{i} - \frac{M_{y'}}{T_2}\boldsymbol{j} + \frac{\left(M_0 - M_{z'}\right)}{T_1}\boldsymbol{k} + \gamma \boldsymbol{M} \wedge \left(\boldsymbol{B}_1 + \Delta \boldsymbol{B}_0\right) \qquad [2.22]$$

where $\boldsymbol{i}, \boldsymbol{j}$ and $\boldsymbol{k}$ are unit vectors along the x', y' and z' directions, respectively. The field offset term $\Delta \boldsymbol{B}_0$ allows for applied magnetic field gradients to be considered in the Bloch equation. So far, only natural processes contributing to the line broadening and loss of phase coherence have been considered. In reality, the static magnetic field will not be homogeneous across the sample, resulting in a spread of $\boldsymbol{B}_0$ values over the sample, giving a spread of resonant frequencies. The observed spin-spin relaxation time T_2^* will then be given by

$$\frac{1}{T_2^*} = \frac{1}{T_2} + \frac{\gamma \Delta B_0}{2} \qquad [2.23]$$

where ΔB_0 is the variation in magnetic field over the sample. The resultant line broadening of the NMR absorption spectrum is then given by $1 / T_2^* = \Delta\omega_0 / 2$. Other mechanisms can lead to line broadening and thus additional T_2 relaxation, and these include magnetic susceptibility induced gradients within the sample and broadening associated with very short T_1 relaxation times. If T_1 is extremely short then the lifetime of a nucleus in a given state will be extremely short resulting in a lifetime broadening where the uncertainty in the resonant frequency is given by $\Delta\omega_0 \approx 1 / T_1$.

2.4.4 Measurement of T_2 relaxation times

The fundamental concept behind measurements of T_2 is the spin-echo, which utilises a 90° - τ - 180° RF pulse scheme and was first discovered by Hahn (5). Equation 2.23 illustrates how magnetic field inhomogeneities preclude the direct measurement of T_2 (only T_2^* measurements are possible) from the FID or NMR spectral linewidth, unless $T_2 << 2 / \gamma \Delta B_0$. Even if magnetic field inhomogeneities can be avoided, there is an uncertainty as to the start of the FID due to receiver dead time and the finite length of the RF pulses, which should ideally be delta functions. Therefore, unreliable T_2 measurements are likely to result when made directly from the FID data. The spin-echo experiment itself does not produce very accurate measurements of T_2, but forms the basis of other pulse sequences that lead to a more accurate determination.

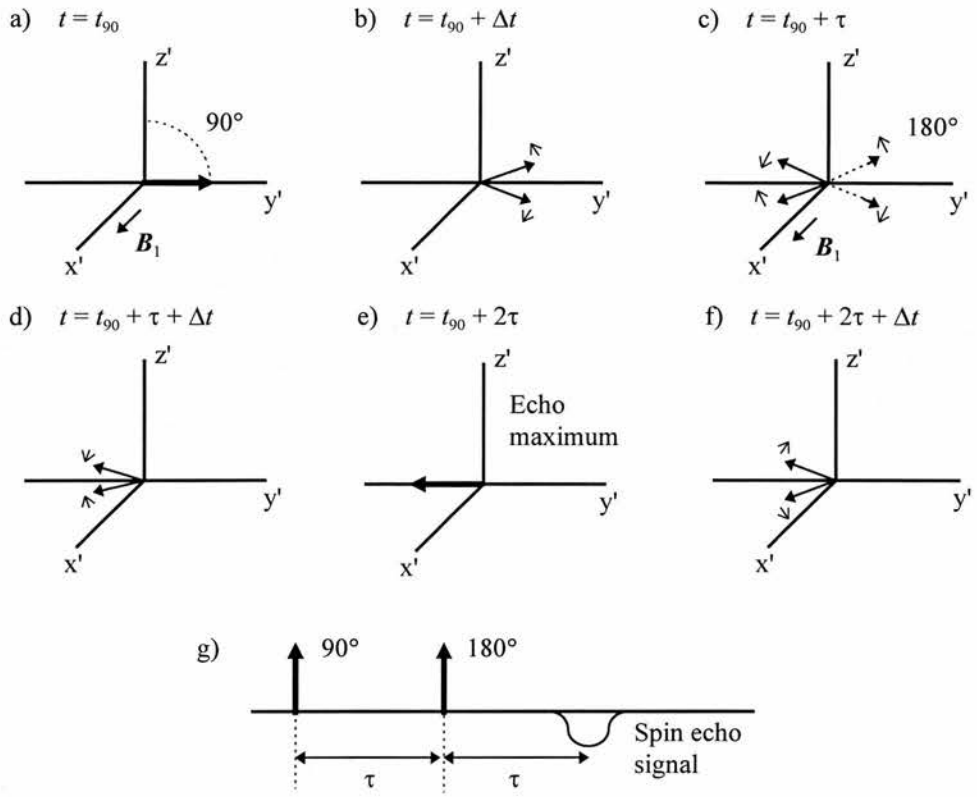


Figure 2.7 Illustration of the stages of proton dephasing and echo formation for a 90° - τ - 180° spin-echo sequence.

A pictorial representation of the spin-echo sequence is given in Fig. 2.7, and a brief description follows. An initial 90° pulse along x' rotates the initial magnetisation $\boldsymbol{M}_0$ into the x-y plane along y' (Fig. 2.7a). Some of the protons precess with a frequency less than ω_0 and some precess with a frequency greater than ω_0, such that the protons lose phase coherence with one another in the x-y plane (Fig. 2.7b). Subsequent application of a 180° RF pulse along x' after time τ results in the spins being rotated through 180° about x' (Fig. 2.7c), and spins that were rotating clockwise, now rotate counter-clockwise, and vice-versa (Fig. 2.7d). At a later time τ after the 180° pulse, the spins will all refocus along the $-y'$ axis (Fig. 2.7e) and then continue to dephase in the negative direction (Fig. 2.7f). This sequence of 90° - τ - 180° results in a rise and fall of negative signal - a spin-echo, which is induced in the receiver coil, centred around a time 2τ (Fig. 2.7g).

The amplitude of the spin-echo signal will be determined by T_2 because spin-spin relaxation, dependent on the dephasing time τ, will have taken place. The effects of magnetic field inhomogeneities will be essentially eliminated if the

individual protons remain in the same magnetic field during the course of the experiment, i.e. the decay due to magnetic inhomogeneity is a coherent process. However, molecular diffusion is an incoherent process (19) and if it occurs between the 90° and 180° pulses then the echo amplitude will be affected as the spins will experience different magnetic fields as they 'diffuse'. Taking into account diffusion processes (14-17), the echo amplitude at time 2τ after the 90° pulse is given by

$$A(2\tau) = A(0)\exp\left(-\frac{2\tau}{T_2} - \frac{2}{3}\gamma^2 G^2 D\tau^3\right) \qquad [2.24]$$

where G is the spatial magnetic field gradient and D is the diffusion coefficient. Measurements can be obtained at different τ intervals and a fitting procedure performed on Eq. 2.24, to obtain values for T_2. Consecutive experiments can only be performed after a time of around $5T_1$ has elapsed, to allow $\boldsymbol{M}_0$ to return to equilibrium. The second term in Eq. 2.24 becomes significant as τ increases, resulting in the spin-echo method only providing an accurate T_2 measurement for short T_2 values.

A modified spin-echo sequence was suggested by Carr and Purcell (14), to overcome the problems associated with diffusion. They apply a train of 180° pulses after the initial 90° pulse such that the sequence takes the form 90° - τ - 180° - 2τ - 180° - 2τ - ..... where all the RF pulses are applied along the x' axis. The sequence proceeds as for the spin-echo sequence illustrated in Fig. 2.7a - f. However, after the first spin echo has formed, a second 180° pulse is applied at time 3τ after the initial 90° pulse resulting in a second spin echo occurring in the $+y'$ direction at time 4τ. Subsequent 180° pulses are applied at times 5τ, 7τ, 9τ etc. resulting in the formation of alternate positive and negative spin echoes at times 6τ, 8τ, 10τ etc. as illustrated in Fig. 2.8. The amplitudes of the resultant spin echoes are given by

$$A(t) = A(0)\exp\left(-\frac{1}{T_2} - \frac{1}{3}\gamma^2 G^2 D\tau^2 t\right) \qquad [2.25]$$

where t is the total length of time for which the sequence is applied. Each subsequent echo in the sequence will have a reduced amplitude determined by T_2 relaxation and measurements of these amplitudes can be fitted to Eq. 2.25, allowing the calculation of T_2. Diffusion effects can be reduced considerably by using a very short τ, thus

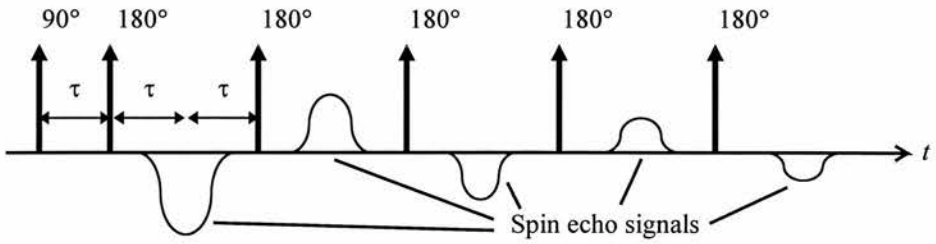


Figure 2.8 The Carr-Purcell modification to the spin echo sequence, where a train of 180° pulses follows the initial 90° pulse. The formation of the spin echoes at 2τ, 4τ , 6τ etc. is shown and their echo amplitude decays over time with T_2.

minimising the second term in Eq. 2.25. This sequence enables measurements of T_2 to be obtained in a very short time because only one sequence is needed of approximate length T_2, whereas the spin-echo sequence requires several different 90° - τ - 180° acquisitions, with a time of $\approx 5T_1$ between them. However, for measurements of long T_2, a large number of 180° pulses are required and accuracy could be compromised if the 180° pulse lengths and $\boldsymbol{B}_1$ are not constant. These problems can be largely overcome by modifying the sequence as proposed by Meiboom and Gill (54), such that the 180° pulses are applied along the y' axis, i.e. $\pi/2$ out of phase with the 90° pulse, reducing the sensitivity of the experiment to pulse imperfections. This modified sequence is known as the Carr-Purcell Meiboom-Gill (CPMG) sequence and is the method of choice for measuring T_2 relaxation times.

2.4.5 NMR signal acquisition

After RF excitation of the spins into the x-y plane, the NMR signal is detected by the EMF it induces in a receiver coil placed along the y direction, tuned to ω_0. The induced EMF oscillates at radiofrequency (ω_0) and is converted to low (audio) frequencies (demodulated) using a phase sensitive detector. Audio frequencies are desirable because they allow the use of simpler and cheaper electronics. The NMR signal is digitised, as determined by the sampling rate, which must be sufficient such that the Nyquist condition in Eq. 2.26 is satisfied,

$$\omega_{nq} = \frac{n_s}{2} t_{aq} \qquad [2.26]$$

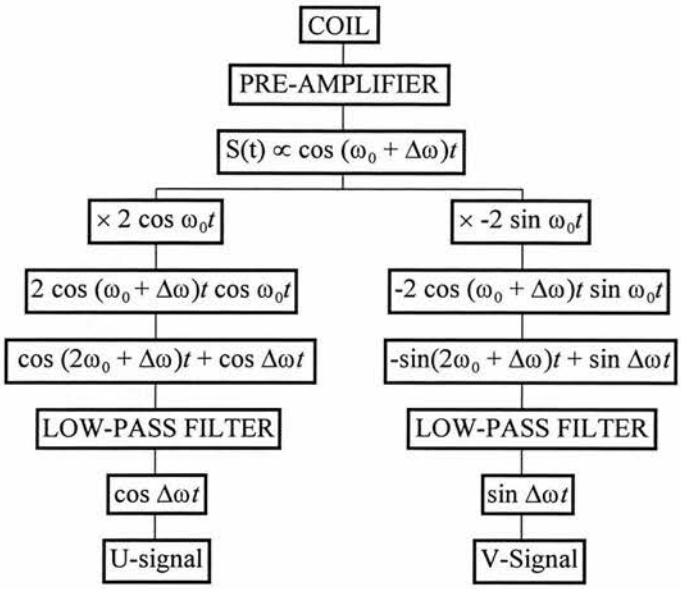


Figure 2.9 A schematic representation of the quadrature phase sensitive signal detection method to obtain complex NMR data using two reference signals 90° out of phase (in quadrature) with one another.

where ω_{nq} is the Nyquist sampling frequency, n_s is the number of sampling points and t_{aq} is the sampling time. The Nyquist frequency is the maximum frequency that can be accurately sampled using the defined sampling time and number of data points.

The phase sensitive detector consists of two channels, capable of measuring different components of the magnetisation in the *x*-*y* plane relative to a reference phase. The two reference phases are applied 90° out of phase (in quadrature) with one another. This technique is advantageous because it allows for a $\sqrt{2}$ improvement in SNR and allows the detection of both positive and negative phase angles. A schematic representation of the quadrature phase sensitive detector is given in Fig. 2.9. If an RF pulse rotates a spin isochromat, precessing at ω_0, into the *x*-*y* plane then phase sensitive detection at ω_0 with reference phase corresponding to the *y*' frame will detect a steady dc voltage. Another channel with the phase shifted by 90° along the *x*' direction would detect no signal as the input signal has no component in this direction. Consider a second set of spins, precessing at a frequency $\omega \neq \omega_0$, while the same RF pulse frequency and reference frequency for the two phase sensitive detectors are used. The spins will now precess at a frequency $\Delta\omega$ about *z*' and the NMR signal will be sinusoidal with a phase dependence on the

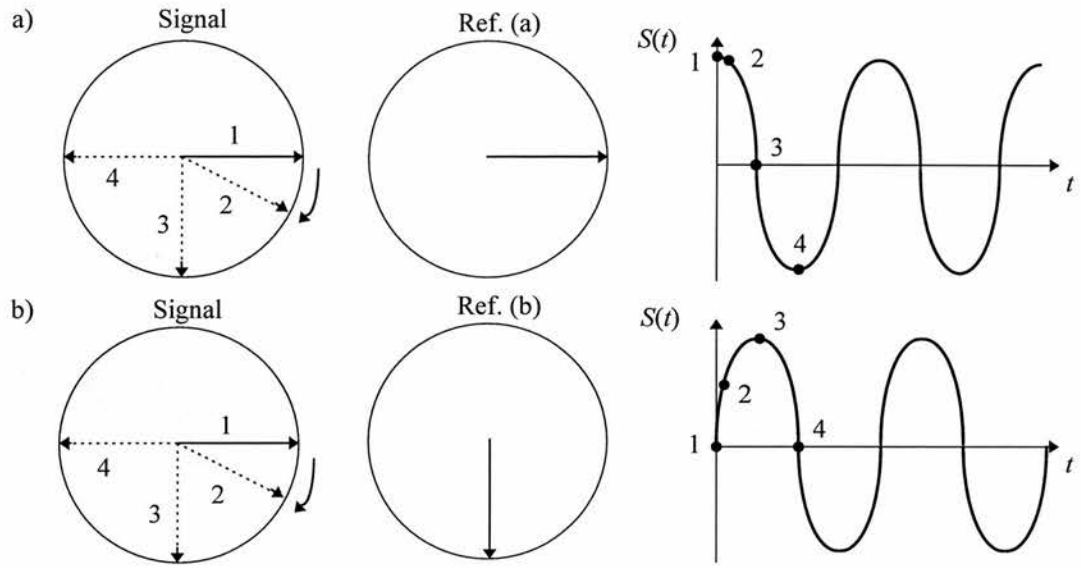


Figure 2.10 Signals obtained using quadrature detection from off-resonance protons. The reference phases in (a) and (b) are in quadrature.

reference phase, as illustrated in Fig. 2.10. The measured signals from the two quadrature channels are conveniently described as the real and imaginary components, a procedure which lends itself particularly well to subsequent Fourier transform.

2.4.6 Fourier analysis of the NMR signal

A spin system that absorbs a single RF frequency leads to a harmonic signal of frequency determined by the difference $\Delta\omega$ between the Larmor precessional frequency and the spectrometer reference frequency (the central frequency of the RF pulse). Protons precessing at this frequency have $\Delta\omega = 0$, resulting in a constant NMR signal, as shown in §2.4.5, and subsequent frequency analysis indicates the presence of a single zero frequency component. For the case where $\Delta\omega = \omega_1$, the signal is a simple function of the form $\cos\omega_1 t$ and subsequent frequency analysis results in a single frequency component or delta function at ω_1. For the case where two frequency offsets are present at ω_1 and ω_2, the signal will be of the form $S(t) = \frac{1}{2}$ $[\cos\omega_1 t + \cos\omega_2 t]$ and the corresponding frequency analysis has two components at ω_1 and ω_2, with half the amplitude of the previous cases. The signal takes the form of a typical beat pattern, where $(\omega_1 + \omega_2) / 2$ is the carrier frequency and $(\omega_1 - \omega_2) / 2$

is the beat frequency. It can be seen that the addition of further frequency components, to represent a realistic situation, would soon lead to a highly complicated time varying signal and the frequency analysis of such a function would be non-trivial.

If the assumption is made that the frequency components that form the time varying signal can be expressed as discrete multiples of a certain frequency ω_s, then the signal $S(t)$ can be written as

$$S(t) = a_0 + \sum_{n=1}^{\infty} a_n \cos n\omega_s t \qquad [2.27]$$

where a_n are the amplitudes of the discrete frequency components. The frequency analysis is then required to find the set of amplitudes a_n, for each frequency in the summation. The orthogonality of harmonic functions can be exploited whereby Eq. 2.27 is multiplied by $\cos m\omega_s t$ and integrated over time limits between $\pm \pi / \omega_s$, to give

$$\int_{-\pi/\omega_s}^{\pi/\omega_s} S(t)\cos m\omega_s t dt = \int_{-\pi/\omega_s}^{\pi/\omega_s} a_0 \cos m\omega_s t dt + \sum_{n=1}^{\infty} \int_{-\pi/\omega_s}^{\pi/\omega_s} a_n \cos n\omega_s t \cos m\omega_s t dt \qquad [2.28]$$

The first term on the right hand side vanishes for $m \neq 0$ and the second for $m \neq n$, giving

$$a_0 = \frac{\omega_s}{2\pi} \int_{-\pi/\omega_s}^{\pi/\omega_s} S(t) dt \qquad [2.29a]$$

$$a_{n\neq 0} = \frac{\omega_s}{\pi} \int_{-\pi/\omega_s}^{\pi/\omega_s} S(t) \cos n\omega_s t dt \qquad [2.29b]$$

Therefore, a spin isochromat of frequency $n\omega_s$ will contribute to the signal $S(t)$ with a proportion determined by a_n and can be evaluated by multiplying the signal by $\cos n\omega_s t$ and integrating.

In a realistic situation, the signal is determined from a continuous range of frequency components and this is accounted for by reducing the interval between discrete frequency components, such that $\omega_s \rightarrow 0$, and, in this limit, the sum in Eq. 2.27 becomes an integral. Thus, the signal is defined as

$$S(t) = \int_{-\infty}^{\infty} G(\omega) \cos \omega t d\omega \qquad [2.30]$$

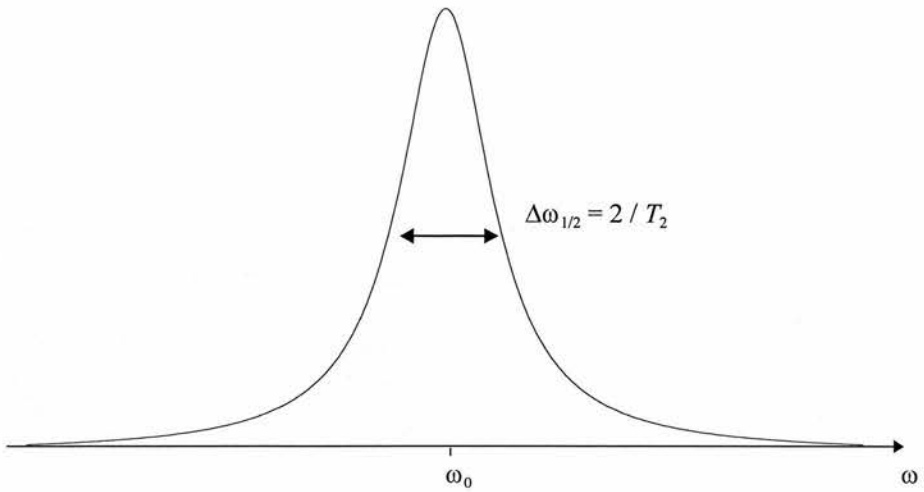


Figure 2.11 The lineshape function G(ω) for a single spectral frequency, in the absence of an applied magnetic field gradient, with no magnetic field inhomogeneities.

where $G(\omega)$ is a lineshape function that replaces the coefficients a_n of the discrete frequencies. The lineshape function can then be determined by inverse Fourier transformation to give

$$G(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S(t)\cos\omega t dt \qquad [2.31]$$

For the simple case of a single spectral frequency, in the absence of an applied field gradient, the FID envelope can be considered to be exponential, such that

$$S(t) = a\exp\left(-t / T_2\right) \qquad [2.32]$$

The frequency lineshape function that gives rise to this is given by the Fourier transform of $S(t)$ calculated from Eq. 2.31, leading to

$$G(\omega) = \frac{a / T_2}{\left(\omega - \omega_0\right)^2 + \left(\frac{1}{T_2}\right)^2} \qquad [2.33]$$

which is a Lorentzian line of full width half maximum (FWHM) $\Delta\omega_{1/2} = 2 / T_2$, as illustrated in Fig. 2.11. This example is a practical demonstration of how the line broadening of the frequency spectrum is determined by T_2 as was discussed in §2.4.3.

2.5 NMR image acquisition using magnetic field gradients

2.5.1 Magnetic field gradients and Larmor precession

The ability to acquire MR images requires a method for producing spatial localisation of the MR signal. The Larmor frequency of a proton was shown in §2.3.1 and §2.3.2 to be dependent on the magnetic field strength at the position where it is situated. If the static magnetic field $\boldsymbol{B}_0$ is made to vary in space, then the precessional frequencies ω_0 will reflect this spatial dependence. The application of a linear magnetic field gradient, typically varying by less than 1% of B_0, will result in a modification of the Larmor precession condition, such that

$$\omega_i = \gamma\left(B_0 + \boldsymbol{G}\cdot\boldsymbol{r}_i\right) \qquad [2.34]$$

where ω_i is the precessional frequency at position $\boldsymbol{r}_i$ and $\boldsymbol{G}$ is the total magnetic field gradient vector. Equation 2.34 illustrates that in the presence of an applied magnetic field gradient, each proton will resonate at a frequency dependent on its position within the gradient field. Thus, the MR image is generated by each position within the image having a unique magnetic field strength resulting in each proton having a unique frequency dependent on its position. The intensity in each pixel is then governed by the number of protons within the voxel, weighted by the T_1 and T_2 relaxation times for the tissues present.

If a constant magnetic field gradient g_0 is applied along the x-direction, the detected signal can be expressed as

$$S(t) = \int \rho(x)\exp\left[2i\pi\nu'(x)t\right]\exp\left(-t/T_2^*\right)dx \qquad [2.35]$$

where $\nu'(x) = \gamma g_0 x / 2\pi$ and $\rho(x)$ is the spin density along x (i.e. integrated over y and z). It can be seen that the frequency, ν', is linearly dependent on x, illustrating the spatial encoding of the frequency. At this stage it is prudent to introduce reciprocal k-space, as in x-ray crystallography, where

$$k_x = \frac{1}{2\pi}\gamma g_x t \qquad [2.36]$$

If the damping term due to T_2^* is ignored, then Eq. 2.35 can be rewritten as

$$S\left(k_x\right) = \int \rho(x)\exp\left(2i\pi k_x x\right)dx \qquad [2.37]$$

which demonstrates the Fourier relationship between the NMR signal $S(k_x)$ and the spin density $\rho(x)$. The inverse Fourier transform then yields the spin density

$$\rho(x) = \int S(k_x)\exp(-2i\pi k_x x)dk_x \qquad [2.38]$$

These equations demonstrate the fundamental NMR Fourier relationship between time and frequency. The signal $S(k_x)$ is a function of time (Eq. 2.36), while the spin density $\rho(x)$ is in the frequency domain as the position x is proportional to the frequency $\nu'(x)$. The expression given in Eq. 2.35 can be transposed into a discrete form, similar to that used in converting the Fourier integral into a discrete summation, so that

$$S(t) = \sum_{j=1}^{N} \rho_j \exp(i\gamma g_0 x_j t)\exp(-t / T_2^*) \qquad [2.39]$$

and ρ_j is the spin density pertaining to a thin slice at position x_j. The signal $S(t)$ represents a classical interferogram containing N signals of amplitude ρ_j at frequency $\gamma g_0 x_j / 2\pi$. Fourier transformation of $S(t)$ yields a spectrum of lines with intensity proportional to ρ_j at frequency $\gamma g_0 x_j / 2\pi$. Superimposition of these lines then represents the spin distribution along x, where x is expressed as units of $\gamma g_0 / 2\pi$.

2.5.2 Selective excitation and slice selection

A magnetic field gradient applied along a direction perpendicular to the desired image plane, conventionally z, results in all protons in a given x-y plane precessing at a unique frequency dependent upon their z position. A frequency selective RF pulse can excite an x-y plane of finite thickness determined by the magnetic field strength and variation along z, as illustrated in Fig. 2.12. The RF pulse has a central frequency (MHz), which determines the position of the slice along the z direction, and contains a narrow bandwidth (1-2 kHz) which gives the slice a finite thickness by exciting a range of frequencies. The duration and amplitude of the RF pulse determines the amount of proton rotation e.g. 90°, 180°. If a slice selective gradient G_{ss} is applied, then the slice thickness Δz can be calculated from the bandwidth of the RF pulse $\Delta\omega_{ss}$, such that

$$\Delta\omega_{ss} = \gamma\Delta(G_{ss} \times \Delta z) \qquad [2.40]$$

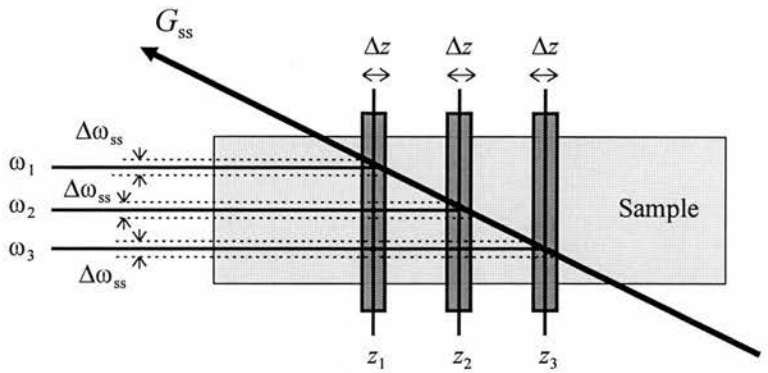


Figure 2.12 Illustration of the slice selection process. Protons resonate at a frequency ω_j determined by their position z_j in a magnetic field gradient G_{ss}. The slice thickness is determined by the gradient G_{ss} and the bandwidth of the RF pulse $\Delta\omega_{ss}$.

Usually, $\Delta\omega_{ss}$ is fixed and the slice thickness is altered by varying G_{ss}, whereby increasing G_{ss} results in a thinner slice being excited. Once the slice thickness has been determined, the central frequency of the RF pulse can be calculated from Eq. 2.34, to excite the required slice location along z. For multi-slice imaging, it is usual to keep G_{ss} constant for all slices and to have an individual RF pulse for each slice with a different central frequency and constant bandwidth.

2.5.3 Frequency encoding

Frequency encoding provides one of the visual dimensions in the imaging plane by the application of a 'frequency encoding' gradient G_{fe} during the signal acquisition period. This additional gradient G_{fe} is applied, perpendicular to the slice direction, during echo formation. This results in protons precessing at different frequencies depending on their position in the frequency encoding gradient field, in accordance with Eq. 2.34. These frequencies are superimposed into the image echo, which is digitised for subsequent Fourier analysis. The position of the proton in the gradient field is determined from the magnitude of G_{fe} and the frequency that is detected, as illustrated in Fig. 2.13. Fourier transformation along the frequency encoding direction enables the position of a proton to be displayed, by transforming the time domain FID into a frequency spectrum that gives the spatial distribution of protons, as demonstrated in §2.5.1.

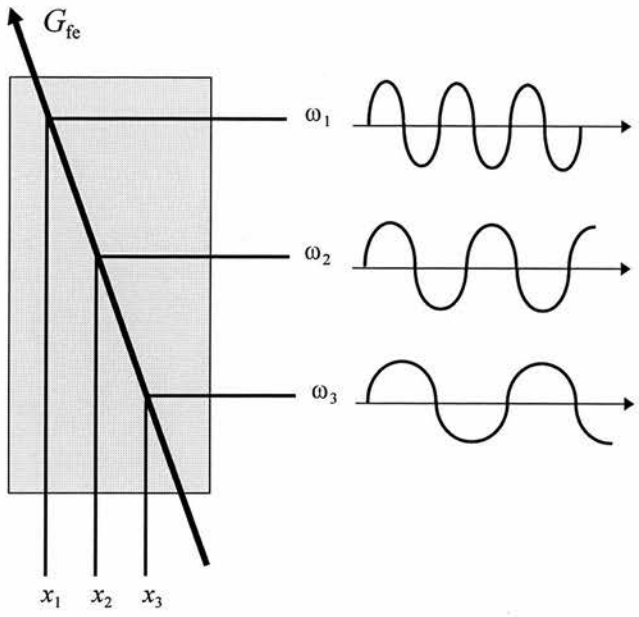


Figure 2.13 Illustration of the process of frequency encoding. The precessional frequency ω_j of each proton depends on its position x_j within the frequency encoding magnetic field G_{fe}.

The echo signal is digitised by sampling discrete data points over a sampling period t_{aq}. The maximum frequency that can be accurately sampled without aliasing is the Nyquist frequency ω_{nq}, given by Eq. 2.26. The readout gradient is selected such that protons at the edge of the field of view (FOV) will be sampled at the Nyquist frequency, while protons nearer to the centre of the FOV will resonate at a lower frequency. Smaller FOVs are achieved by increasing G_{fe} such that the Nyquist frequency and thus receiver bandwidth remain constant given by

$$\Delta\omega_{fe} = 2\omega_{nq} = \gamma\Delta\left(G_{fe} \times \mathrm{FOV}\right) \qquad [2.41]$$

If tissue is excited that is outside the FOV then this tissue will exceed the Nyquist limit and high frequency aliasing will result, whereby the signal will be mapped back to a lower frequency. To overcome this, the number of data points sampled can be doubled, while keeping G_{fe} and t_{aq} constant, thus doubling the Nyquist frequency as shown by Eq. 2.26. As G_{fe} is constant, the frequencies for the chosen FOV are unchanged and so the desired image can be reconstructed from the central range of data points. This process is known as oversampling.

2.5.4 Phase Encoding

The final image direction is visualised by the application of a phase encoding gradient G_{pe}, perpendicular to G_{ss} and G_{fe}. While the phase encoding gradient is applied the precessional frequency of the protons either increases or decreases as determined by Eq. 2.34. Once G_{pe} is turned off, the protons precess at their original frequency, but will be either ahead or behind in phase relative to unencoded protons with $G_{pe} = 0$. The phase shift induced by G_{pe} (conventionally applied along the y direction) will be determined by the magnitude and duration of G_{pe} such that

$$\phi = \int_0^{\tau} \gamma G_{pe}(t) y dt = \gamma y \int_0^{\tau} G_{pe}(t) dt \qquad [2.42]$$

where τ is the time for which G_{pe} is applied for. Equation 2.42 demonstrates how the amount of phase shift induced depends on the position along y, whereby protons located at the edge of the FOV experience the maximum phase shift, as illustrated in Fig. 2.14. The image is formed by repeating the excitation and signal acquisition

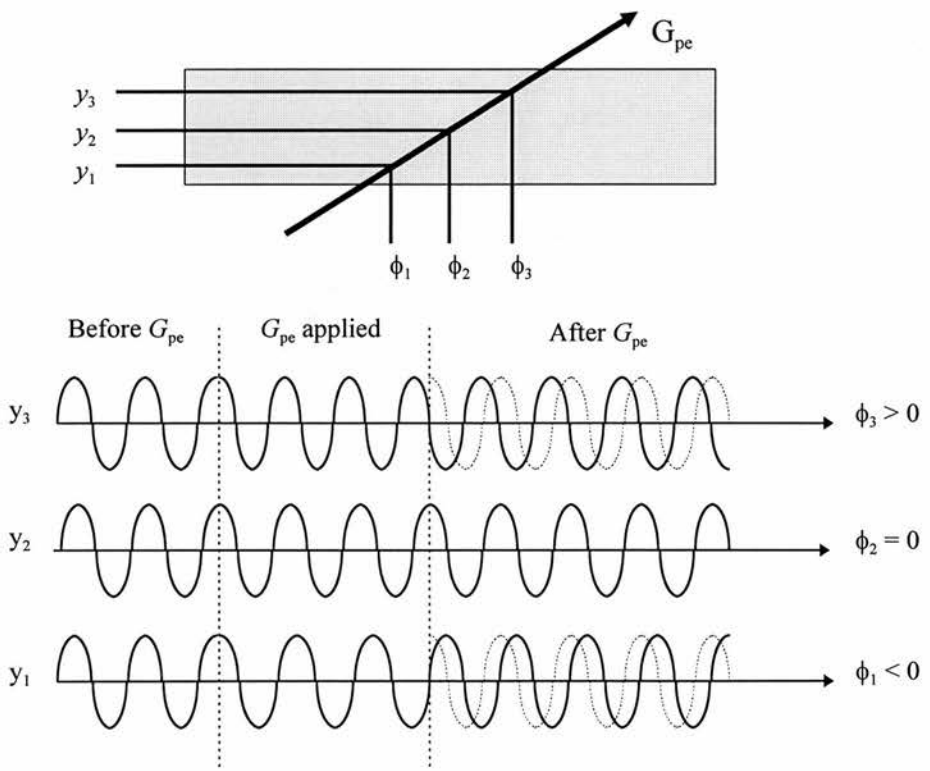


Figure 2.14 Illustration of the phase encoding process. While the phase encoding gradient is applied protons precess at a different frequency depending on their y position y_j, resulting in a phase shift ϕ_j dependent on G_{pe}.

several times, each with a different value for G_{pe} applied prior to signal detection. The second Fourier transform is then applied along the phase encode direction, such that the signal amplitude at each readout frequency is converted from a function of time (or k_y) into a function of frequency (or y position).

The resolution in the phase encoding direction is determined from the FOV in this direction and the number of phase encoding increments n_{pe}. The FOV is governed by the change in G_{pe} between successive phase encoding steps. This condition can be expressed mathematically, whereby the phase accumulated during the smallest non-zero gradient $G_{pe(min)}$, is determined such that, by the extremity of the sample y_{max}, a total phase accumulation of π occurs, so that

$$\gamma y_{max} \int_0^{\tau} G_{pe(min)}(t)dt = \pi \qquad [2.43]$$

As with the frequency encoding direction, aliasing can occur when protons outside the FOV are excited. These protons undergo phase changes that correspond to frequencies greater than the Nyquist limit and are thus mapped to lower frequencies in the subsequent Fourier analysis. Phase encoding aliasing can only be overcome by increasing the FOV in that direction, either by reducing the gradient amplitude from one phase encoding step to the next or by increasing the Nyquist sampling frequency.

2.6 Two-dimensional NMR Imaging strategies

2.6.1 k-space sampling

The concept of reciprocal k-space was briefly introduced in §2.5.1 to simplify the Fourier analysis of a spin system in the presence of a magnetic field gradient. The fundamental Fourier relationship between the NMR signal in the time domain and the NMR spectrum obtained in the frequency domain was discussed in §2.4.6. It was also demonstrated that under the application of a magnetic field gradient, the precessional frequency becomes proportional to the position in the field. With the introduction of a reciprocal space as given by Eq. 2.36, the Fourier transform relationship between time and frequency is transposed into a relationship between reciprocal space and real (Cartesian) space (Eqs. 2.37-8). As such r and k_r are Fourier transform pairs, in the same way as ω and t. Equation 2.36 also demonstrates

how k_x is proportional to both t and G_x and so the application of a magnetic field gradient results in the sampling of reciprocal k-space at intervals determined by the gradient strength G_x and the time t, as shown in Eq. 2.44.

$$k(t) = \gamma \int_0^t G(t')dt' \qquad [2.44]$$

The design of pulse sequence acquisition schemes can then be facilitated by observing the effect of the gradient and signal acquisition strategies on the sampling of k-space.

The central portion of k-space, corresponding to low amplitude G_{pe} and the central region of the readout data, contains most of the signal content and provides the majority of the image contrast, while the edges of k-space provide the fine detail from high frequency components. The k-space data (raw data) has a Hermitian symmetry due to the positive and negative G_{pe} values introducing opposite phase shifts to the protons. This symmetry can be exploited by using half Fourier techniques to reduce the acquisition time. The half Fourier approach samples approximately 60% of k-space, corresponding to the negative phase encoding values and the first few positive values. The regions of k-space corresponding to the positive G_{pe} values can be reconstructed using the symmetry properties of the data, while the additionally acquired positive k-space lines are used for phase correction and contrast. Half-Fourier imaging results in the same spatial resolution and FOV, but reduces the acquisition time by around 40%. However, the SNR is reduced and the method is more sensitive to artefacts arising from the duplication of data.

The k-space formalism results in a convenient description of the different sampling methods used to acquire the raw data matrix. The traditional method sequentially fills each line of k-space, corresponding to a different phase encoding step. Many different k-space sampling strategies have been devised, some providing improved image quality or reduced scan time, such as those utilising spiral k-space or zig-zag trajectories. The rest of this chapter discusses the different imaging sequences used in this study, and in particular how the combinations of gradient pulses, RF pulses and timing result in the sampling of k-space.

2.6.2 Spin-warp imaging

The spin-warp imaging technique was developed in 1980 by Edelstein et al. (10), and consists of a two-dimensional Fourier imaging strategy, which utilises frequency and phase encoding as described in §2.5.3 and §2.5.4. Spin warp imaging is a modified version of the original Fourier methods developed by Kumar et al. in 1975 (11). The original method utilised a constant phase encoding gradient G_{pe}, that was applied for varying degrees of time in order to produce the variable phase shift, as given by Eq. 2.42. The step taken by Edelstein et al. was to introduce a programmable, variable amplitude, phase encoding gradient pulse, in order to produce the desired variable phase shift. This constant time, stepped, gradient pulse has two advantages; it allows the use of both positive and negative gradients, without sign ambiguity; and as the gradient is applied for a constant time the relaxation effects are constant between phase encoding steps. The original spin-warp method was developed as a gradient echo sequence (Fig. 2.15a), but is equally suited to spin echo techniques as is illustrated in Fig. 2.15b. The k-space sampling representation of the imaging sequences is illustrated in Fig. 2.15c & 2.15d.

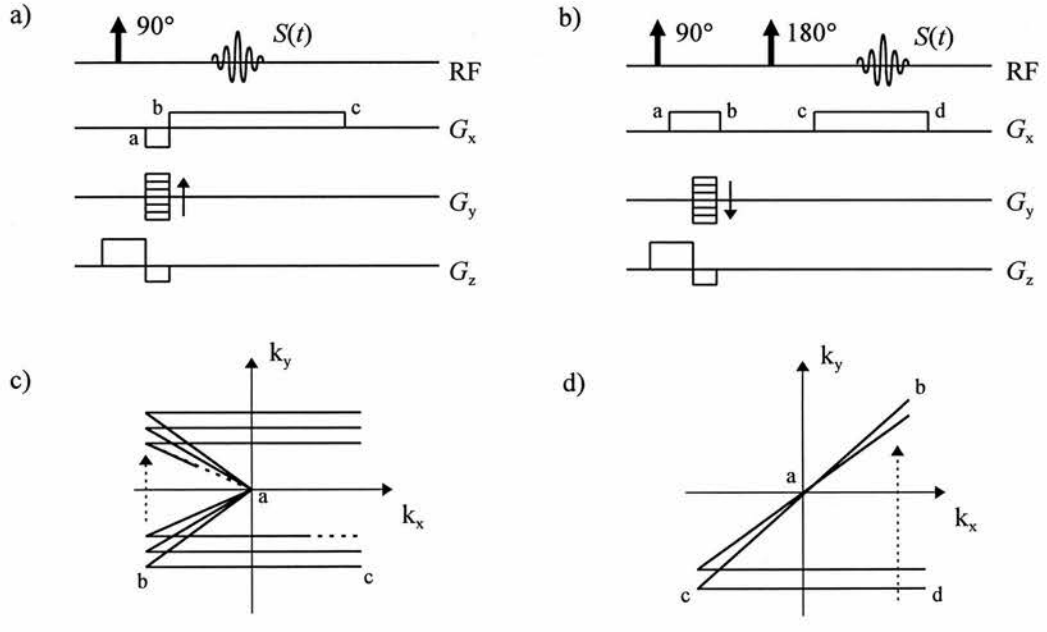


Figure 2.15 Schematic illustration of the gradient (a) and spin (b) echo forms of the spin-warp imaging sequence, along with their respective k-space trajectories (c) & (d).

2.6.3 Echo planar imaging

'Multi-planar image formation using NMR spin echoes' was first proposed by Mansfield et al. in 1977 (12). This technique subsequently became known as echo planar imaging and utilises a raster scan to sample the whole of k-space after a single RF excitation, hence the terminology 'single-shot' imaging. The whole data set must be acquired in around 100ms ($< T_2$), so that the signal has not decayed to the noise level, due to T_2 or T_2^* relaxation processes (55). Therefore, very fast gradient switching is required to 'move' through k-space rapidly and so high quality hardware is required. The k-space trajectory for a typical EPI sequence (in this case MBEST (56) or Instascan (57)) is given in Fig. 2.16b, showing the raster like traversal. The corresponding pulse sequence is given in Fig. 2.16a. The EPI sequence shown is of the spin-echo form, but can be easily modified to have a gradient echo preparation period prior to the acquisition.

After the initial RF excitation of the magnetisation into the transverse x-y plane, the trajectory is moved to the negative corner of the k-space field by application of a negative phase encode and readout gradient. The process then performs a raster scan by alternate application of readout and phase encode gradients. The phase encode gradient is applied as a positive short duration gradient 'blip', to move the trajectory between subsequent phase encoding lines (56). The readout gradient is applied alternately in the positive and negative directions, to perform the raster as indicated in Fig. 2.16a. The requirement for the EPI readout to be performed in < 100ms, means that the conjugate symmetry of k-space is frequently exploited, with EPI readout strategies utilising half-Fourier (58,59), zig-zag, spiral (60) and square-spiral (61) techniques.

EPI techniques are sensitive to a number of artefacts, particularly regarding errors in gradient timing or amplitudes. These result in an error in the position of the sampling points with respect to the time integral of the gradient, leading to improper location of data points in k-space. To achieve a regular grid of data points suitable for 2D-Fourier transform, data must be sampled either uniformly in the presence of a linear magnetic field gradient or non-uniformly in the presence of a non-linear gradient. The latter is often the case with EPI as time constraints mean that data

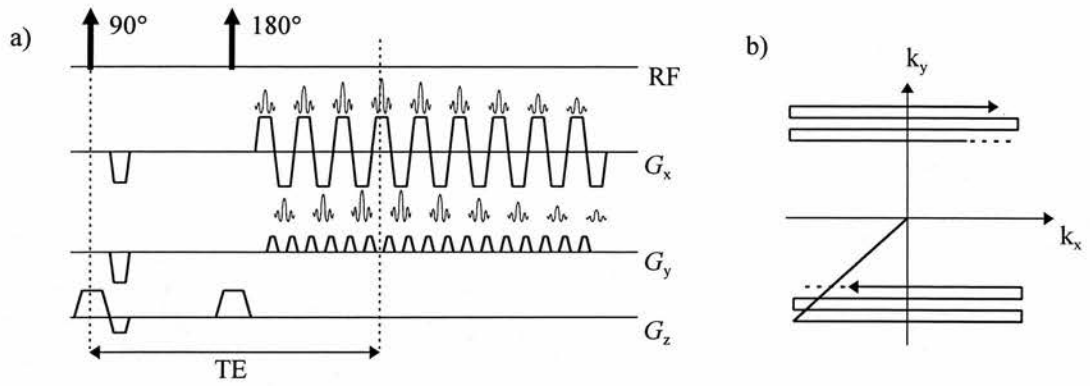


Figure 2.16 A spin-echo based echo-planar imaging sequence with MBEST acquisition scheme (a) and the corresponding k-space representation (b).

sampling is required on the rise and fall periods of the gradient pulses. Due to the fast rise and fall of the EPI gradient pulses, eddy currents are induced in the magnet bore (62) resulting in gradient field distortions. Therefore, gradient shielding is a prerequisite for EPI data acquisition (63). Further artefacts can result if the sampling rate is not precisely in time with the gradient field, but becomes offset by a few microseconds, which can occur if eddy currents are induced in the RF coils, resulting in a phase lag. As the EPI raster technique utilises alternate positive and negative trajectories in k-space, alternate lines must be reversed prior to Fourier transform. If a phase lag is present then this reversal will lead to a line dependent phase discrepancy, resulting in the classic Nyquist ghost, where a ghost image appears, shifted by half the FOV. These ghosts appear along the phase encoding direction, due to its low bandwidth per pixel.

The narrow bandwidth per pixel in the phase encoding direction also results in a large water-fat shift and geometric distortions due to $\boldsymbol{B}_0$ inhomogeneity. The bandwidth per pixel in the phase encoding direction is typically 10 Hz pixel^{-1} as opposed to 2000 Hz pixel^{-1} in the frequency encoding direction. The fat-water shift is around 3.3 ppm, resulting in a frequency difference of 210 Hz, which corresponds to a shift of 0.12 pixels in the frequency encoding direction and 21 pixels in the phase encoding direction. Clearly, the small shift in the frequency encoding direction can be ignored because it is considerably less than the pixel size, but the phase encoding shift results in a large, shifted, fat signal being observed if fat suppression measures

are not taken. The narrow bandwidth also results in an enhanced sensitivity to field inhomogeneities, either resulting from main field $\boldsymbol{B}_0$ variations or from local magnetic susceptibility differences. Main field inhomogeneities can be reduced by shimming procedures (see §4.2.2), as $\boldsymbol{B}_0$ inhomogeneities vary slowly across the magnet. However, susceptibility variations take the form of a step function in the field and so shimming is ineffective. Magnetic susceptibility artefacts result in large geometric distortions, particularly at tissue-air and tissue-bone interfaces, such as in the frontal sinuses, and are one of the primary inhibitions to the use of in vivo EPI.

3

MR Diffusion Imaging: Theory and Applications

3.1 Introduction

This chapter extends the theory described in Chapter 2 to explain how MR methods can be utilised to measure diffusivities. The fundamental concepts necessary for understanding the work performed in subsequent chapters is presented. A brief overview of molecular self-diffusion theory and how diffusion must be characterised by a tensor, is given in §3.2. In §3.3, it is demonstrated how diffusion measurements can be made using MR techniques, by making simple modifications to standard MR pulse sequences described in Chapter 2. In §3.4 it is shown how the diffusion tensor can be estimated from a series of diffusion measurements and how the tensor can be utilised to characterise the diffusion environment by means of a diffusion ellipsoid. An overview of the diffusion-weighted pulse sequences suggested in the literature is presented in §3.5, concentrating on the 2DFT navigated spin-echo and EPI spin-echo pulse sequences that were used during the course of this work. In §3.6, it is shown how mathematical tensor operations can be used to calculate physically meaningful, quantitative, scalar parameters for describing the diffusion environment. Finally, in §3.7 a summary of the clinical applications of diffusion imaging in the human brain is presented.

3.2 Self-diffusion and the diffusion tensor

3.2.1 What is diffusion?

Diffusion describes the random motion of molecules due to thermal processes, so called Brownian motion (65). The diffusion process is primarily observed in fluids rather than solids because diffusion in solids is extremely slow due to the relatively low thermal energy available when compared to the barriers separating atomic sites in a solid (66). Diffusion can take place in either a single pure substance (self-diffusion), or the diffusion of one type of molecule amongst others can be considered. Self diffusion is, conventionally, not directly observable because only when substances differ in some way, can flow and concentration measurements be undertaken. The conventional method for measuring self-diffusion has been to use two different isotopes of the same chemical element because their inter-diffusion will, to a good approximation, be equal to the self-diffusion of the single pure substance (67). This approximation holds particularly well for heavy elements, where the mass difference between the isotopes can be small compared to their atomic mass.

Diffusion is described by the classical relationships of Fick's Laws (68) which describe how the flux density J depends on the gradient concentration $c(\mathbf{r},t)$ of the sample.

$$J = -D\nabla c(\boldsymbol{r},t) \qquad [3.1a]$$

$$\frac{dc(\boldsymbol{r},t)}{dt} = D\nabla^2 c(\boldsymbol{r},t) \qquad [3.1b]$$

Equation 3.1a is known as Ficks first law, while equation 3.1b is Ficks second law. The diffusion coefficient D can be calculated by considering the diffusion process on a macroscopic statistical scale. The solution of Eq. 3.1b for molecules satisfying the initial condition $c(\boldsymbol{r},0) = \delta(\boldsymbol{r} - \boldsymbol{r}_0)$ is (69)

$$c(\boldsymbol{r},t) = (4\pi Dt)^{-3/2} \exp\left[-\frac{(\boldsymbol{r}-\boldsymbol{r}_0)^2}{4Dt}\right] \qquad [3.2]$$

Therefore, the probability of finding a molecule originally at a position $\boldsymbol{r}_0$, at a position $\boldsymbol{r}$, at time t, due to the diffusion process is also given by a Gaussian distribution (70). It follows that

$$\left\langle \left(\boldsymbol{r} - \boldsymbol{r}_0 \right)^2 \right\rangle = 6Dt \qquad [3.3]$$

Equation 3.3 is known as the Einstein relation and provides a description of diffusion more suitable for application in NMR than that given by Fick's laws (Eq. 3.1), D being described in terms of the mean squared diffusion distance, rather than gradient concentration and flux density.

3.2.2 The diffusion tensor

In the more general case, diffusion may not be identical in all directions and so D is more adequately characterised by a tensor, which describes molecular mobility along orthogonal x, y and z directions (diagonal elements) and also any coupling between these terms (off diagonal elements) (24). Diffusion has been shown to exhibit different rates in different directions in white matter (71-73) and in muscle (74), where diffusion along the direction of the fibres is greater than that perpendicular (75). Diffusion measurements can, therefore, be misleading in anisotropic media because the value obtained will depend on the direction of the diffusion measurement. Also, the values obtained from a series of diffusion measurements will be dependent upon the choice of the measurement coordinate system relative to the fibre or sample system. Therefore, to provide an adequate description that can reliably characterise anisotropic diffusion, independent of the coordinate system, diffusion must be treated as a tensor quantity.

In an anisotropic medium, the relationships given in §3.2.1, do not provide a complete description and must be extended (24,76). The diffusion tensor $\boldsymbol{D}$ is a macroscopic measure that relates the concentration gradient ∇c and the macroscopic diffusive flux $\boldsymbol{J}$ and is given by extending the condition in Eq. 3.1.

$$\boldsymbol{J} = -\boldsymbol{D}\nabla c$$

or

$$\begin{pmatrix} J_x \\ J_y \\ J_z \end{pmatrix} = -\begin{pmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{pmatrix} \begin{pmatrix} \dfrac{\partial c}{\partial x} \\ \dfrac{\partial c}{\partial y} \\ \dfrac{\partial c}{\partial z} \end{pmatrix} \qquad [3.4]$$

The tensor $\boldsymbol{D}$ must have real elements and is hermitian, thus $\boldsymbol{D}$ is also symmetric, for uncharged moieties such as water (24). This is a consequence of reciprocity theorem and the principle of microscopic reversibility of non-equilibrium thermodynamics (77,78). These theorems indicate that for uncharged moieties (such as water), an equal probability exists of the molecule moving in opposite directions. In contrast to the isotropic case given in Eq. 3.1, the anisotropic case (Eq. 3.4) illustrates how the diffusive flux may not necessarily be parallel to the concentration gradient (24).

3.3 Diffusion measurement using NMR

3.3.1 Description of diffusion with time dependent field gradients

The natural sensitivity of the NMR process to diffusion was highlighted in §2.4.3 and §2.4.4. This sensitivity arises because of the inhomogeneity in the static magnetic field $\boldsymbol{B}_0$, and static gradients caused by magnetic susceptibility differences within the sample. The random diffusive motion of the protons through these fields results in random phase accumulations in the *x-y* plane, which combine to interfere destructively, resulting in an attenuation of the NMR signal (14,17,79). Thus, self-diffusion of the sample causes an attenuation of the measured signal, superimposed onto the decay due to T_1 and T_2. This attenuation is usually negligible in conventional imaging sequences, except for some liquids of low viscosity which have a large T_2.

The vector representation of the Bloch equations was given in Eq. 2.22 and can be extended to include terms accounting for the effects of diffusion under the application of a constant magnetic field gradient $\boldsymbol{G}$. The resulting magnetisation evolution in the *x-y* plane was derived by Torrey (15), such that

$$\frac{dM_x}{dt} = \gamma B_{eff} M_y - \frac{M_x}{T_2} + D\nabla^2 M_x \qquad [3.5a]$$

$$\frac{dM_y}{dt} = \gamma B_{eff} M_x - \frac{M_y}{T_2} + D\nabla^2 M_y \qquad [3.5b]$$

where $B_{\text{eff}} = \boldsymbol{G}.\boldsymbol{r}$ and D is the diffusion coefficient. The solution to these equations for a standard 90° - τ - 180° spin echo sequence, with a constant applied gradient was given in Eq. 2.24, in terms of the signal at the echo time $A(2\tau)$, relative to the initial

signal after the 90° pulse $A(0)$. The general derivation for a time varying applied magnetic field gradient is presented below. Equation 3.5 can be rewritten to give

$$\frac{\partial \psi}{\partial t} = -i\gamma(\boldsymbol{r}.\boldsymbol{G})\psi + D\nabla^2\psi \qquad [3.6]$$

where $\psi(\boldsymbol{r}, t)$ is defined by

$$M_x + iM_y = \psi \exp\left[-\left(i\omega_0 + 1/T_2\right)t\right] \qquad [3.7]$$

and $M_x + iM_y$ represents the component of magnetisation in the rotating x-y plane (16). Ignoring the diffusion term in Eq. 3.6, the evolution of the magnetisation ψ in the absence of this term, assuming that $\psi = A$ immediately following the 90° pulse, is given by

$$\psi = A\exp\left(-i\gamma \boldsymbol{r}.\boldsymbol{F}\right) \qquad [3.8]$$

where

$$\boldsymbol{F}(t) = \int_0^t \boldsymbol{G}(t')dt' \qquad [3.9]$$

and represents the time integral of the magnetic field gradient (16). The 180° pulse then reverses the phase shift, i.e. sets the phase back by twice the amount which it has evolved, such that after the 180° pulse applied at time τ

$$\psi = A\exp\left[-i\gamma \boldsymbol{r}.\left(\boldsymbol{F} - 2\boldsymbol{f}\right) + i\phi\right] \qquad [3.10]$$

where $\boldsymbol{f} = \boldsymbol{F}(\tau)$. The phase angle ϕ depends on the phase of the 180° pulse relative to the 90° pulse and can be set to zero without loss of generality (16). The echo usually occurs at $t = \tau'$ such that $\boldsymbol{F}(\tau') = 2\boldsymbol{f}$, resulting in $\psi = A$ for all values of $\boldsymbol{r}$ as in the initial state after the 90° pulse. The echo would occur at $\tau' = 2\tau$ if a constant gradient was applied, but this is not necessarily the case for pulsed gradients (16). Rewriting Eqs. 3.8 & 3.10 gives

$$\psi = A\exp\left\{-i\gamma \boldsymbol{r}.\left[\boldsymbol{F} + \left(\xi - 1\right)\boldsymbol{f}\right]\right\} \qquad [3.11]$$

with $\xi = +1$ before the 180° pulse and $\xi = -1$ after the 180° pulse. Assuming a solution of the form in Eq. 3.11 and substituting into Eq. 3.6, assuming that A is a function of t only, gives

$$\frac{dA}{dt} = -\gamma^2 D\left[\boldsymbol{F} + \left(\xi - 1\right)\boldsymbol{f}\right]^2 A \qquad [3.12]$$

Subsequent integration of Eq. 3.12 gives the echo attenuation,

$$\ln\left[\frac{A(\tau')}{A(0)}\right] = -\gamma^2 D\left[\int_0^{\tau'} F^2 dt - 4\boldsymbol{f}.\int_{\tau}^{\tau'} \boldsymbol{F} dt + 4f^2(\tau' - \tau)\right] \qquad [3.13]$$

It is often convenient to introduce a diffusion attenuation *b*-value such that

$$\frac{A(\tau')}{A(0)} = \exp(-bD) \qquad [3.14a]$$

where

$$b = \gamma^2 \int_0^{\tau'}\left[\int_0^{t'} \boldsymbol{G}(t'')dt''\right]^2 dt' \qquad [3.14b]$$

and the integral in Eq. 3.13 has been simplified in Eq. 3.14b (69). The *b*-value is just determined from the pulse sequence gradient timing parameters and determines the degree of 'diffusion weighting'. The ratio $A(\tau')$ / $A(0)$ represents the signal attenuation due to diffusion, where $A(\tau')$ is the signal at the echo peak and $A(0)$ is the signal immediately after the 90° pulse.

3.3.2 Pulsed gradient spin-echo diffusion analysis

Equations 3.13 or 3.14 can be used to calculate the signal attenuation for the standard pulsed gradient spin-echo diffusion experiment, illustrated in Fig. 3.1, where additional diffusion gradient pulses of magnitude $\boldsymbol{G}$ are placed around the 180° RF pulse. The first pulse commences at time t_1, for length δ and the second commences at time $t_1 + \Delta$, again for length δ and the resulting signal attenuation is given by (16)

$$\ln\left[\frac{A(2\tau)}{A(0)}\right] = -\gamma^2 D\begin{bmatrix}\frac{2}{3}\tau^3 G_0^2 + \delta^2\left(\Delta - \frac{1}{3}\delta\right)G^2 \\ -\delta\left[\left(t_1^2 + t_2^2\right) + \delta\left(t_1 + t_2\right) + \frac{2}{3}\delta^2 - 2\tau^2\right]\boldsymbol{G}.\boldsymbol{G}_0\end{bmatrix} \qquad [3.15]$$

where $t_2 = 2\tau$ - $(t_1 + \Delta + \delta)$. It is assumed that a constant background gradient $\boldsymbol{G}_0$ is also present, which is necessary so that spins lose phase coherence between the diffusion gradient pulses. However, if it is assumed that this background gradient approaches zero then

$$\ln\left[\frac{A(2\tau)}{A(0)}\right] = -\gamma^2 D\delta^2 G^2\left(\Delta - \tfrac{1}{3}\delta\right) \qquad [3.16]$$

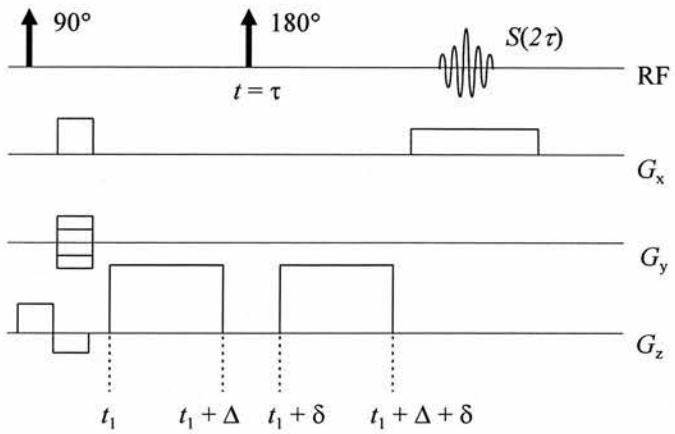


Figure 3.1 A typical pulsed-gradient spin-echo diffusion sequence incorporated with a spin-warp acquisition strategy. In this example the diffusion gradients are in the slice select (z) direction.

This is the well known Stejskal-Tanner diffusion equation (16), that provides a simple relationship between the degree of diffusion attenuation and the gradient timing parameters in the conventional spin-echo diffusion sequence.

The effect of diffusion on the NMR signal can be understood by considering the pulsed gradient experiment illustrated in Fig. 3.1. The gradient pulses magnetically label the spins to act as an endogenous tracer, replicating the radioactive tracer techniques conventionally used. The experiment can be described as follows. After the 90° pulse there is negligible loss of phase coherence until the first gradient pulse (as $\boldsymbol{G} >> \boldsymbol{G}_0$). This pulse causes an almost instantaneous phase shift $d(\phi_1)$, depending upon the position of the nucleus in the field at that time. To simplify matters, a gradient pulse along the z-direction is considered and the induced phase shift is given by

$$d(\phi_1) = \gamma \int_0^{\delta} Gzdt = \gamma G \delta z_1 \qquad [3.17]$$

where z_1 is the spin position, assumed to be constant during the short duration δ of the gradient pulse.

Between the gradient pulse and the 180° pulse, phase coherence changes are again negligible, but nuclei move around due to diffusion effects. The 180° pulse then inverts the phase shifts of the nuclei i.e. $d(\phi_1)$ is transformed into $-d(\phi_1)$ and the second diffusion gradient pulse produces phase shifts $d(\phi_2)$ identical to the first, given by

$$d(\phi_2) = \gamma \int_{\Delta}^{\Delta+\delta} Gz dt = \gamma G \delta z_2 \qquad [3.18]$$

where z_2 is the spin position during the second gradient pulse. The net dephasing $d\phi$ is thus, given by

$$d(\phi) = \gamma G \delta (z_1 - z_2) \qquad [3.19]$$

It can be seen that, for stationary spins, $z_1 = z_2$ and so there is no net phase shift induced. Thus, the second pulse should exactly undo the phase shift produced by the first and any incomplete refocusing is due to diffusion.

3.4 Diffusion tensor imaging

3.4.1 Diffusion description in an anisotropic medium

In §3.3, an analysis was presented for obtaining scalar diffusion measurements in an isotropic medium using magnetic resonance techniques. However, if the measured diffusion coefficient is directionally dependent, this simple analysis is no longer applicable and it is essential to define a macroscopic diffusion tensor as described in §3.2.2. The Bloch equations with diffusion terms (Eq. 3.5) must be modified to account for the directionality of $\boldsymbol{D}$, such that

$$\frac{\partial \boldsymbol{M}}{\partial t} = \gamma \boldsymbol{M} \wedge \boldsymbol{B} - \frac{\left(M_x \boldsymbol{i} + M_y \boldsymbol{j}\right)}{T_2} + \frac{\left(M_0 - M_z\right)\boldsymbol{k}}{T_1} - \nabla . \boldsymbol{v}\boldsymbol{M} + \nabla . \boldsymbol{D} . \nabla \boldsymbol{M} \qquad [3.20]$$

where $\boldsymbol{v}$ is the velocity of flowing spins (coherent motion) (76). The evolution of the spins in the x'-y' frame can be considered as before, and the resulting evolution is determined by

$$\frac{\partial \psi}{\partial t} = -i\gamma (\boldsymbol{r} . \boldsymbol{G})\psi - \nabla . \boldsymbol{v}\psi + \nabla . \boldsymbol{D} . \nabla \psi \qquad [3.21]$$

which can be compared to Eq. 3.6 for the scalar diffusion case (the flow term was omitted in Eq. 3.6). By analogy with the scalar diffusivity, the solution for the 90° - τ - 180° spin echo experiment, assuming no flow ($\boldsymbol{v} = 0$) is given by

$$\ln\left[\frac{A(\tau')}{A(0)}\right] = -\gamma^2 \int_0^{\tau'} \left[\boldsymbol{F}(t) - 2\xi(t)\boldsymbol{f}\right] \boldsymbol{D} \left[\boldsymbol{F}(t) - 2\xi(t)\boldsymbol{f}\right]^T dt \qquad [3.22]$$

where $\xi(t) = 0$ when $t < \tau' / 2$ and $\xi(t) = 1$ when $t > \tau' / 2$. $\boldsymbol{F}(t)$ is as defined in Eq. 3.9, $\boldsymbol{f} = \boldsymbol{F}(\tau)$ and the equation has been rewritten as shown by Basser et al. (24).

The effective diffusion tensor $\boldsymbol{D}$ can then be calculated as the mean value of the right hand side of Eq. 3.22 over the time interval $\{0, \tau'\}$. Using this definition in Eq. 3.22, leads to a simplified expression for the echo attenuation, such that

$$\ln\left[\frac{A(\tau')}{A(0)}\right] = -\sum_{i=1}^{3}\sum_{j=1}^{3} b_{ij} D_{ij} = -\boldsymbol{b}\cdot\boldsymbol{D} \qquad [3.23]$$

demonstrating the linear dependence between the logarithm of the signal attenuation, $\ln[A(\tau') / A(0)]$, and each element of the effective diffusion tensor. The b-matrix is a function of the pulse sequence gradient timing parameters, defined by

$$\boldsymbol{b} = \gamma^2 \int_0^{\tau'} [\boldsymbol{F}(t) - 2\xi(t)\boldsymbol{f}][\boldsymbol{F}(t) - 2\xi(t)\boldsymbol{f}]^T dt \qquad [3.24]$$

such that the b-matrix indicates the directional degree of diffusion weighting inherent in the sequence (24). The additional diffusion weighting resulting from imaging gradients is also accounted for, with interactions between diffusion and imaging gradients along the same direction being represented by the diagonal elements of $\boldsymbol{b}$ and those along orthogonal directions being represented by the off-diagonal elements of $\boldsymbol{b}$. For simple cases $\boldsymbol{b}$ can be calculated analytically from the integral in Eq. 3.24 (80-82), but for more complicated sequences a numerical approach to its evaluation is taken.

3.4.2 Estimation of the effective diffusion tensor

The linear relationship given in Eq. 3.23 allows for the measurement of different components of the $\boldsymbol{D}$ by applying diffusion gradients along various oblique directions. Application of a diffusion-weighted pulse sequence with a gradient component solely along x would allow for the measurement of $\boldsymbol{D}_{xx}$, while application of a gradient with all components of $\boldsymbol{G}$ being non-zero would allow for the echo attenuation to be observed resulting from all of the diagonal and off-diagonal tensor elements (24). The problem amounts to determining the six independent elements of $\boldsymbol{D}$ and $A(0)$, which can only be obtained by performing at least seven independent measurements of the echo intensity (83). In practice, due to errors in the measurement of the echo signal, many more trials are performed, such that $\boldsymbol{D}$ and $A(0)$ are estimated by statistical means.

Estimation of $\boldsymbol{D}$ and $A(0)$ from the measured echo signals is undertaken by using a multivariate linear regression analysis of Eq. 3.23. The optimal solution for $\boldsymbol{D}$ is found to be the one which minimises the sum of the squares of the differences between the observed and theoretical echo intensities (24). The tensor experiment consists of applying n non-collinear gradient directions at m different gradient strengths to measure $A(\boldsymbol{b})$, where $A(\boldsymbol{b})$ represents the signal dependence on the b-matrix. These observations of $A(\boldsymbol{b})$ are stored in an $nm \times 1$ column vector, $\boldsymbol{x}$. The seven parameters to be estimated, are stored in a 7×1 column vector, α, such that

$$\alpha^T = \left\{ D_{xx} \quad D_{yy} \quad D_{zz} \quad D_{xy} \quad D_{xz} \quad D_{yz} \quad \ln\left[A(0)\right] \right\} \qquad [3.25]$$

An $nm \times 7$ matrix of b matrix elements ($\boldsymbol{B}$) is calculated with the elements determined from Eq. 3.24 (either numerically or analytically), such that each row of $\boldsymbol{B}$ corresponds to the elements of the b-matrix pertaining to the i^{th} observation, as in Eq. 3.26, in an order corresponding to Eq. 3.25.

$$\boldsymbol{B} = \begin{bmatrix} b_{xx}^{i=1} & b_{yy}^{i=1} & b_{zz}^{i=1} & b_{xy}^{i=1} & b_{xz}^{i=1} & b_{yz}^{i=1} & 1 \\ b_{xx}^{i=2} & b_{yy}^{i=2} & b_{zz}^{i=2} & b_{xy}^{i=2} & b_{xz}^{i=2} & b_{yz}^{i=2} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{xx}^{i=nm} & b_{yy}^{i=nm} & b_{zz}^{i=nm} & b_{xy}^{i=nm} & b_{xz}^{i=nm} & b_{yz}^{i=nm} & 1 \end{bmatrix} \qquad [3.26]$$

An $nm \times 1$ column vector of predicted outcomes, ξ, is then defined as the product of $\boldsymbol{B}$ and α,

$$\xi = \boldsymbol{B}\alpha \qquad [3.27]$$

The χ^2 parameter is defined as

$$\chi^2(\xi) = (\boldsymbol{x} - \xi)^T \Sigma^{-1} (\boldsymbol{x} - \xi) \qquad [3.28]$$

which represents the weighted sum of the squares of the deviations between the observed and the predicted echo intensities. The term involving the $nm \times nm$ matrix, Σ, arises as a weighting factor, where the diagonal elements of Σ represent the corrected reciprocal error variance, x_i^2 / σ_{ii}^2 for the i^{th} observation. These terms correct for the expected variation in each trial and account for errors associated with taking the logarithm of the echo attenuation in Eq. 3.23. In practice, this weighting factor requires the calculation of the pixel intensity (x_i) in each voxel, divided by the root-mean-squared background noise (σ_{ii}) in the corresponding image.

The χ^2 minimisation procedure for ξ results in seven linear equations, which are given by

$$\left(\boldsymbol{B}^T\Sigma^{-1}\boldsymbol{B}\right)\alpha_{opt} = \left(\boldsymbol{B}^T\Sigma^{-1}\right)\boldsymbol{x} \qquad [3.29]$$

resulting in the optimal parameters α_{opt} being

$$\alpha_{opt} = \left(\boldsymbol{B}^T\Sigma^{-1}\boldsymbol{B}\right)^{-1}\left(\boldsymbol{B}^T\Sigma^{-1}\right)\boldsymbol{x} = \boldsymbol{M}^{-1}\left(\boldsymbol{B}^T\Sigma^{-1}\right)\boldsymbol{x} \qquad [3.30]$$

The expression $\boldsymbol{M}^{-1} = \left(\boldsymbol{B}^T\Sigma^{-1}\boldsymbol{B}\right)^{-1}$ is a 7 × 7 matrix whose diagonal elements represent the error variances in each of the seven estimated parameters of α. So, Eq. 3.30 allows for the estimation of the diffusion tensor elements and ln[$A(0)$], along with their associated errors for each pixel in an image (24).

3.4.3 Principal axes and the diffusion ellipsoid representation

Once the diffusion tensor $\boldsymbol{D}$ has been calculated, it is possible to assign a principal coordinate system in which the diffusive fluxes and concentration gradient are decoupled (25). In this coordinate system all of the off-diagonal tensor elements are zero and the three diagonal diffusion coefficients represent the three principal diffusivities. The eigenvectors ε_1, ε_2 and ε_3 of the tensor are orthogonal, because $\boldsymbol{D}$ is symmetric and positive definite. The eigenvectors represent the principal diffusion directions, while the corresponding eigenvalues, λ_1, λ_2 and λ_3, represent the principal diffusivities. The relationship between the principal coordinate system and the measurement system in which $\boldsymbol{D}$ is obtained is (84,85)

$$\Lambda = \boldsymbol{E}^T\boldsymbol{D}\boldsymbol{E} = \boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T \qquad [3.31]$$

where

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \qquad [3.32]$$

$$\boldsymbol{E} = \begin{pmatrix} \varepsilon_1 & \varepsilon_2 & \varepsilon_3 \end{pmatrix} \qquad [3.33]$$

and

$$\boldsymbol{R} = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix}\begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad [3.34]$$

The angles ϕ, θ and ψ are the three Euler angles that rotate the measurement coordinate system (x , y , z) into alignment with the principal diffusion system (x''', y''', z''') as illustrated in Fig. 3.2 (85). The coordinate systems (x', y', z') and (x'', y'', z'') represent the axes after the first (ϕ) and second (θ) Euler rotations, respectively. Physically, the principal diffusion directions correspond to the orthotropic directions of the tissue within the voxel, where contributions from microscopic restrictions add to give a macroscopic voxel averaged orientation. The principal diffusivities then indicate the magnitude of restriction along the corresponding principal direction.

Diffusion ellipsoids provide a convenient graphical description of the diffusion tensor, where the distance from the origin to the surface of the ellipsoid represents a scaled measurement of the diffusivity in that direction (86). To understand the physical interpretation of the diffusion ellipsoid, the molecular motion is considered to be a random Brownian process characterised by a macroscopic Gaussian conditional probability density function $\rho(\boldsymbol{r}|\boldsymbol{r}_0, t)$, given by

$$\rho\left(\boldsymbol{r}|\boldsymbol{r}_0,\tau\right)=\frac{1}{\sqrt{|\boldsymbol{D}(\tau)|(4\pi\tau)^3}}\exp\left[\frac{-\left(\boldsymbol{r}-\boldsymbol{r}_0\right)^T\boldsymbol{D}(\tau)^{-1}\left(\boldsymbol{r}-\boldsymbol{r}_0\right)}{4\tau}\right] \qquad [3.35]$$

which represents the probability that a proton initially at $\boldsymbol{r}_0$ at time $t = 0$, reaches position $\boldsymbol{r}$ at time $t = \tau$ (25,86).

Surfaces of constant probability can then be constructed by setting the exponent in Eq. 3.35 to a constant. A particularly convenient choice is ½ so that

$$\frac{\left(\boldsymbol{r}-\boldsymbol{r}_0\right)^T\boldsymbol{D}(\tau)^{-1}\left(\boldsymbol{r}-\boldsymbol{r}_0\right)}{2\tau}=1 \qquad [3.36]$$

For a microscopically homogenous anisotropic medium, Eq. 3.36 describes a surface of constant mean translational displacement at time $t = \tau$. This can be observed by transforming Eq. 3.36 into the principal coordinate system, using $\boldsymbol{r}''' = \boldsymbol{E}^{\mathrm{T}}(\boldsymbol{r} - \boldsymbol{r}_0)$ to give

$$\frac{\boldsymbol{r}'''^{T}\Lambda^{-1}\boldsymbol{r}'''}{2\tau}=1 \qquad [3.37]$$

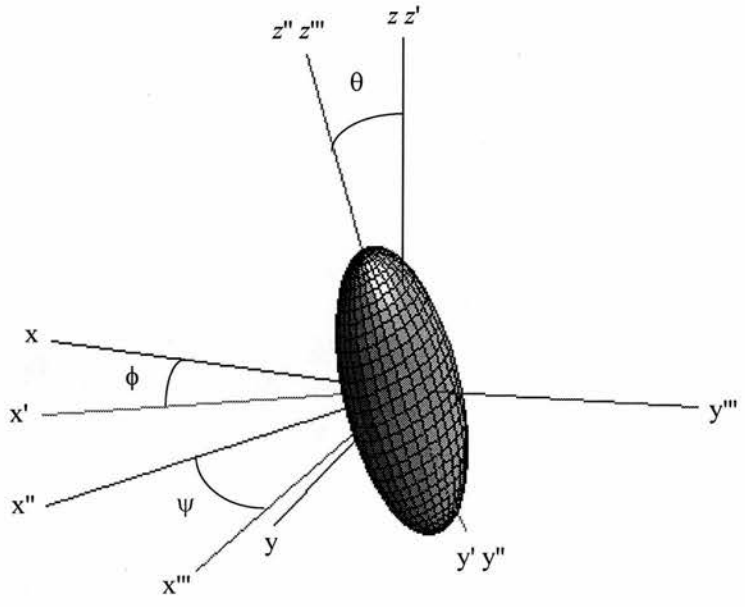


Figure 3.2 Illustration of a diffusion ellipsoid with principal diffusivities in the ratio 2:1:4 rotated through Euler angles $\phi = 30°$, $\theta = 15°$ and $\psi = 45°$. The measurement axes are illustrated by the (x, y, z) coordinate system and the principal diffusion system is represented by (x''', y''', z''').

which on expansion gives

$$\left(\frac{x'''}{\sqrt{2\lambda_1\tau}}\right)^2 + \left(\frac{y'''}{\sqrt{2\lambda_2\tau}}\right)^2 + \left(\frac{z'''}{\sqrt{2\lambda_3\tau}}\right)^2 = 1 \qquad [3.38]$$

Thus, in the principal coordinate system, the major axes of the diffusion ellipsoid are the root mean squared diffusion distances $\sqrt{\langle x_i'''^2 \rangle} = \sqrt{2\lambda_i\tau}$ in the three orthotropic principal directions at time τ (25). The factor $\sqrt{2}$ arises in the diffusion distance term due to the one-dimensional form of the expressions (as the individual x, y and z components are considered). In Eq. 3.3, the factor $\sqrt{6}$ arises due to the three dimensional nature of this expression.

It should be noted that $\boldsymbol{D}(\tau)$ is generally a function of the time τ, because the degree of restriction encountered by diffusing protons will depend on the diffusion time and so the duration of the measurement period (87). For example, in living tissues, if the experimental diffusion time (Δ in Eq. 3.16) is very short then few protons will have encountered a permeable barrier, resulting in the observation of isotropic diffusion. However, as the diffusion time is increased, barriers will be encountered (87,88) and the underlying anisotropic structure will be observed as the

ellipsoids become more prolate. Inaccurate measurements may be obtained if membranes are impermeable because the Gaussian distribution assumed in Eq. 3.35 may not adequately describe the displacement distribution (64).

3.5 MR diffusion pulse sequences

3.5.1 Navigated spin-echo

The conventional spin-echo experiment and spin-warp acquisition procedure described in §2.4.4 and §2.6.2 can be easily sensitised to diffusion effects by the addition of two large gradient pulses around the 180° RF refocusing pulse, as described in §3.3.2 and illustrated in Fig. 3.1 (16). However, the addition of these gradients makes the sequence highly sensitive to motion and even sub-millimetre displacements during the experiment will result in large phase changes in the resultant echo signal. These phase changes can vary between phase encoding steps (view-to-view changes) and also within a phase encoding step (intraview changes) (89). The phase changes due to motion can be monitored by the addition of a navigator echo into the imaging sequence, that provides a reference phase which enables correction of the phase perturbations in the image echo (90-93). Navigator correction algorithms have been developed to shift the apparent distribution of spins in each successive spin-echo, compensating for view-to-view motion and applying a phase rotation that is opposite to the observed phase perturbation due to intraview motion (91).

The spin-echo diffusion sequence is modified as shown in Fig. 3.3, such that two 180° RF refocusing pulses follow the initial 90° pulse, resulting in the formation of two spin-echo signals. One of the echo signals is phase encoded as usual, the image echo, and the other is the non-phase encoded navigator echo. Ordidge et al. (90) developed a navigator correction scheme that assumes linear, bulk motion of the whole sample, which results in a phase shift proportional to the velocity of the motion. Anderson and Gore (91) further developed this technique, such that the phase shifts were obtained for rotational, as well as translational motion. This technique allows for correction of all rigid body motions and the resulting image

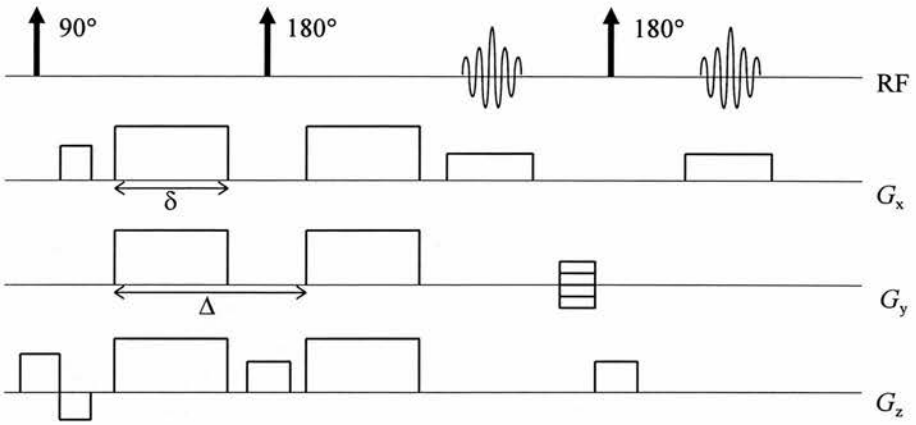


Figure 3.3 The navigated spin-echo diffusion sequence, utilising a non-phase encoded navigator echo followed by a conventional phase encoded image echo. The navigator echo is used to provide phase correction of the image echo correcting for phase distortions resulting from patient motion.

echo signal is of the form

$$S_{img}\left(k_x, k_y\right) = \exp\left(-i\phi\right)\hat{S}_{img}\left(k_x - dk_x, k_y\right) \qquad [3.39]$$

where S_{img} is the corrected image echo and $\hat{S}_{img}$ is the motion corrupted image echo. So, to correct for the effects of motion, the navigator echo needs to sample the phase shift ϕ due to translational motion (90) and the term dk_x due to rotation (91). Equation 3.39 has been simplified by assuming that through slice phase errors, $\pm dk_z z$, are negligible as the slice thickness is generally much less than the FOV. Also, it is assumed that $dk_y = 0$, which results from the diffusion gradients being applied along the phase encoding (y) direction or from the rotation angles about the x and z axes, θ_x and θ_z being negligible (91). If rotations are significant and the diffusion gradient is applied along the frequency encoding (x) or slice select (z) axes then the phase error cannot be predicted using the navigator echo and so image ghosting will result after Fourier transformation.

The zero-order correction algorithm was developed by Ordidge et al. producing correction of the time domain data (90). A net vector was calculated from the first arbitrary reference echo, summed over the central complex data points around the navigator echo maximum and then compared to vectors generated from subsequent navigator echo signals. The phase angles between each vector and the

reference vector were calculated and used to correct all of the data points in the respective phase encoding step of the image echo by adding the phase difference to each point.

The method was extended to account for rotational motion by Anderson and Gore (91), where a shift of the *k*-space data or equivalently a multiplication of the Fourier transform by a linear phase ramp is required. As such, it is convenient to carry out the correction using projections obtained by Fourier transformation of the data along the frequency encoding direction. Expressions for the navigator and corresponding image projections (p_n and p_i) are shown in Eqs. 3.40 and 3.41, respectively

$$p_n(x;0) = e^{i\phi} e^{idk_x x}\left[\int e^{idk_z z} dz\right] \int \rho(x,y) e^{i(dk_y)y} dy \qquad [3.40]$$

$$p_i(x;k_y) = e^{-i\phi} e^{-idk_x x}\left[\int e^{-idk_z z} dz\right] \int \rho(x,y) e^{i(k_y - dk_y)y} dy \qquad [3.41]$$

where $\rho(x,y)$ is proportional to the transverse magnetisation (91). The first factor involving ϕ is the zero order phase shift due to translational motion. Subsequent terms involve rotations which result in *k*-space shifts, dk_x, dk_y and dk_z along the frequency encoding, phase encoding and slice select directions, respectively. It can be seen that the phase shifts due to rotation have a first order linear dependence with position. The first two terms involving ϕ and dk_x differ only by the sign of the phase error between the projections p_n and p_i, with the dk_x term varying linearly with x, and so the phase error due to these terms can be corrected using the navigator echo information. The third term can be ignored because the phase error resulting from $dk_z z$ is assumed to be negligible for typical slice thickness to FOV ratios. The phase error induced by the final term is only accounted for when $dk_y = 0$, as the form of $\rho(x,y)$ is not known. This only occurs when either the diffusion gradient is applied only along y, or the rotation around the x and z axes, θ_x and θ_z is negligible (34).

The first stage of a practical implementation of the navigator reconstruction scheme involves Fourier transformation along the frequency encoding direction of both the navigator and image *k*-space data sets to obtain the projection data. A masking filter is usually applied to the data, such that only data points with a significant signal are included in the calculations to eliminate erroneous phase errors

resulting from noisy data. A point-by-point phase comparison is then made between samples along the first projection (chosen as an arbitrary reference) and subsequent projections. A modified strategy suggested by de Crespigny et al. (92) was to integrate the magnitude along each of the projections and to take the highest value as the reference projection, as it is assumed that artefacts generally result in a signal reduction. The phase difference between the n^{th} samples of the reference and m^{th} projections in the navigator data is then added to the phase of the n^{th} sample of the m^{th} projection in the imaging data. Fourier transformation of the image data along the phase encoding direction can then be performed after all the phase corrections have been undertaken. If the phase errors are assumed to be linear along the projection as is indicated by Eqs. 3.40 & 3.41, then a first order polynomial could be fitted to the phase errors along the projection. However, this approach was shown to be less effective than evaluating the phase errors on a point-by-point basis, which effectively accounts for non-rigid body motion (94).

3.5.2 Echo planar

Both the gradient and spin-echo forms of the EPI acquisition scheme, illustrated in Figs. 2.15a & 2.15b, are easily modified to produce a diffusion sensitised imaging sequence (72). In the spin-echo case, a pair of unipolar gradient pulses is applied around the 180° RF pulse as described in §3.3.2 and for the gradient echo case a pair of bipolar gradient pulses is placed before the EPI readout train. EPI is particularly suited to diffusion imaging applications because of the single-shot nature of the acquisition and its inherent insensitivity to motion artefacts. The same problems result from EPI diffusion sequences as were described in §2.6.3, particularly regarding susceptibility related artefacts. Also, the whole image must be acquired in a time of around 100ms so that the signal has not decayed too far, due to T_2 and T_2^* relaxation. This means not only that rapid switching of the EPI readout gradients is required, but also that the diffusion gradients must be of relatively short duration δ. This implies that relatively high gradient strengths are needed if sufficient signal attenuation due to diffusion is to be observed (Eq. 3.16).

A further problem resulting from the application of diffusion gradients is the introduction of additional eddy currents due to the steep rise and fall of these strong gradients (95-99). Long time constant eddy currents can be introduced that overlap the EPI readout train, resulting in distortion of the phase evolution and hence the resulting image. Residual eddy currents result in geometric image distortions, taking the form of a translation if the residual gradient is along the slice select direction, a shear if the gradient is along the frequency encode direction and a magnification if the residual gradient is along the phase encode direction (97). These distortions largely occur along the phase encoding direction due to the low bandwidth in that direction. Problems arise when images are combined to calculate quantitative diffusion maps because the images are geometrically different, depending on the direction and magnitude of the applied diffusion gradient. The resulting images show a lack of sharpness due to blurring at the interfaces between different structures, edge artefacts are observed and an erroneous increase in the measured diffusion anisotropy may result, likely to be particularly apparent when measuring small structures (100).

Several methods have been developed to overcome the eddy current induced distortions (95-100). The form of the diffusion gradients can be optimised to minimise eddy currents, particularly by the use of bipolar gradients where the opposite induced eddy currents cancel one another (96,98). However, this technique results in a reduced diffusion sensitivity with the effective diffusion weighting being 25% of the standard unipolar experiment. Another technique involves mapping the eddy current fields by the sampling of a one-dimensional field map, allowing for subsequent correction of the *k*-space data (99). The addition of the field mapping strategy into the pulse sequence extends the imaging time of the experiment by a factor of three, which could make DTI experiments unacceptably long for a clinical setting.

Post-processing techniques have also been developed to correct for the distortions retrospectively (97,100). Haselgrove and Moore (97) suggested a technique based on iterative cross-correlation of the distorted diffusion-weighted images relative to the baseline image which is assumed to be undistorted. The cross-correlation method requires that the baseline and diffusion images have similar signal

intensity distributions and so is only effective for correcting sets of images with similar *b*-values, where the diffusion induced contrast has not altered the appearance of the image greatly. Direct comparison of the images would also be possible if an inversion recovery CSF suppressed sequence was used (101,102), such that the baseline and diffusion-weighted images demonstrate similar contrast. However, the distortion parameters calculated for the low *b*-value images can be extrapolated to correct for higher *b*-value images, but this is a computationally intensive process and may be inaccurate for correcting images with a high degree of diffusion-weighting. Bastin (100) demonstrated that it was possible to calculate the distortion parameters for each slice position in the magnet bore by performing a separate water phantom calibration, and then using this data to correct for the distortions on the patient data, thereby reducing the post-processing time considerably.

3.5.3 Other diffusion sequences

A further type of echo formation has been used to provide images with diffusion weighting involving the use of the 90° - τ_1 - 90° - τ_2 - 90° stimulated-echo pulse sequence (103,104), illustrated in Fig. 3.4. The first 90° RF pulse rotates the magnetisation into the transverse *x-y* plane where the spins lose phase coherence due to T_2^* effects (103). The second 90° pulse is applied after τ_1 and stores the magnetisation longitudinally, where it undergoes relaxation due to T_1. The third 90° pulse then rotates the magnetisation back into the transverse plane, but with reversed sign, and the spins rephase and converge to form an echo. The advantage of the stimulated echo sequence is that the storage period τ_2 (sometimes called the mixing time T_m) decays with T_1 and so a longer pulse separation Δ can often be used, providing that $T_1 >> T_2$. The advantage of using a longer Δ is that it allows for a higher degree of diffusion weighting, without the loss of signal due to T_2 and also it allows time for eddy currents from the first diffusion gradient pulse to decay. However, the echo signal is reduced by a factor of two in the stimulated-echo sequence as only components of magnetisation perpendicular to the phase of the second 90° pulse are transferred into the longitudinal plane (103). Thus, if the echo

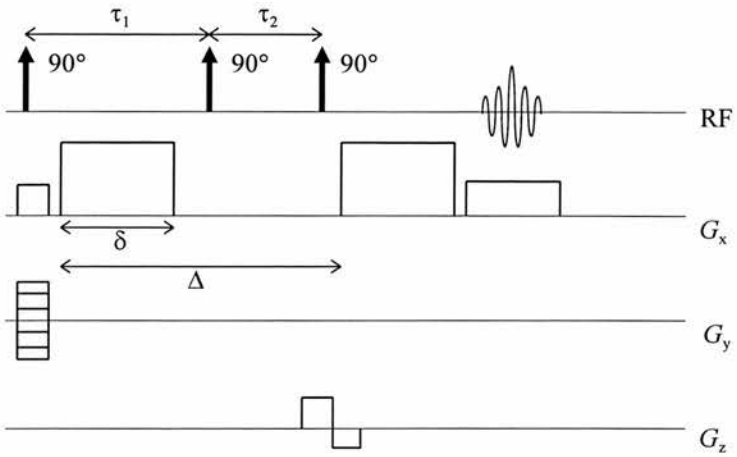


Figure 3.4 The 90° - τ_1 - 90° - τ_2 - 90° pulsed gradient stimulated-echo (PGSTE) diffusion-weighted pulse sequence, with diffusion gradients applied along the frequency encoding (x) direction.

signal is initially weak then the ratio T_1 / T_2 must be high for the stimulated-echo sequence to be advantageous.

Another technique suggested for measuring diffusion is to use the field produced by the RF coil, the B_1 field (105,106). This technique is useful to overcome hardware limitations resulting from the use of high B_0 fields. A surface coil is oriented perpendicular to the main transmit / receive coil to produce an RF gradient pulse $G_1 = dB_1 / dr$ of duration δ that flips the magnetisation within the sample by an angle θ, which depends on position r such that

$$\theta = \gamma\delta\left[\frac{dB_1}{dr}\right]r \qquad [3.42]$$

The effect of this flip angle is then cancelled after a time interval Δ by application of a symmetric RF pulse. This procedure acts such that stationary spins will be completely refocused, but any moving spins will not be refocused resulting in a loss of longitudinal magnetisation that can be probed using an 'observe' pulse or an imaging sequence. The signal loss depends on the diffusion coefficient and the longitudinal relaxation T_1 (105). The use of RF gradients means that extremely short switching times can be used due to a lack of induced eddy currents. Large gradient strengths can also be used, depending on the output capabilities of the RF transmitter, which could allow for the measurement of very small diffusivities. However, the use of these long and strong RF pulses is likely to result in specific absorption rate (SAR)

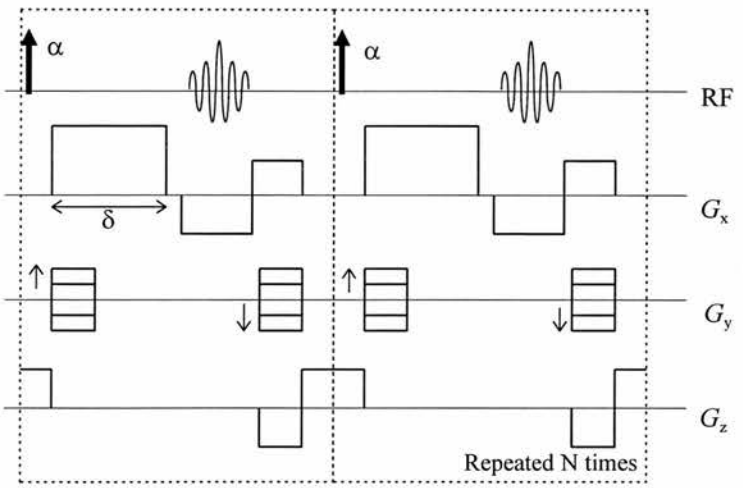


Figure 3.5 Illustration of the steady-state diffusion sequence, with diffusion gradients applied in the frequency encoding direction.

limits being exceeded, precluding its use in the clinical setting. The SAR is the limit imposed on patient absorption of RF radiation and is measured in watts per kilogram.

Diffusion sequences have been devised that are based on the steady-state principles involved when a repeating series of α-pulses are applied (107-109), usually of 90° flip-angle, as illustrated in Fig. 3.5 (see also §2.4.2). In the steady state, the resulting NMR signal is composed of a free induction decay after each pulse and an echo signal formed before each pulse, with the decay being T_1 dependent. For the steady state to be reached, the phase-encoding gradient must be completely refocused as illustrated in Fig. 3.5. The echo signal is found to be a complicated mixture of low and high order spin and stimulated-echoes and so calculation of diffusion coefficients from the measured signal is not a simple matter (107). A multi-exponential approximation to the form of the signal has been made, but tends to result in the overestimation of diffusion coefficients as it does not consider contributions from lower order echoes. The sequence does have the advantage of being relatively fast, with an image being typically acquired in around 15 seconds.

Several acquisition schemes have been developed as alternatives to the standard spin-warp or echo planar methods. Most of these techniques involve the application of low flip angle RF pulses, providing fast acquisition but usually resulting in poor SNR. The high speed stimulated-echo acquisition mode (STEAM)

sequence involves a stimulated echo diffusion-weighted preparation period followed by a series of α-pulses which are repeated *n*-times, where *n* is determined by the number of phase encoding steps required (110,111). Thus, the sequence has the form 90° - τ_1 - 90° - τ_2 - α_n. Each phase encoding step has a slightly different diffusion weighting and the effective diffusion weighting of the sequence is taken at the zeroth phase encoding step as this contributes most to the image contrast (110). Single-shot STEAM sequences provide a fast acquisition method with a low sensitivity to susceptibility induced artefacts, but have poor SNR. A further adaptation of this method is to use a stimulated-echo diffusion preparation period followed by an acquisition strategy that consists of a combination of readout gradient reversals and 180° RF pulses. This provides a mixture of spin and gradient echoes that are optimised to increase the SNR for a given readout time. This method is known as GRASE and results in a fast imaging strategy that has less of a T_2^* or susceptibility dependence than EPI (112). Fast spin-echo (FSE) or fast low angle shot (FLASH) sequences are also based on a low flip angle acquisition scheme (113,114). The preparation period is a 90° - τ - 180° spin-echo sequence, followed by a further 90° pulse to store the magnetisation longitudinally. This magnetisation is then read out using a repeating series of *n* α-pulses along with a reversal of the readout gradient to provide a gradient echo.

3.6 Quantitative diffusion measurements

3.6.1 Measurements of diffusivity

The diffusivity *D* can be calculated in any given direction from Eq. 3.14, or the whole diffusion tensor $\boldsymbol{D}$ can be evaluated by sampling in at least seven independent directions as given by Eq. 3.23. Clearly, the measured diffusivity will be dependent on the direction of the applied diffusion gradient, if the media is anisotropic. Therefore, it is desirable to obtain an average measure of the diffusivity that is orientationally independent (115). The trace of the diffusion tensor $tr(\boldsymbol{D})$ is just such a measure and is calculated from the diffusion tensor using Eq. 3.43.

$$tr(\boldsymbol{D}) = D_{xx} + D_{yy} + D_{zz} = \lambda_1 + \lambda_2 + \lambda_3 \qquad [3.43]$$

However, values are often quoted for an average diffusivity D_{av} which is given by

$$D_{av} = \bar{\lambda} = tr(\boldsymbol{D}) / 3 \approx ADC_{av} \qquad [3.44]$$

where ADC_{av} represents the directionally averaged apparent diffusion coefficient, measured in vivo. The diffusivity is described as the apparent diffusion coefficient, ADC, because of the contributing factors to the measurement of diffusion in vivo, where a small proportion of the measured diffusivity may result from other IVIM contributions due to perfusion or incoherent flow effects (19,116). ADC_{av} is the average of three ADCs measured along orthogonal directions and provides a good approximation to D_{av} when the imaging gradient 'cross-terms' (117) have been accounted for in the calculation of the *b*-values (i.e. off-diagonal elements of the *b*-matrix are negligible) (81,82). This can be easily seen by expanding Eq. 3.23 and comparing to Eq. 3.14. It is usual not to use the term ADC to describe diffusivities calculated with DTI experiments, although these measurements may still include contributions from other IVIM sources.

Diffusivities reported in the literature show a large amount of variation between research groups and there are many reasons for these systematic variations. The measured diffusivity is highly dependent on the temperature of the spin system (3), this relationship is well known for water and is given by

$$D = D_0 \exp\left(- E_a / kT\right) \qquad [3.45]$$

where k is the Boltzmann constant and E_a is the activation energy (67,70). This expression results in a change in diffusivity of around 2.4% per degree Kelvin temperature change. The sensitivity of diffusion to temperature has been exploited to monitor temperature in induced hypothermia (118), in laser treatments (119,120) and to study the effects of thermal convection (121).

The measured diffusivity, in vivo, is also dependent on the diffusion time used in the experiment (87,122). The distance moved by the diffusing molecules during the experiment is given by Eq. 3.3 and is determined by the experimental diffusion time, defined by Eq. 3.16 such that $t_d = (\Delta - \delta / 3)$. For very low diffusion times, water appears to diffuse isotropically, as the mean free path is small enough such that few molecules encounter a cell membrane (71,87). For intermediate diffusion times, a local anisotropy results as a large number of molecules encounter cell membranes and preferential diffusion occurs around the cells. For longer

diffusion times, diffusion may again appear isotropic as the mean free path becomes much greater than the cell dimensions and the water then has equal probability of diffusing either through or around the cells (87). It is believed that hindered diffusion is likely to be observed in typical cellular environments for diffusion times of around 30 ms. However, it should be noted that the diffusion time is only well defined if $\Delta >> \delta / 3$, and has little physical meaning otherwise (69).

The mono-exponential relationship assumed in Eq. 3.14 between the signal attenuation and the *b*-value has also been demonstrated to be an over-simplification (88,123,124). Restricted diffusion effectively creates separate compartments between the intracellular and extracellular water, with the intracellular compartment exhibiting a lower diffusion coefficient (122). Experiments performed at high *b*-values (up to 1×10^4 smm^{-2}) have demonstrated that this results in a bi-exponential dependence of the signal attenuation with the *b*-value (88). In effect, by assuming a mono-exponential fit at typical *b*-values used for in vivo studies (<1000 smm^{-2}), only the fast part of this curve is being sampled which corresponds to the extracellular water. This results in an ADC measurement that is biased towards the extracellular diffusivity. A similar bi-exponential dependence results from voxels which contain more than one type of diffusing motion, such as CSF and grey matter (124). If a mono-exponential fit is assumed, then the measured diffusivity will be an average of the constituent diffusivities in the voxel. It may be desirable to separate out the different components, when attempting to measure diffusion in brain tissues, particularly to eliminate the partial volume contamination resulting from the high diffusivity of CSF. An alternative to reduce this problem is to use a CSF suppressed inversion recovery sequence as described in §2.4.2 (101,102).

3.6.2 Diffusion anisotropy

The previous section dealt with quantitative measurements of the magnitude of diffusion, particularly regarding the production of a rotationally invariant average measure. It is intuitive to think of $tr(\boldsymbol{D})$ or D_{av} as providing a measure of the average 'size' of the diffusion ellipsoid. However, $tr(\boldsymbol{D})$ provides no indication as to the shape or form of the diffusion ellipsoid. Diffusion anisotropy indices (DAIs) have,

therefore, been defined to provide a quantitative indication as to the shape of the diffusion ellipsoid or the form of the cellular and fibre restrictions encountered (21,85,115). Most DAIs are scaled such that a value of zero represents isotropic diffusion and a value of one represents the case of an infinitely long, thin fibre (highly anisotropic diffusion). The indices can then be displayed as a grey scale image, whereby isotropic structures such as free water or CSF appear black and anisotropic structures, such as highly ordered white matter tracts appear white. The idea behind producing a scalar measurement of diffusion anisotropy is that it behaves like a quantitative histological or physiological stain, enabling the evaluation of diffusion anisotropy and fibre-tract organisation (84).

The first DAIs were based on the difference between two orthogonal diffusion measurements, usually expressed in the form of a ratio (125). A more recent variation uses the information obtained from three orthogonal diffusion measurements to provide a scaled anisotropy measurement and is known as the standard deviation anisotropy (A_{sd}) given by (21)

$$A_{sd} = \frac{1}{\sqrt{6}ADC_{av}} \sum_{j=x,y,z} \sqrt{\left(ADC_j - ADC_{av}\right)^2} \qquad [3.46]$$

These indices provide a qualitative analysis of the diffusion anisotropy as they do not use the full content of the diffusion tensor and as such are rotationally variant (115). Only when the principal diffusion and measurement coordinate systems coincide will a reliable assessment be made as to the degree of anisotropy. Nevertheless, these indices have indicated qualitative changes of anisotropy in several studies (21,126), although the degree of error resulting from the rotational variance has rarely been considered (127).

To overcome the deficiencies of anisotropy indices based on two or three orthogonal diffusion measurements and enable the production of rotationally invariant anisotropy measurements, diffusion tensor imaging must be used (24,25). Several scalar parameters can be defined from the diffusion tensor to provide measures of distinct, intrinsic, microstructural features of the diffusion environment (84). Various tensor operations can be performed on $\boldsymbol{D}$, to produce meaningful scalar invariant quantities. Firstly, the tensor can be decomposed into isotropic and anisotropic tensors, where the isotropic tensor is given by the identity tensor

multiplied by the scalar mean diffusivity, D_{av}. The anisotropic part of $\boldsymbol{D}$ is known as the deviatoric $\boldsymbol{D}_{dev}$ and indicates the deviation of $\boldsymbol{D}$ from being an isotropic tensor, as indicated below

$$\boldsymbol{D} = D_{av}\boldsymbol{I} + \left(\boldsymbol{D} - D_{av}\boldsymbol{I}\right) = D_{av}\boldsymbol{I} - \boldsymbol{D}_{dev} \qquad [3.47]$$

The scalar magnitudes of $\boldsymbol{D}$, the isotropic part of $\boldsymbol{D}$ and the deviatoric $\boldsymbol{D}_{dev}$ can be calculated from their tensor dot products (cf. the vector dot product), such that

$$\sqrt{\boldsymbol{D}\cdot\boldsymbol{D}} = \sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2} \qquad [3.48]$$

$$\sqrt{D_{av}\boldsymbol{I}\cdot D_{av}\boldsymbol{I}} = D_{av}\sqrt{\boldsymbol{I}\cdot\boldsymbol{I}} = D_{av}\sqrt{3} \qquad [3.49]$$

$$\begin{aligned}\sqrt{\boldsymbol{D}_{dev}\cdot\boldsymbol{D}_{dev}} &= \sum_{j=1}^{3}\sqrt{\left(\lambda_j - D_{av}\right)^2} \\ &= \sum_{j=x,y,z}\sqrt{\left(D_{jj} - D_{av}\right)^2} - \sqrt{2D_{xy}^2 + 2D_{xz}^2 + 2D_{yz}^2}\end{aligned} \qquad [3.50]$$

These equations enable scalar anisotropy indices to be produced by comparing the magnitudes of the isotropic and anisotropic components of $\boldsymbol{D}$ (84). A particularly logical choice is to compare the ratio of the magnitude of the anisotropic component to the isotropic component, to obtain a measure of the relative anisotropy (A_{ra}) such that

$$A_{ra} = \frac{1}{\sqrt{3}}\frac{\sqrt{\boldsymbol{D}_{dev}\cdot\boldsymbol{D}_{dev}}}{D_{av}} = \frac{1}{\sqrt{3}D_{av}}\sum_{j=1}^{3}\sqrt{\left(\lambda_j - D_{av}\right)^2} \qquad [3.51]$$

The similarity between A_{ra} and the rotationally variant index A_{sd} (Eq. 3.46) becomes immediately apparent, illustrating how A_{sd} becomes equivalent to A_{ra} when the principal diffusion and measurement systems are aligned i.e. $ADC_x = \lambda_1$, $ADC_y = \lambda_2$ and $ADC_z = \lambda_3$. In fact, the index $A_{ra} / \sqrt{2}$ is often used as the rotationally invariant equivalent of A_{sd} as this is scaled between the conventional limits of zero and one (85). A further anisotropy index suggested is the fractional anisotropy (A_{fa}) which gives a measure of the fraction of the magnitude of $\boldsymbol{D}$ that can be ascribed to anisotropic diffusion and is given by

$$A_{fa} = \sqrt{\frac{3}{2}}\frac{\sqrt{\boldsymbol{D}_{dev}\cdot\boldsymbol{D}_{dev}}}{\sqrt{\boldsymbol{D}\cdot\boldsymbol{D}}} = \sqrt{\frac{3}{2\left(\lambda_1^2 + \lambda_2^2 + \lambda_3^2\right)}}\sum_{j=1}^{3}\sqrt{\left(\lambda_j - D_{av}\right)^2} \qquad [3.52]$$

The anisotropy is normalised for A_{fa} in a similar way to A_{ra} and provides an easily interpretable physiological 'stain' represented as a grey scale image (128).

Several other anisotropy indices have been proposed, notably those involving intervoxel averaging methods, such as the lattice index (A_{li}) (84). These indices compare diffusion tensors (or ellipsoids) from surrounding voxels to provide a scalar measure of diffusive similarity. However, while intervoxel averaged indices tend to be relatively insensitive to the effects of noise, they provide anisotropy maps with poor spatial definition near to the intersections of different structures (129). This is a consequence of the intervoxel averaging process and results in the indices being of little use when studying small areas of tissue.

3.7 Clinical applications of diffusion imaging

The motion sensitivity of diffusion imaging conventionally precluded its use in many clinical situations. However, with faster pulse sequences and improved motion correction techniques being developed, diffusion imaging is finding increased clinical use, particularly in the head, due to its relative immobility (21,127,128). Diffusion imaging has been shown to be useful in identifying many pathologies that result from changes to the biophysical properties of the cellular environment. The most common application is for the early identification of ischaemic brain injury associated with stroke (21,92,111,130-138). However, other studies have demonstrated diffusion changes in multiple sclerosis plaques (139,140), intracranial tumours (141-143), hydrocephalus (144) and schizophrenia (145). With the advent of DTI and the possibilities for further characterising structural processes, more applications are likely to arise, where a detailed analysis of the breakdown of structural features, is required.

3.7.1 Ischaemic brain injury

Conventional T_1 and T_2 weighted MRI techniques show a disappointing level of tissue contrast for ischaemic lesions, particularly in the early hours after stroke onset when the insults are, possibly, in a reversible phase (146). Diffusion-weighted MRI shows much improved contrast due to a drastically reduced diffusion coefficient

(typically 40%), particularly in the acute stages of the ischaemic process (21,92,111). The underlying biophysical reasons for the diffusion changes are not well understood, although several theories have been proposed (87,88,122,123,147). It was initially thought that the change may be due to alterations in the permeability of the cell membranes, although this has since been demonstrated to be an unlikely cause. Experiments on red blood cell suspensions suggest that permeability changes cannot fully account for the reduction in ADC (123). Monte Carlo simulations of a simple model of diffusion in grey matter also suggest that the ADC is relatively insensitive to changes in the membrane permeability (122).

It is believed that the reduction in ADC is due to processes involved with the cell swelling associated with cytotoxic oedema. It is hypothesised that a shift of water from the extracellular space into the intracellular space results in the path of the extracellular water becoming more tortuous, and hence a lower measured diffusivity. This increased tortuosity has been supported by tracer experiments and in Monte Carlo simulations (123), where a change in extracellular diffusion has been shown to have a significant effect on the ADC. It is also hypothesised that the change in ADC may directly result from the water shift between extracellular and intracellular compartments and their resulting change in space fractions (147). The ADC is assumed to be an average of the intracellular and extracellular diffusivities. However, as discussed in §3.6.1, the intracellular diffusivity is considerably lower, resulting in a biexponential decay of the signal attenuation with b-value (Eq. 3.14). For current in vivo MR diffusion experiments, relatively low diffusion weighting b-values are used (<1000 smm^{-2}), sampling the faster portion of this biexponential decay which biases the measurements towards the extracellular diffusivity (123).

Norris et al. (87) explained the phenomenon of restricted or hindered diffusion in the context of the changes occurring in ischaemic brain injury. The cell swelling associated with ischaemic tissue causes the extracellular water to become hindered and so to diffuse less freely. In cytotoxic oedema the swelling reduces the extracellular space by around 50%, resulting in some extracellular water becoming trapped in pockets between the swollen cells. The water trapped in these pockets can only diffuse into the cells at the same rate as water diffuses out; a process that is governed by the intracellular diffusivity. Thus, if the experimental diffusion time (Δ)

is greater than the time taken to diffuse across the pocket, the ADC will no longer be an average of the intracellular and extracellular diffusivities, but will tend to the intracellular value. In practice, only some of the extracellular water within a lesion is trapped in pockets and so the observed ADC is considerably higher than the intracellular diffusivity.

During the later acute and chronic stages of ischaemia, the ADC begins to recover, eventually reaching a value that is higher than its pre-stroke normal value. This behaviour has been shown to be consistent with disintegrating membranes and a decreasing cellular volume fraction, associated with vasogenic oedema (122). The time scale of these changes is highly patient dependent and is not well known. It is believed that the ADC is drastically reduced minutes after an ischaemic attack (132), but sometimes further continuing to a minimum value around 2-3 days later. The recovery of ADC associated with vasogenic oedema is also highly variable, but a trend towards higher than normal values is believed to occur after several weeks (135,137).

3.7.2 Other neurological pathologies

Multiple sclerosis (MS) is a chronic disease of the nervous system, mainly prevalent in young and middle aged adults. The myelin sheaths surrounding the nerves are damaged during the process, resulting in improper nerve function. However, the underlying causes for the nerve damage are not well understood. MS plaques have been identified with imaging and show a higher than normal diffusivity relative to normal white matter regions (139). The diffusivity was found to be higher in the acute phase as compared to chronic plaques, which is believed to be due to an increase in the extracellular space due to the demyelination process. It has also been observed that diffusivities may be higher in normal appearing white matter for patients diagnosed with MS as compared to normal control patients (140).

Schizophrenia is believed to result from disturbances in the frontal-striatal-thalamic circuitry (145). The evidence to suggest this is devised from indirect evidence from psychopharmacology and analogies with animal research. A technique such as DTI that enables the degree of ordering of structural fibres, such as

white matter tracts, to be observed would potentially enable verification of such a hypothesis. It would be expected that if the fibre tracts were broken or not well ordered then a decrease in diffusion anisotropy would be observed. Significantly lower anisotropy has been observed in a number of studies, both within the white matter of the prefrontal cortex (145) and more widely within the frontal-occipital regions of both hemispheres (26).

Hydrocephalus results from an abnormally large amount of CSF within the ventricles in the brain. Combined with the unyielding nature of the skull in adults, this condition often results in an increase of intracranial pressure resulting in drowsiness and vomiting. Hydrocephalus, of both normal and high pressure varieties has been observed to give rise to elevated diffusivities in many areas of the brain, both in grey and white matter (144). The increase is believed to be due to changes in myelin associated bound water, increased Virchow-Robin perivascular CSF spaces and an increased extracellular brain water fraction.

Conventional imaging strategies often give poor discrimination between different types of intracranial brain tumours (23). The ability to identify the different types of tumours pre-operatively would be of great value to the process of surgical planning. Diffusion imaging shows promise to identify the differences between solid and fluid filled tumours as well as characterising the extent of tumour and surrounding oedema (143). Studies suggest that the ADC of surrounding oedema is considerably higher in cerebral metastases when compared to high grade gliomas (141). The ADC has also generally been found to be higher in cystic or necrotic tumours, as would be expected. This observation suggests that diffusion imaging could be used to distinguish between cerebral metastasis and high grade gliomas pre-operatively. Diffusion imaging has also been used quantitatively, in the form of ADC and anisotropy measurements, to distinguish between normal white matter, areas of necrosis or cyst formation, regions of oedema and solid enhancing tumour (142).

4

Diffusion Imaging: Acquisition and Post-Processing Procedures

4.1 Introduction

This chapter describes the general methods used during the course of this study to acquire and process diffusion data. Firstly, the MRI hardware is discussed in §4.2 with particular emphasis being placed on the differences and similarities between the Siemens 63SP Magnetom and Elscint Prestige clinical MRI scanners used during this work. The specifics of the two diffusion-weighted pulse sequences are discussed in §4.3; the navigated spin-echo sequence on the Siemens scanner; and the EPI spin-echo sequence on the Elscint scanner. Differences between the data formats used by the two scanners are highlighted, along with the software developed to transfer the image data from the scanners off-line to the workstations. The software and methods used to perform the navigator motion correction are presented in §4.3.2. Finally, the software developed to calculate the quantitative diffusion maps, using both orthogonal and diffusion tensor acquisition schemes is discussed in §4.3 for the Siemens scanner and §4.4 for the Elscint scanner. The development of this post-processing software occupied a large proportion of the available project time and was written mostly in C.

4.2 MRI hardware

4.2.1 The magnet and B_0 field gradients

During the course of this work, two clinical, whole body MRI scanners were used; the Siemens 63SP Magnetom, operating at a B_0 of 1.5T; and the Elscint Prestige, operating at 1.9T. The magnetic field strength determines the proton resonant frequency, as governed by the Larmor equation (Eq. 2.6), giving 63 MHz at 1.5T and 80 MHz at 1.9T. These relatively high magnetic field strengths cannot be produced uniformly over whole body imaging volumes with conventional resistive magnets, resulting in the use of superconducting magnets (148). These magnets typically consist of a superconducting material, such as a niobium-titanium alloy, wound axially into the form of a solenoid. If the solenoid were infinitely long then the magnetic field would be uniform over its cross-section, but due to the finite length of the magnet (and hence solenoid) additional correction coils are required. The superconducting windings must be maintained at or below the critical temperature of around 4 K, resulting in the need for an efficient cooling system, currently based on a liquid helium and nitrogen cryogenic system. Much of the overall scanner size results from the need for a well insulated system to minimise expensive helium boil off.

The main B_0 magnetic field gradients are those applied along x, y and z to provide spatial localisation. The gradients are produced by an arrangement of coils placed within the magnet bore. The G_z gradient is the simplest to produce, using an arrangement of concentric axial loops (usually Helmholtz coils) in which the current direction is opposed. The G_x and G_y gradients are produced by Golay-saddle coils as illustrated in Fig. 4.1. The maximum gradient strength, produced by the Siemens scanner is 10 mTm^{-1}, while the Elscint scanner has a maximum of 15 mTm^{-1}. The magnitude of these gradients has particular importance for diffusion imaging, as one of the factors that determines the maximum b-value and hence the smallest measurable diffusivity is the gradient strength. As seen from Eq. 3.14, the b-value increases with the square of the diffusion gradient strength.

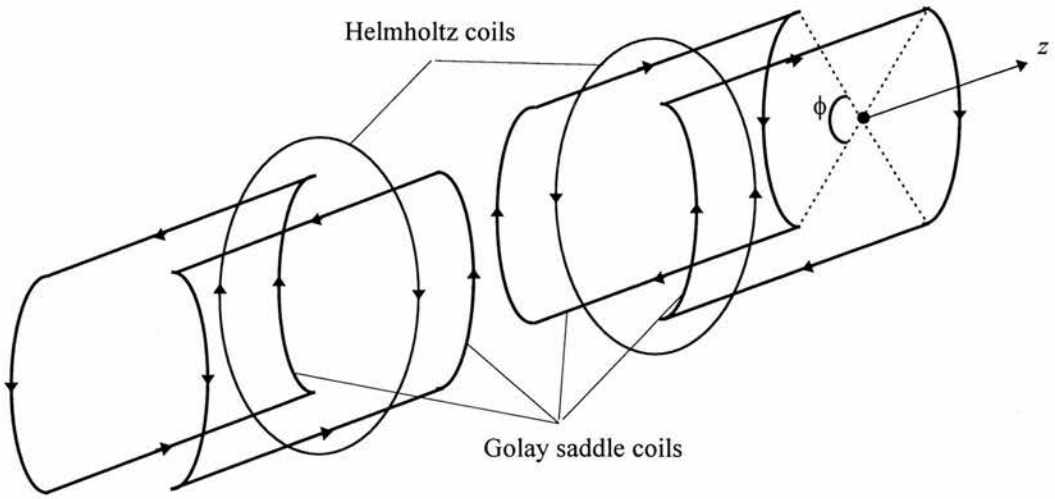


Figure 4.1 A typical gradient coil arrangement demonstrating the Helmholtz coils which produce the G_z field and the four Golay saddle coils which produce the G_x and G_y fields.

Another important gradient parameter is the rise time (the time taken to reach the maximum gradient strength) as this determines whether it is possible to perform rapid imaging techniques that rely on fast gradient switching, such as EPI. The Siemens scanner has a gradient ramping rate of 10 $mTm^{-1}ms^{-1}$, while the Elscint has a rate of 30 $mTm^{-1}ms^{-1}$. The increased gradient switching capabilities of the Elscint machine make it possible to perform EPI imaging, although it has been recommended that switching rates of faster than 100 $mTm^{-1}ms^{-1}$ be used for performing reliable EPI (40). The other feature of the Elscint machine is that the gradient coils are actively shielded. This feature greatly reduces the induction of eddy currents in the gradient coils, which is essential when faster gradient switching is used (63,149,150).

4.2.2 Shimming: Improving the B_0 field homogeneity

The specified B_0 homogeneity for both the Elscint Prestige 1.9T and the Siemens 63SP Magnetom 1.5T is for 5 ppm or less over a 50cm diameter sphere. This homogeneity is sufficient for producing acceptable images using most conventional acquisition procedures. Any residual inhomogeneity resulting from imperfections in the coils producing the B_0 field, or due to magnetic susceptibility induced gradients within the sample, can be reduced by using additional current

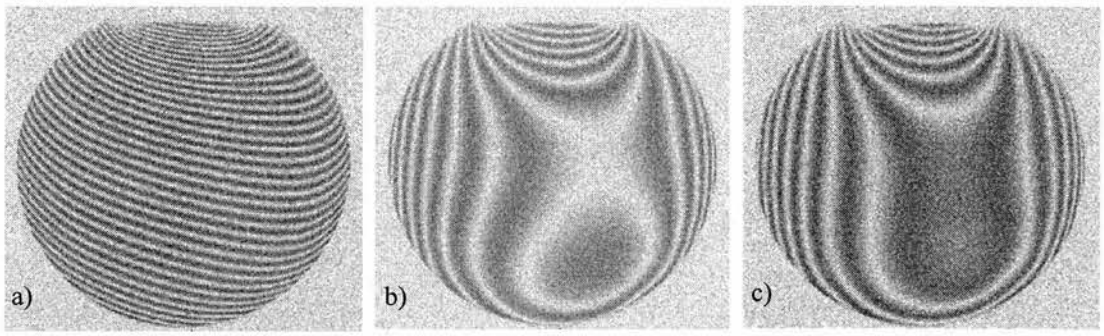


Figure 4.2 Field maps obtained from a water phantom using the Siemens scanner with a) zero, b) three and c) five shim coils with optimised currents. Each contour represents a field change of 0.1ppm. [Images appear courtesy of Dr Jim Wild]

carrying shim coils arranged strategically within the magnet bore. For spectroscopy applications, magnetic field homogeneity is critical, as it is necessary to resolve small chemical frequency shifts. Shimming is therefore required to achieve an acceptable homogeneity, typically of less than 0.1 ppm over the selected tissue volume. Echo planar imaging techniques are also sensitive to magnetic field inhomogeneities because of the long EPI readout trains, typically of 30 - 100ms compared with 5 - 15 ms for a conventional sequence. This results in a low bandwidth per pixel in the phase encoding direction and so shimming is highly desirable to minimise inhomogeneity induced artefacts.

The magnetic field $\boldsymbol{B}$ within the bore can be expressed as the gradient of a scalar potential V, satisfying Laplace's equation, $\nabla^2 V = 0$. The solution to Laplace's equation for the current carrying coils enables the representation of the magnetic field within the bore as a series of spherical harmonics (151,152). These harmonic solutions form an orthonormal set, which facilitates the design of additional shim coils of a specific shape and orientation to generate a single spherical harmonic that is independent of the field generated by other spherical harmonics (153,154). The currents in each of the coils can be adjusted separately to modify the magnetic field, without altering the components due to the other harmonics. Several shim coils can be constructed to produce fields corresponding to the different harmonic solutions and by altering each one individually, an increasingly homogeneous field can be built up. The Elscint scanner consists of just three shim coils corresponding to the linear

x, y and z solutions, while the Siemens scanner has 12 shim coils, which enable higher order corrections to be performed. The axial solutions along z (zonal harmonics) enable field corrections by passing currents through an axial arrangement of coils, while the orthogonal solutions along x and y (tesseral harmonics) require coils oriented obliquely to the x-y plane. Figure 4.2 illustrates field maps obtained from a water phantom using the Siemens scanner with currents optimised in zero, three and five shim coils.

Automated shimming algorithms can be implemented to determine and optimise the currents passed through the various shim coils. These techniques are usually based either on maximising the energy content of the FID envelope, or by using preliminary field measurements of the unshimmed region. The field mapping technique utilises a gradient echo sequence to provide measurements of the spatially localised phase difference $\Delta\phi$ between images obtained with slightly different echo times. The B_0 field can then be calculated from

$$\Delta\phi = \gamma B(x,y,z)\Delta TE \qquad [4.1]$$

where ΔTE represents the difference in echo time between the two gradient echoes (155-157). Most optimisation techniques then minimise the magnetic field inhomogeneity by performing a least squares fit of a target function based on a model of the inhomogeneity (158).

4.2.3 The RF transmitter and receiver (B_1 field)

RF coils are used to transmit the B_1 RF excitation pulses and to receive the resulting FID or echo signal. In some systems, a separate transmit and receive coil is employed, while others rely on a single coil to perform both tasks. As the transmit and receive coils operate at similar frequencies and need to be switched in and out rapidly, separate transmit and receive systems generally result in a reduction in interference related artefacts. However, the convenience of using one coil to perform both tasks means that many clinical systems use this method. Both the Siemens and Elscint RF head coils used during this study, were circularly polarised (quadrature) coils, designed to perform both the transmit and receive operations. The coils are circularly polarised to enable quadrature phase sensitive detection, acquiring real and

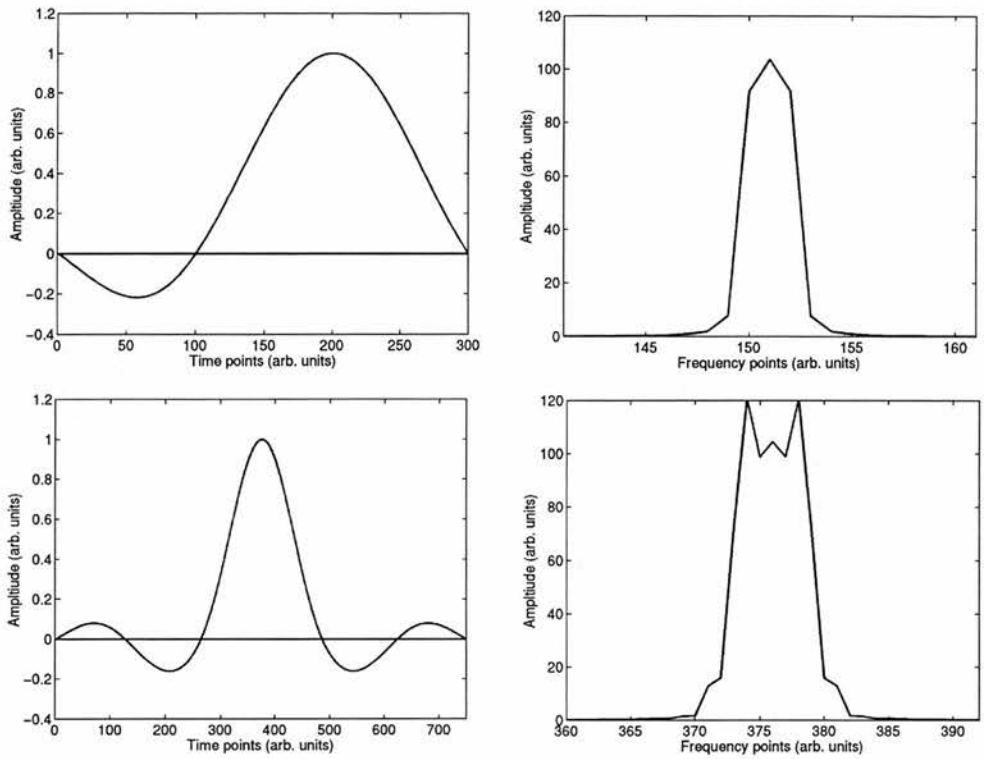


Figure 4.3 Illustration of the time modulated RF signal and its corresponding frequency spectrum for a) 90° and b) 180° RF pulses, used in the Elscint EPI spin-echo pulse sequences.

imaginary data channels. The data are converted from a MHz signal into a more computer friendly signal of the order of kHz, as described in §2.4.5.

The design and optimisation of RF coils is extremely complicated, involving sophisticated engineering and electronics principles and so only a brief overview is presented. The fundamental design requirement of an RF transmit-receive coil is that it produces a strong and homogenous RF field over the given excitation volume, while retaining a low resistance. These requirements suggest that the coil must have a high quality factor Q, which leads to an improved SNR. However, a practical compromise must be reached due to increased pulse ringdown time and bandwidth limitations with higher Q-values (as $\Delta\omega_0 = \omega_0 / Q$ at 3dB from maximum). Producing a uniform RF field is a difficult criterion to fulfil and many different solutions have been suggested for specific applications, although it is generally found that the coil must be considerably larger than the sample (159). The RF coil driver must be able to modulate effectively the RF signal to provide the necessary waveform which enables the excitation of a narrow frequency band corresponding to

the required spatial location. Examples of the RF pulse waveform and corresponding frequency excitation range are shown in Fig. 4.3 for the 90° and 180° RF pulses used in the Elscint spin-echo EPI sequence. To enable sufficiently short duration (hard) RF pulses, the transmitter coil is required to produce a substantial peak power, typically of several kilowatts. This requires considerable amplification of the modulated RF signal which demands robust coil circuitry design. In contrast, the signal induced into the receiver coil is very small and so must also be amplified. This is usually done in several stages, with the pre-amplifier being the most critical, since any extra noise that it induces will be amplified by subsequent stages. The strong RF signal required in the transmitter circuit must be isolated from the small signal induced in the receiver if interference is to be avoided, a process known as gating. This gating procedure serves a second purpose because pulsed NMR techniques also require a reliable signal cut off method to isolate the spin system from the transmitter RF after the required pulse duration (154,160).

4.2.4 The control systems

The computer operating system controls the operation of the transmitter, receiver, shim coils and gradient coils by means of microprocessors, as well as providing the user with an operating interface. The computer also performs the image reconstruction from the sampled raw (quadrature) data and enables viewing either directly on the screen or via hardcopy films or printouts.

The Siemens computer system is VAX / VMS based, using the Siemens Numaris user interface. The display system is relatively dated, using a rather laborious system of menus to select and run sequences and to view images. The RF pulse sequences, which include all RF pulse shapes, gradient waveforms and timing parameters are written in the Siemens pulse sequence programming language PARGEN. Pulse sequences were modified off-line on a Sun workstation and then transferred back to the VAX, where they can be compiled in a file appended with .UDB and run under Numaris.

The Elscint computer system is based on a Silicon Graphics Onyx, with a Unix operating system. The front end display is a modern windows environment,

which allows for selecting, running and displaying sequences. Again, pulse sequences are written in a dedicated programming language. The Elscint language has two basic levels for pulse sequence programming, so called .tsk task files and the RF and pulse timing .ptg files. The task files are programmed to expect RF pulses and gradients in a certain order, although the specific timing and pulse waveforms are not specified in the task file. The .ptg file gives specific details of the sequence timing, the shape of RF pulses and the gradient waveforms. The user only has access to the .ptg files, which is quite restrictive as it only makes it possible to alter the sequence timing and waveforms, it is not possible to add in new RF pulses or gradients. Once the sequence .ptg file has been modified, it is compiled along with its respective .tsk file, and can be run on the scanner system.

4.3 The diffusion-weighted pulse sequences

4.3.1 Navigated spin-echo

The diffusion imaging sequence implemented on the Siemens scanner was based on the navigated spin-echo technique described in §3.5.1. The pulse sequence used was modified from a standard Siemens double spin-echo sequence, with the addition of diffusion-weighting gradients. This work was performed by Dr Ian Marshall, prior to the author joining the group. The pulse sequence is illustrated schematically in Fig. 4.4. The image echo is acquired first (TE = 121 ms) and the navigator second (TE = 152 ms), following the method of de Crespigny et al. for improved image SNR (92). The phase encoding gradient is applied along G_y before the first 180° pulse, and must be unwound before the second echo forms. This is achieved by applying a second identical phase encoding gradient before the second 180° pulse. The unwinding gradient is applied in the same sense as the initial gradient because the 180° pulse effectively reverses its phase, relative to the first phase encoding gradient pulse.

The sequence also has a series of spoiler gradients, both around the second 180° pulse and at the end of the acquisition, as illustrated in the pulse sequence diagram in Fig. 4.4. The spoiler (or crusher) gradients at the end of the sequence serve to dephase any residual magnetisation before subsequent excitations. Those

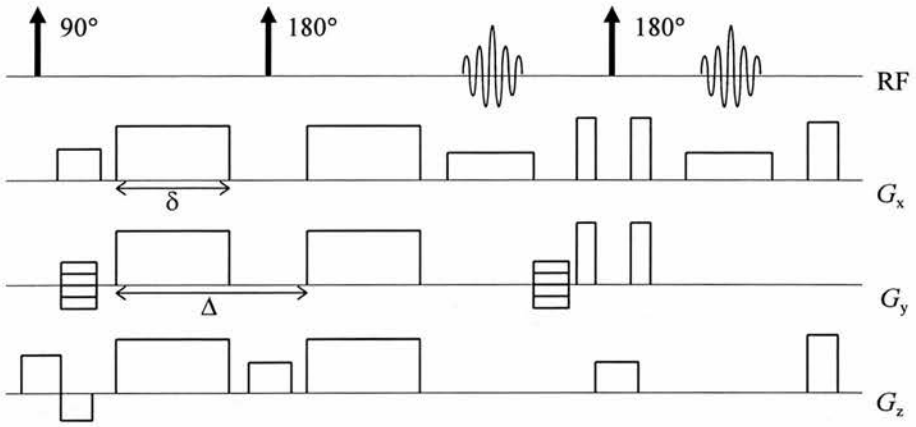


Figure 4.4 The navigated spin echo pulse sequence implemented on the Siemens 63SP Magnetom clinical scanner. The phase encoded image echo is acquired first and then the phase encoding is unwound before the acquisition of the navigator echo. The spoiler gradients placed around the second 180° RF pulse and at the end of the sequence are also shown.

applied around the second 180° pulse dephase unwanted magnetisation resulting from spins that undergo incomplete inversion, i.e. that see a flip angle different to 180°. The diffusion gradients around the first 180° pulse serve this purpose and so additional spoiler gradients are not required.

Due to the relatively low maximum gradient strength of 10 mTm^{-1}, the diffusion gradients have to be of relatively long duration (δ = 51 ms) relative to their separation (Δ = 57.4 ms), in order that sufficient diffusion weighting is obtained (Eqs. 3.14 & 3.16). This results in a relatively poorly defined diffusion time as the condition that $\Delta >> \delta / 3$ is not satisfied and the diffusion pulses provide poor approximations to the required delta functions (16,69). The diffusion gradients can be applied along any combination of the frequency-encode (x), phase-encode (y) and slice-select (z) directions. However, these combinations along with their respective magnitudes must be hard-coded into the PARGEN sequence and cannot be altered as an option at run time. The standard clinical protocol utilised a baseline diffusion sequence with the diffusion gradients set to zero and three diffusion-weighted sequences each with diffusion gradients separately applied along x, y and z with a magnitude of 9.64 mTm^{-1}, resulting in an approximate scalar b-value of 700 smm^{-2} (127).

4.3.2 Echo planar spin-echo

The diffusion sequence implemented on the Elscint scanner was based on the spin-echo EPI technique described in §3.5.2. The pulse sequence and EPI post-processing was written by Dr Paul Harvey (Elscint, Haifa, Israel) and was provided as standard on the Elscint machine, with the diffusion gradient applied along the slice-select direction. User access was provided to the .ptg sequence timing files and these were modified as part of this work to allow a combination of diffusion gradients to be applied along the frequency encoding, phase encoding and slice select directions. The pulse sequence is illustrated schematically in Fig. 4.5. Diffusion sensitisation was again provided by placing symmetrical gradients around the 180° refocusing pulse, of duration δ = 40 ms and separation Δ = 43.9 ms. Again, the diffusion time is not well defined as $\Delta >> \delta / 3$ is not satisfied. However, the maximum gradient strength of 15 mTm^{-1} allows for shorter duration diffusion gradient pulses to be applied than on the Siemens scanner, which is essential to minimise T_2 decay before the long EPI readout train. The standard clinical sequences employed diffusion gradients applied separately along the *x*, *y* and *z* directions with magnitude 14 mTm^{-1}, giving a nominal scalar *b*-value of around 686 smm^{-2}. A baseline sequence was also applied with the diffusion gradients replaced by spoiler gradients of magnitude 11 mTm^{-1} and duration 1 ms, to overcome the effects of

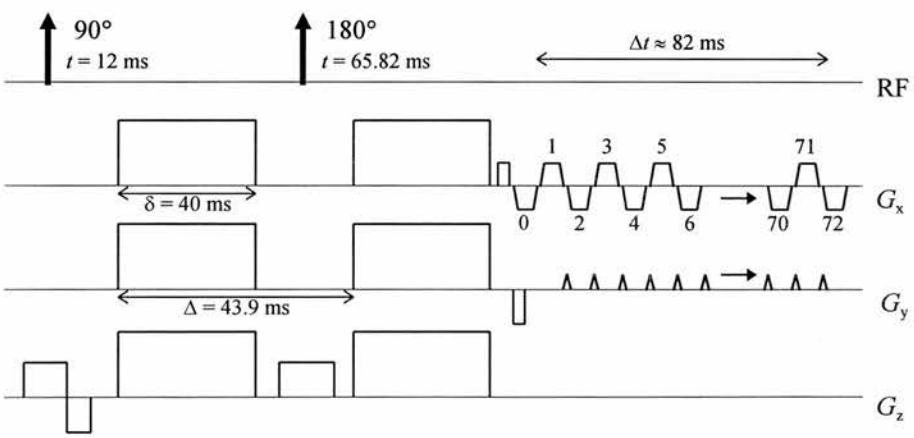


Figure 4.5 Illustration of the diffusion-weighted spin-echo EPI sequence implemented on the Elscint scanner. The echo maximum occurs at t = 119.64 ms, corresponding to the 8th echo in the EPI readout train.

incomplete refocusing, described in §4.3.1. Several diffusion tensor sampling schemes were also implemented with different combinations of gradients applied along x, y and z, but these are discussed in more detail in subsequent sections.

The EPI readout train is longer than used in typical sequences (20,161,72), because of the limited gradient switching capabilities of the Elscint scanner (30 $mTm^{-1}ms^{-1}$ maximum), which result in a total sampling time of around 82 ms. The readout is performed using a series of alternating gradients, applied along the frequency encoding direction, of duration 1120 μs and separation 10 μs, with a pulse shape as shown in Fig. 4.6a. During the interval between the readout pulses, a triangular phase encoding 'blip' (see Fig. 4.6b) of duration 200 μs is applied at 930μs intervals. Clearly, this pulse is considerably longer than the 10 μs interval between the readout pulses and so a considerable overlap occurs between the readout and phase encode gradients (95μs on both the rise and fall). As data is sampled at a constant rate every 2.5 μs, this overlap results in a distortion of k-space such that points are no longer evenly spaced on a rectangular grid. In fact, when the MR signal data is acquired at a constant time interval, any acquisitions that occur during the application of time-varying magnetic field gradients (e.g. trapezoidal rise periods), will result in non-equidistant k-space data points as indicated by Eq. 2.44. This effect can be corrected for, either by using non-linear sampling of the data points (56)

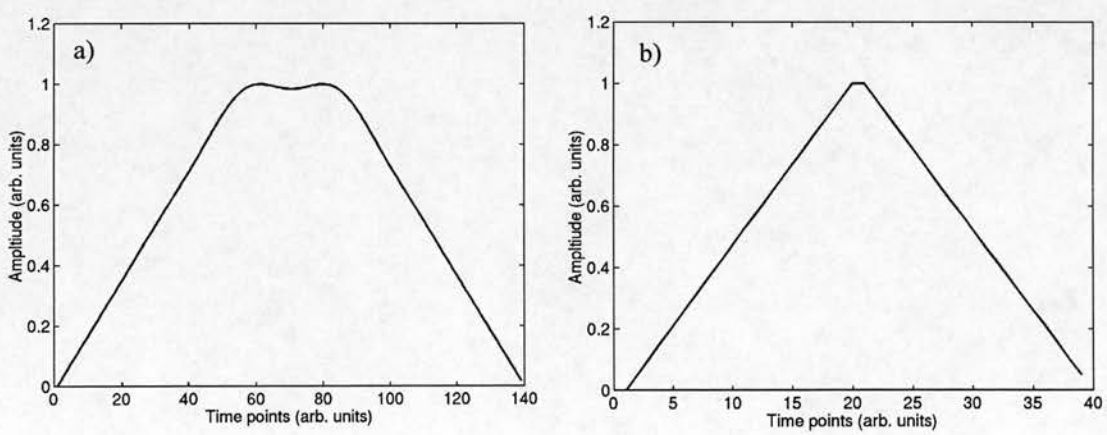


Figure 4.6 Illustration of the pulse shapes used for the EPI readout train in the Elscint pulse sequence, showing; a) the form of the readout pulse (duration 1120 μs) and b) the form of the phase encoding blip (duration 200 μs).

or by using sinc interpolation rasters which exploit the known gradient-time variation to shift the data points to their correct k-space positions (162-164). Unfortunately, the method used to correct for the k-space distortions was not made available to the author. Other correction procedures are also performed automatically on-line in IDL before the images are displayed. For example, phase correction is performed to eliminate the classic Nyquist ghost artefact, resulting from phase errors induced by acquiring positive and negative k-space lines alternately. However, the exact algorithms and methods used for these processes were also not made available to the author.

4.4 Siemens data transfer and post-processing

4.4.1 The data format and transfer

The data format used on the Siemens scanner is based on the digital imaging and communications in medicine (DICOM) format, which is the standard for medical imaging applications. The DICOM format stores information on patient details, sequence timing parameters, details of contrast agents used, as well as the image data. Each item is identified by a tag, which indicates the type of data stored (e.g. double, integer, char) and its length. Many of the tags are standard to all DICOM compliant formats, but there are also private tags available for storing additional information. The required data can be identified by using an appropriate data dictionary, listing the tags corresponding to the patient information, the image data etc. The standard DICOM tags are identified with a standard data dictionary, while the private tags and the information they contain are company specific and a private dictionary is required.

The Siemens image data is stored in two basic forms: the raw k-space data (.raw files) and the magnitude image data (.ima files). The file names consist of a 5 digit number followed by the .ima or .raw suffix, e.g. 00042.img for the 42nd image file. The .raw files contain alternating real and imaginary values along each row of k-space in turn, thus enabling construction of both phase and magnitude images after a 2DFT. The .ima files contain real numbers running sequentially along each row of the image (real space) matrix, with each row ordered by column number. Each data

point is in a byte-reversed format and so the bytes must be reordered, before the data can be used. The data transfer from the VAX system of the Siemens scanner, off-line to the UNIX based Sun workstations is performed using the UNIX batch files `dcntransfer` and `dcnrawtransfer`, written by Martin Connell. The .ima files from a specified patient directory on the VAX are transferred (via the patient order number) using `dcntransfer`, while `dcnrawtransfer` performs the transfer for both .ima and .raw files. The .ima files are then converted from the DICOM format to an ANALYZE (CNS, Rochester, MN.) readable format, using the program `conv_analyze_acrnema`, also written by Martin Connell. The ANALYZE data format consists of a file containing the image data (.img) and a header file (.hdr), containing the patient, sequence and image parameters. The data in the .img file is stored as signed short int values ranging from -32768 to 32768.

4.4.2 Navigator motion correction

As described in §3.5.1 and §4.3.1, the diffusion-weighted pulse sequence implemented on the Siemens scanner is based on the navigated spin-echo technique. The post-processing software was developed by Dr Ian Marshall, with different versions being evaluated based on the zero-order correction of Ordidge et al. (90), and the zero and first-order correction methods of Anderson and Gore (91) and de Crespigny et al. (92). The Anderson and Gore and de Crespigny methods differed in their implementation, with the former method applying a linear fit to the phase variation along the navigator profile, while the latter calculated independent phases for each point along the profile. A comparative study of the different correction schemes demonstrated that the zero and first-order correction scheme of de Crespigny et al. provided the most successful correction of motion artefacts in a series of 15 acute stroke patients (94). This work was extended by the author to include data obtained from 26 acute stroke patients. The zero-order phase correction was found to be successful in only 17% of images, compared with 27% for the zero and first-order correction, while the independent zero and first-order correction provided acceptable images in 67% of the cases. A typical example is given in Fig. 4.7, showing images obtained using the navigated spin-echo sequence with no

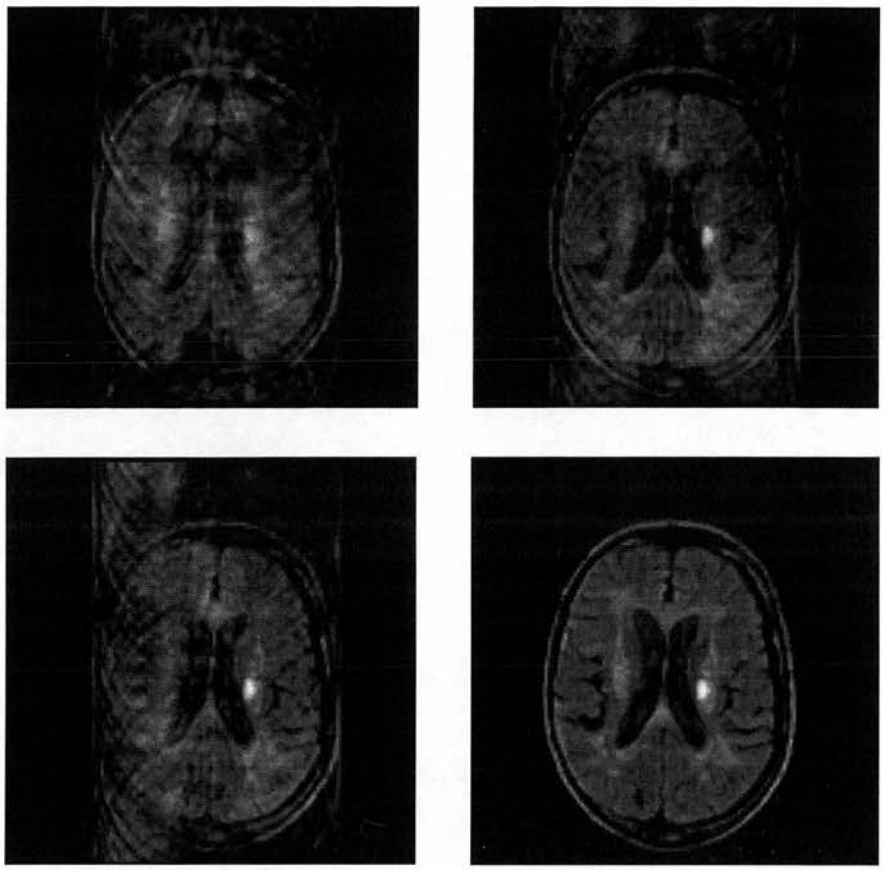


Figure 4.7 The effectiveness of the navigator motion correction technique, illustrated for typical data obtained from a patient suffering with symptoms of acute ischaemic stroke. The images shown are with; no correction (top left); zero order correction (top right); zero and first order linear correction (bottom left); and zero and first order independent correction (bottom right).

correction, zero-order correction, zero and first-order linear correction and zero and first-order independent correction schemes. As such, the independent zero and first-order method was used for all subsequent navigator data correction.

The navigator correction program `crespigny_nav_diff` was written in C and the processes involved are summarised by a flow chart representation in Fig. 4.8. Firstly, the raw k-space data is read in from the navigator and image .raw files, a process which requires the conversion of a VAX/VMS style floating point number into a C floating point form, along with byte reversal of each data point. The row (frequency encoding) Fourier transform is performed to obtain the navigator projection using a fast Fourier transform (FFT) algorithm obtained from Numerical Recipes (165) and based on the method of Cooley and Tukey (166). The output from the FFT algorithm must be reordered such that the data points run from the largest

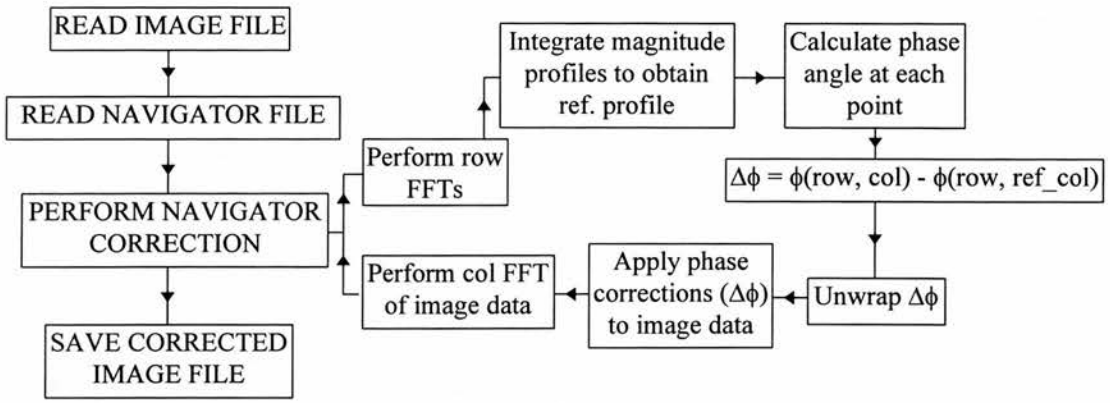


Figure 4.8 A flow chart illustrating the principal processes involved in the navigator correction program `crespigny_nav_diff`.

negative frequency component, through the zero component, up to the highest positive frequency component. The signal magnitude is then calculated and summed for each point along the projection and the column (phase encoding line) that corresponds to the maximum magnitude integral is then used as the reference projection.

The phase angle ϕ at each navigator data point is then calculated and subtracted from the reference phase ϕ_{ref} of the corresponding data point along the reference projection, to give the phase difference $\Delta\phi$ at each point. This phase difference must be unwrapped, such that if consecutive data points have a phase differing by greater than π, then $\Delta\phi$ must have a factor of 2π either added or subtracted such that the phase difference lies in the range $\{-\pi < \Delta\phi < \pi\}$. The phase correction can then be applied to each point in the image data projections (after the row FFT), by applying phase rotation such that

$$R_c = R\cos\Delta\phi - I\sin\Delta\phi \qquad [4.2a]$$

$$I_c = I\cos\Delta\phi + R\sin\Delta\phi \qquad [4.2b]$$

where R_c and I_c represent the real and imaginary components of the corrected image data point, and R and I represent the real and imaginary components of the uncorrected image data point. Thus, the phase error ($\Delta\phi$) calculated from the navigator data is used to correct the image data. The corrected magnitude projections are then normalised relative to the reference projection integral using the navigator integrals and the column FFT is applied. The magnitude image is then obtained and

saved in the Siemens .ima format. The first two numbers in the image file name were used to identify the form of the image data (as indicated by Table 4.1) and the last three to identify the image number, e.g. 02046.ima would indicate independent zero and first-order correction performed on image number 46. The corrected .ima files can then be converted using `conv_analyze_acrnema` to .img and .hdr files and displayed using ANALYZE.

4.4.3 Quantitative orthogonal diffusion analysis

As stated in §1.1, an early goal of this project was to obtain a quantitative diffusion analysis from the images obtained using the Siemens navigated spin-echo sequence. This required the writing of a post-processing program to perform the pixel-by-pixel diffusion calculation. The program `dse` was written in C and contains the basic functions necessary to produce quantitative diffusion maps, as illustrated by the flow chart in Fig. 4.9. The program listing is also given in Appendix 1.

The necessary input to `dse` is provided using a simple step-by-step on screen menu that appears at run time. The program requires, the study name, the navigator correction type (as image file names are prefixed by a number corresponding to the correction type used), the baseline and diffusion-weighted image numbers, the nominal *b*-value used and the number of slices. The standard FOV of 230 × 230 mm

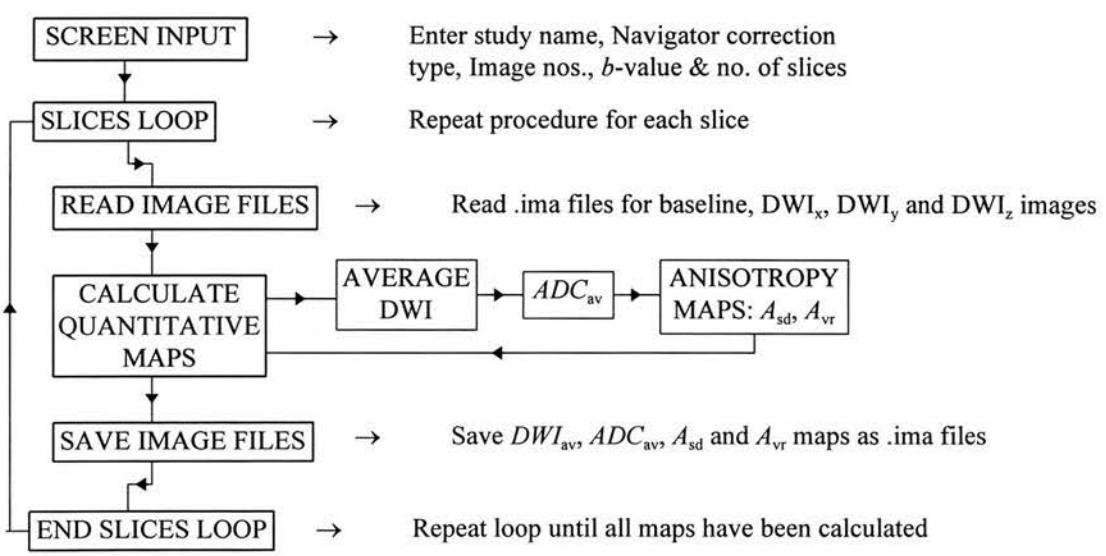


Figure 4.9 A flow chart illustrating the main processes involved in the calculation of quantitative diffusion data using the orthogonal sampling technique.

and matrix size of 128 × 128 are hard-coded into the program. The data from the baseline and three orthogonal diffusion-weighted images (.ima format) are read and stored in data arrays for the first slice, with appropriate byte-reversal. A pixel-by-pixel analysis is performed, whereby the average diffusion-weighted image (DWI_{av}), the ADC_{av} and the anisotropy maps are calculated. The DWI_{av} map is calculated on a pixel-by-pixel basis, simply by summing together the three orthogonal DWIs and dividing by three. The ADC_{av} map is calculated by averaging the ADC maps calculated along the three orthogonal directions, i.e. ADC_x, ADC_y and ADC_z, which are calculated from Eq. 3.14, such that

$$ADC = \frac{\ln\left(S_{b_1} / S_{b_2}\right)}{b_2 - b_1} \qquad [4.3]$$

where S_{b_1} is the signal obtained with the baseline b-value b_1 and S_{b_2} is the signal obtained with the diffusion-weighting gradients applied to give a b-value of b_2. The b-values were calculated numerically from the pulse sequence by evaluating the integral in Eq. 3.14b. This procedure was performed by using the program `b_matrix` written by Dr Mark Bastin. The b-matrix is calculated for each nominal scalar b-value and then the appropriate element can be used in the calculation i.e. b_{xx}, b_{yy} or b_{zz}. The anisotropy maps are then calculated on a pixel-by-pixel basis from the three orthogonal ADC maps, for example by using Eq. 3.46. The various calculated

File Prefix	Data Type
00	Standard Siemens .ima File
01	2D-FFT: No navigator correction
02	Independent zero and first order correction
05	Linear zero and first order correction
06	Zero order correction
07	Average diffusion-weighted image DWI_{av}
08	Trace ADC_{av}
09	Standard deviation anisotropy A_{sd}
10	Volume ratio anisotropy A_{vr}

Table 4.1 The file identification codes for the processed Siemens data. The prefix code is followed by a three digit image number and appended with the .ima suffix.

diffusion maps are then saved as .ima format files, prefixed by an appropriate number as indicated in Table 4.1. The post-processing procedure is then repeated until all of the slices have been processed. Finally, the files are converted to the .img and .hdr format using `conv_analyze_acrnema` and can be displayed using ANALYZE.

4.5 Elscint data transfer and post-processing

4.5.1 The data format and transfer

The data obtained from the Elscint scanner is also stored in a DICOM based format. Again, there are two basic data forms, the magnitude image files (IMG) and the raw data files (FID). The magnitude IMG files consist of data values for each row and column of real space, as before. The phase corrections necessary to obtain undistorted EPI images have been undertaken on the IMG data, but not on the FID data. The Elscint scanner performs the frequency encoding Fourier transform automatically during image acquisition and the FID data files therefore consist of data stored with the frequency encoding Fourier transform already carried out. Therefore, to obtain true k-space data, it is necessary to perform the inverse Fourier transform along the frequency encoding direction. Every image obtained on the Elscint scanner is stored with a unique identification number and the file name is of the form s003e1IMGf2.IMG.52.123456, where s003 identifies the third slice, e1 indicates the first echo, f2 indicates the second frame (or acquisition), IMG identifies the data type and the final set of numbers is the uniquely generated identification number.

The data is transferred off-line from the Onyx computer to the Sun workstation using standard ftp procedures. The data in the IMG files is stored without byte reversal and is sorted and converted to the ANALYZE readable .img and .hdr files, using the program `dicom_scale_order`, written by Martin Connell. This program also sorts the unique image file names by slice and acquisition number and renames them into a more logical form of [scan number_slice number_acquisition number].img. A scale factor is also read from the header in the IMG file and stored into a file appended .scale. For display on the scanner, all images are scaled between zero and 4095, such that the pixel with the highest signal

value will be assigned a value of 4095. The IMG file contains data values of this scaled form and so to obtain the actual values required for quantification, it is necessary to divide each data value in the IMG file by the scale factor given in the header. The raw FID data files have not been used for this study because of the need to develop EPI phase correction software as detailed in §4.3.2, a task that was considered to be beyond the scope of this work, due to a lack of availability of technical information on the pulse sequence and acquisition methods.

4.5.2 Quantitative orthogonal diffusion analysis

The software developed for producing quantitative diffusion maps from data obtained using the Elscint scanner, follows the same basic procedures as the Siemens data processing strategy illustrated in Fig. 4.9. However, the Elscint data processing was performed on the .img format files after conversion using `dicom_scale_order`. This was considered to be a more flexible approach as it enables the software to be used with data obtained from any scanner, once it has been converted to the ANALYZE readable format. A few additional features were also added to make the processing routine more flexible, and to perform the image scaling procedure. The processing was split into three main stages, the first program, `scale_ort`, reads in all of the baseline and diffusion-weighted images, slice-by-slice, and divides each pixel by the appropriate scale factor obtained from the .scale file as described in §4.5.1. The corrected images are saved into a folder corresponding to the slice number. At this stage, an option exists to crop the image, as the rectangular FOV of 256 × 128 is not efficient for brain imaging and can be cropped to 128 × 128 to reduce subsequent data storage and processing requirements.

The sorting of images into folders corresponding to their slice number facilitates the easy implementation of the second processing stage: image alignment using SPM95 (Hammersmith hospital, London, UK). This stage can either be omitted or performed, depending on whether a stationary phantom has been used, or a subject that is likely to have moved between scans. For all potentially non-stationary subjects, it is essential that the individual images are realigned before subsequent calculation of the quantitative diffusion maps. The implications of image

alignment are discussed in more detail in subsequent chapters. Images that are aligned using SPM95 have a prefix 'r' added to the image file name.

The final processing stage involves the calculation of quantitative diffusion maps using the C program `map_ort`. This program performs the same basic procedures as `dse` does for the Siemens data, in §4.4.3, with a few additional features described below. Different numbers of baseline and diffusion-weighted acquisitions are allowed, which is essential for implementing efficient diffusion encoding schemes as discussed in Chapter 7. The program was extended to enable the automated processing of SPM aligned diffusion images. Also, image cropping can be implemented at this stage, if required, with appropriate reductions in the matrix size and storage requirements. The programs `scale_ort` and `map_ort` allow for a variable FOV (over a restricted range). An entirely flexible FOV choice is not available because of the dependency of the *b*-value on the FOV. Clearly, it would be impractical to calculate the *b*-values numerically from the pulse sequence parameters during run-time, using Eq. 3.14b, for every FOV and so the *b*-values calculated are hard-coded into the program for a reasonable choice of FOVs. The values obtained are calculated, numerically using the C-program `b_matrix`, written by Dr Mark Bastin.

4.5.3 Quantitative diffusion tensor analysis

The ability to acquire diffusion-weighted images with a short acquisition time using EPI methods, facilitates the realistic introduction of DTI into the clinical setting. Thus, with the arrival of the EPI-compatible Elscint scanner, the opportunity was taken to develop, optimise and evaluate a DTI acquisition and post-processing scheme. Evaluating the diffusion tensor from the DWIs requires a considerable amount of post-processing and so the development and evaluation of suitable software occupied a large proportion of the available project time. As with the software developed for the Elscint orthogonal imaging technique, the tensor post-processing was split into three main stages; the scaling and sorting of image files; image alignment using SPM and custom developed software; and finally the

calculation of the diffusion tensor and quantitative maps. Some custom written display software was also required and this is discussed in more detail in Chapter 8.

The image scaling procedure is analogous to the orthogonal case in §4.5.2, scaling the images using the appropriate .scale file and sorting the images into folders corresponding to the slice number. Obviously, this involves processing many more images as the tensor analysis requires at least seven DWIs (24) (usually with multiple acquisitions) as opposed to the four used in the orthogonal technique. The C program `scale_ten7` written to perform this task also samples the background noise in each image, saving the information to a file with a .noise suffix. This noise information is required for calculation of the covariance matrix used in the multivariate linear regression analysis. As before, SPM image alignment can be performed, to ensure that all the images are co-registered before the tensor is calculated from the component DWIs. It may also be desirable to perform corrections for eddy current distortions at this stage. As described in §3.5.2, eddy currents result in geometric distortions of the resulting images, dependent on the applied gradient strength and direction. Before images are registered, it may be necessary to correct for these distortions. This can be achieved by using a series of programs, written by Dr Mark Bastin, to perform the iterative cross-correlation correction (97,100).

The final stage of the data processing involves performing multivariate linear regression to obtain the diffusion tensor from the DWIs, followed by subsequent calculation of the various quantitative maps. These processes are performed using the C program `map_ten7fsr` and are summarised by flow-chart in Fig. 4.10. The program listing is presented in Appendix 1. The necessary input is again provided by following an on screen step-by-step menu, as summarised in Fig. 4.10. It should be noted that the number of *b*-values and acquisitions obtained at each *b*-value is variable. However, as before, the *b*-matrices are calculated using the program `b_matrix` and are then hard-coded into the program. This limits the choice of *b*-value to those that have been calculated *a priori*, by numerically evaluating the integral in Eq. 3.24, a process that is too time consuming to integrate directly into `map_ten7fsr`.

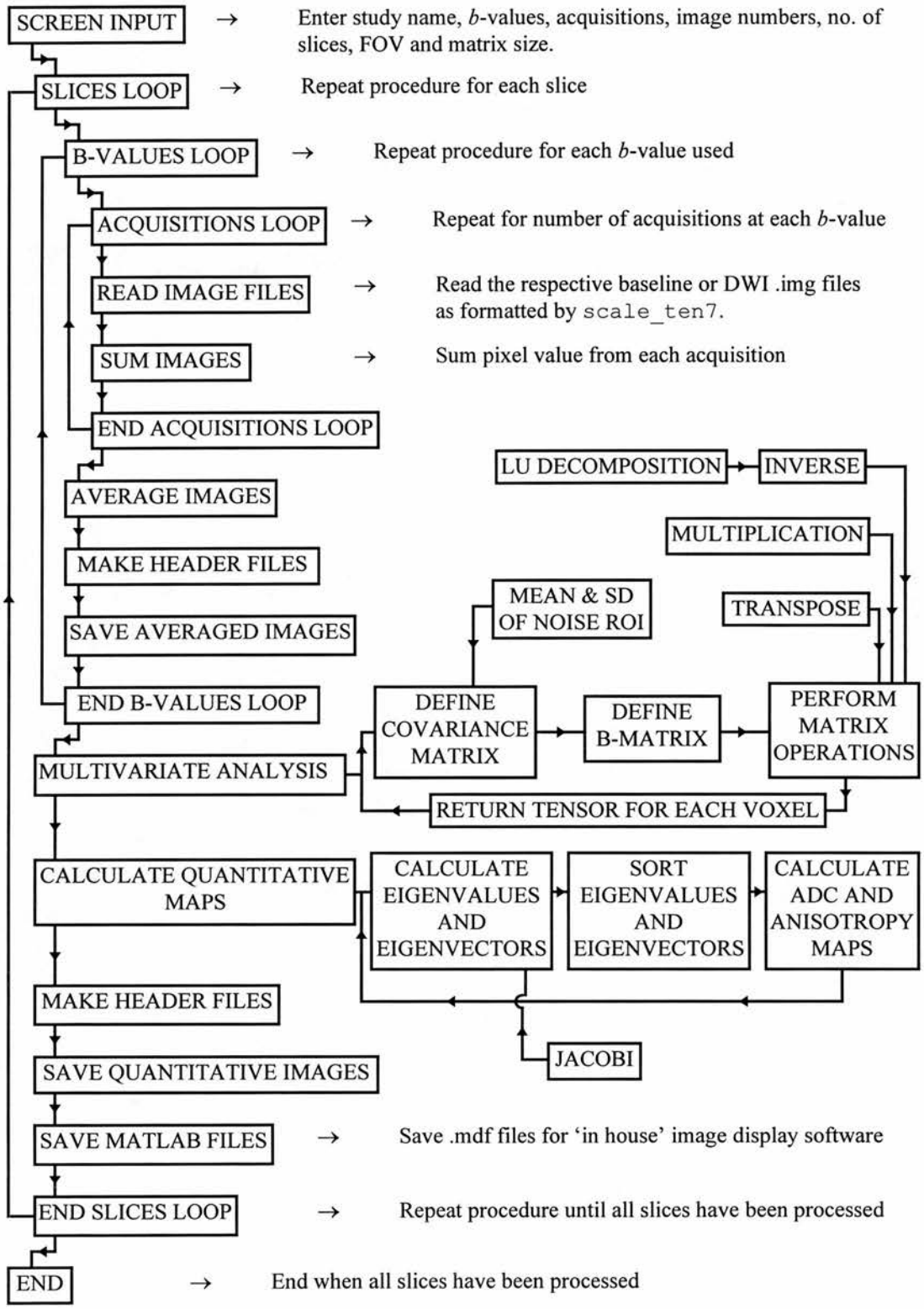


Figure 4.10 Flow chart illustrating the main processes involved with performing the DTI post-processing, using `map_ten7fsr`.

The processing is performed on a slice-by-slice basis, with the ANALYZE format image output from `scale_ten7` (following realignment) being read in for each *b*-value and individual acquisitions are averaged. The averaged images are saved for each *b*-value and gradient direction as .img and .hdr files. Once all of the images have been averaged, multivariate linear regression is performed on a pixel-by-pixel basis, as described in §3.4.2. The diagonal elements of the covariance matrix Σ are calculated by taking the mean signal in the pixel and dividing it by the standard deviation of the background noise in the respective image, which is obtained from the .noise file outputted from `scale_ten7`. The matrix $\boldsymbol{B}$ is defined as in Eq. [3.26] and the optimal tensor parameters α_{opt} can be estimated by performing the matrix operations given in Eq. [3.30]. The matrix inverse was calculated using the LU decomposition algorithm given in Numerical Recipes (30). The optimal solution for the diffusion tensor can then be stored in a data array, and the process repeated until the optimal tensor has been estimated for all pixels in the image.

With the tensor estimated in the measurement coordinate system, it is useful to obtain the representation in the principal coordinate system as described in §3.4.3. This is achieved by calculating the eigenvalues and eigenvectors for each pixel in the image. An algorithm based on Jacobi transformations was implemented and the eigenvectors were then sorted into descending order along with their appropriate eigenvectors (165). The required ADC and anisotropy maps are then calculated as required, using for example Eqs. [3.43, 3.44, 3.51 & 3.52]. Once calculated, the maps are saved as ANALYZE format .img and .hdr files, ready for subsequent display. The eigenvalues and eigenvectors are also saved into a binary data format with suffix .mdf, to comply with the in house image display software written in Matlab (The Mathworks, Natick, MA., USA).

5

Orthogonal Diffusion Imaging in the Clinical Setting: Preliminary Observations

5.1 Introduction

There has been considerable interest in measuring diffusion changes in stroke patients, particularly in the early acute stages, where the process is, possibly, reversible (21,111,167,168). The ability to quantify diffusion changes reliably over time, potentially enables a quantifiable evaluation as to the effectiveness of reperfusion therapies at returning the tissue to its normal state. The normal time course of diffusion changes in ischaemia is not well established and is extremely subject dependent (111,135,169). However, most studies suggest that the ADC is considerably reduced, typically by 40%, in the early acute stages (< 3 days). The majority of this reduction occurs within the first few minutes in animal models (146) which means that by the time a patient has undergone normal hospital admissions procedures, a significant ADC reduction has occurred before the initial scan can be obtained. Evidence suggests that this initial reduction may be followed by a more gradual decrease, until a minimum value is reached after around 3 days. After this period, the ADC begins to rise due to the breakdown of cell membranes, and finally becomes elevated above normal with the onset of vasogenic oedema, typically occurring after a period of several weeks. Significant diffusion anisotropy has been demonstrated in normal white matter on many previous occasions (71,128), but

studies of anisotropy changes during stroke have not been extensively reported (126,169). The few studies to date suggest that there may be an increase in diffusion anisotropy during the early acute stages, followed by a decrease during later acute and chronic stages. The possible biophysical mechanisms resulting in these diffusion changes were discussed in §3.7.1.

Most diffusion measurements obtained in vivo have used state of the art scanners with high specification gradient performance, resulting in the technique being predominately confined to larger research institutions. The ability to obtain clinically useful, quantitative diffusion images, using a conventional, low specification, clinical scanner would clearly be extremely useful for many hospital sites. The navigated spin-echo method offers the potential to obtain diffusion images using conventional gradient systems (90-92). Therefore, this chapter discusses the implementation of the navigated spin-echo diffusion imaging technique in the clinical setting. The feasibility of performing an in vivo quantitative evaluation using diffusion sampling along three orthogonal directions (21) is assessed and its limitations discussed. The technique was evaluated using in vitro phantom data and in vivo data obtained from 36 patients suffering with symptoms of acute ischaemic stroke. Measurements of ADC_{av} and diffusion anisotropy were performed, with a view to establishing the likely reliability of making such measurements in vivo, using the clinical equipment available. The reliability of these measurements is discussed in terms of the effectiveness of the navigator motion correction algorithms, the limitations of the orthogonal sampling scheme and the likely contribution from noise contamination.

5.2 Experimental methods

5.2.1 Acquisition techniques

Diffusion-weighted imaging was implemented on the Siemens 63SP Magnetom clinical scanner, equipped with conventional gradients of maximum amplitude 10 mTm^{-1} and 1 ms rise time. A spin-echo based sequence, with an image echo time of TE = 121 ms was adapted with diffusion-sensitising gradient pulses of magnitude G = 9.64 mTm^{-1} and duration δ = 51 ms, placed symmetrically around the

180° RF pulse as discussed in §4.3.1. The separation of the leading edges of the pulses was Δ = 57.4 ms, leading to a scalar diffusion attenuation *b*-value of approximately 700 smm^{-2}. Motion artefacts were corrected using a non-phase encoded navigator echo (TE = 152 ms), which allowed phase distortions caused by non-rigid body motion to be corrected on a point-by-point basis, as discussed in §3.5.1 and §4.4.2 (90-92). Artefacts due to pulsatile motion were reduced by cardiac gating of the image acquisition with finger pulse triggering, using a repetition rate of two or three cardiac cycles depending on the patient heart rate.

Diffusion-weighted images were obtained with diffusion gradients applied in three orthogonal directions, corresponding to the left-right (x), anterior-posterior (y) and head-feet (z) directions. A reference image (b = 0) was also obtained with the same sequence parameters as above, but with no diffusion gradients applied. The directions of the phase encoding and readout gradients were swapped, such that the diffusion gradient was applied along the phase encoding direction for both the x and y acquisitions. This was implemented to achieve a greater success rate with the navigator correction, as described in §3.5.1. Eight transverse slices were obtained for each patient, leading to 32 images in total. All images were obtained with a matrix size of 128×128, a slice thickness of 5mm and a square field of view of 230mm.

5.2.2 Post-processing procedures

Raw k-space data was transferred to a Sun Ultrasparc workstation (Sun Microsystems, Mountain View, CA., USA.) for further analysis as discussed in §4.4. Fourier transformation of the navigator data was applied in the frequency encoding direction to yield navigator profiles. Phase angles along these profiles, relative to a reference profile, were then determined using (i) a zero-order phase angle (90), (ii) a zero and first-order linear phase variation (91) and (iii) independent phase angles for each data point (92). Image data profiles were then corrected using the navigator phase information and Fourier transformed in the phase-encoding direction. The images were displayed using ANALYZE and judged as to whether they were acceptably free from artefact, for each of the navigator correction procedures. The

set of images deemed to be the most artefact free were then used for further quantification.

ADC_{av} images were calculated from the navigator corrected image data sets on a pixel-by-pixel basis using Eqs. 3.43, 3.44 and 4.3, to obtain maps of $tr(\boldsymbol{D})$. The ADCs were calculated from the diagonal elements of the b-matrices, used for diffusion tensor imaging, such that remaining gradient interaction effects were negligible and ADC_{av} was found to be a good approximation of $tr(\boldsymbol{D}) / 3$. The b-matrices were calculated numerically from the pulse sequence parameters as described in §4.4.3. Rotationally variant anisotropy indices were calculated, again on a pixel-by-pixel basis, using Eq. 5.1 and 5.2, to produce maps of the standard deviation (A_{sd}) and rotationally variant volume ratio (A_{vr}) indices.

$$A_{sd} = \frac{1}{\sqrt{6}ADC_{av}} \sum_{j=x,y,z} \sqrt{\left(ADC_j - ADC_{av}\right)^2} \qquad [5.1]$$

$$A_{vr} = \frac{ADC_x \times ADC_y \times ADC_z}{\left(ADC_{av}\right)^3} \qquad [5.2]$$

The physical interpretation of A_{sd} was described in §3.6.2, where the anisotropy is scaled between zero and one, for the cases of isotropic diffusion and anisotropic diffusion along an infinitely long, thin fibre, respectively. Similarly, A_{vr} can be interpreted physically because it represents the ratio of the volume of an ellipsoid with orthotropic axes defined by the three orthogonal ADCs to the volume of a sphere of radius equal to ADC_{av}. The scaling of A_{vr} is opposite to A_{sd}, with isotropic diffusion being assigned a value of one and high-order anisotropic diffusion a value of zero. The quantitative maps were calculated using the C program `dse`, described in §4.4.3. The diffusion-weighted image data sets were also aligned using SPM95, then ADC_{av} and anisotropy maps were recalculated, to investigate the effects of patient motion on the measured anisotropy. The ADC_{av} and anisotropy maps were displayed using ANALYZE, where regions of interest (ROIs) could be defined and measurements taken.

5.2.3 In vitro evaluation

The algorithms used to calculate the ADC_{av} and anisotropy maps for the patient studies were first tested in various phantom experiments, to verify that the algorithms were giving sensible information on the anisotropic behaviour of water in restricted media. The maps of ADC_{av} and anisotropy were calculated for a spherical water phantom, to ensure that the values obtained for the ADC were as expected and that the anisotropy indices indicated that the phantom was isotropic.

Noise is known to contaminate anisotropy images such that the measured anisotropy is increased above its true value, when the level of noise is significant (115,129). To evaluate the level of noise contamination in this study, the sequence was tested in vitro using isotropic and anisotropic phantoms, at various SNRs and then plots of anisotropy against SNR evaluated. The number of acquisitions were altered to vary the SNR, but no other sequence parameters were changed. The SNR was then measured from the patient images to establish whether it was sufficiently large such that noise contamination would be expected to have a minimal effect on the measured anisotropy. The SNR was calculated from the mean signal of the diffusion-weighted images in the region of interest $\bar{S}$ and the standard deviation of the background noise σ_n, measured from a region of interest of identical size containing background noise, using the relationship shown in Eq. 5.3,

$$SNR = 0.66\frac{\bar{S}}{\sigma_n}, \qquad [5.3]$$

where the factor 0.66 accounts for Rayleigh statistics (170).

5.2.4 In vivo patient studies

Patients admitted to hospital having suffered a mild stroke within the last 6 days were identified by a stroke physician. Some had a CT scan prior to MR (as CT is the routine examination for stroke in our hospital), while some others went straight to MR. The CT scans were utilised to exclude patients with haemorrhagic lesions from the study. A set of T_2-weighted whole brain images were obtained first, then the eight slices for DWI were set to cover the area of the brain likely to be involved from the symptoms, whether a relevant lesion was visible on the T_2 images or not.

ADC_{av} and anisotropy images were obtained from 12 patients (mean age 66 ± 14 years). The patients were obtained from a larger study of 36 patients selected using the criteria described above. Of those patients studied, 21 were shown to have hyperintense regions using DWI, and 12 showed images sufficiently free from artefact to enable ADC_{av} and anisotropy images to be calculated. ROI measurements were obtained from ischaemic and contralateral normal areas of the ADC_{av} and anisotropy images, both for the SPM aligned and unaligned cases. The ROIs were defined on an average diffusion-weighted image (DWI_{av}), constructed from the three orthogonal diffusion-weighted images and were positioned to include well defined hyper-intense regions. Highly dispersed hyper-intense regions containing large amounts of normal appearing tissue were excluded from the study.

5.3 Experimental findings and their implications

5.3.1 The effect of noise on the diffusion measurements

The measured ADC_{av} of the spherical water phantom at room temperature was $1.98 \pm 0.09 \times 10^{-3}$ mm^2s^{-1} and is in good agreement with literature values (67). The anisotropy indices, A_{sd} and A_{vr}, measured from the spherical water phantom are close to their ideal isotropic values. The standard deviation index was measured as 0.028 ± 0.006 (ideal isotropic case 0) and the volume ratio index was measured as 0.997 ± 0.002 (ideal isotropic case 1). Possible remaining sources of experimental error include residual noise contamination (as multiple acquisitions were not performed), thermal convection within the phantom, vibration and hardware instabilities. Allowing for the possible experimental errors, it would appear that the anisotropy indices do assign isotropic values to a homogeneous, isotropic medium as would be expected. If the measurements were contaminated by noise, the isotropic phantom would be expected to have a significantly elevated anisotropy.

Figure 5.1 shows the effect of noise on the two anisotropy indices, for a water phantom exhibiting isotropic diffusion and an anisotropic vegetable phantom. The anisotropic phantom used for this study was a leek, as this was found to produce ADC values similar to those in the human brain ($\approx 1.0 \times 10^{-3}$ mm^2s^{-1}), while also producing a degree of restriction that would be expected to conform with the axial

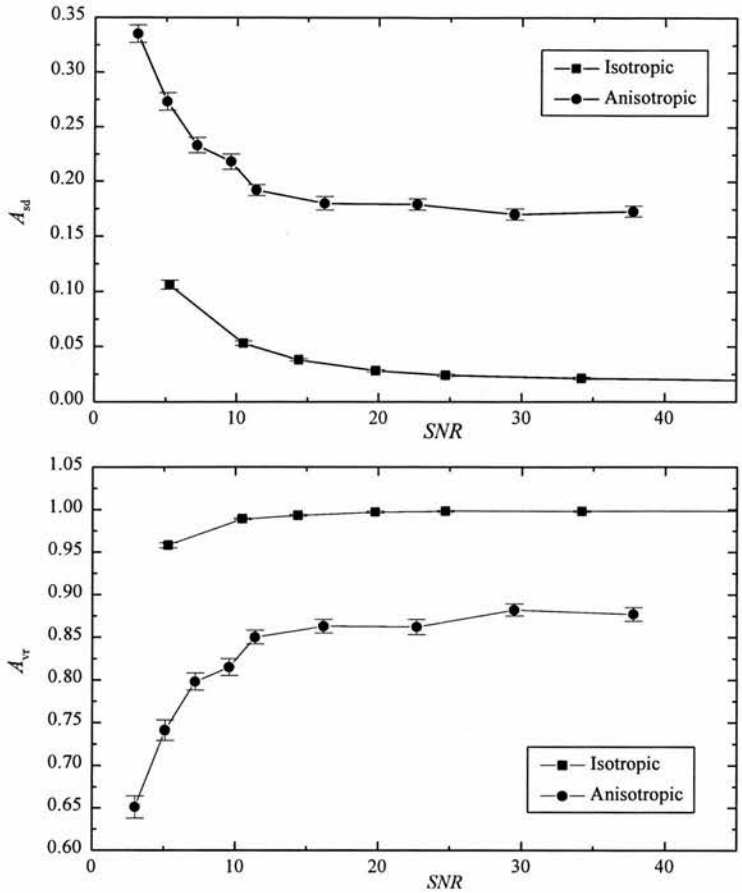


Figure 5.1 The variation of the two diffusion anisotropy indices with signal-to-noise ratio, *SNR*, for the standard deviation (A_{sd}) (top) and volume ratio (A_{vr}) (bottom) anisotropy indices. The data is illustrated for isotropic water and anisotropic vegetable phantoms.

symmetry of the vegetable. However, the restrictive properties of the vegetable are likely to be different to human tissues due to the impermeability of plant cells (69). The SNR was altered by acquiring up to 32 multiple acquisitions of each image and signal averaging. The figure shows the anisotropy to be increased as the SNR is decreased. This causes isotropic diffusion to be seen as anisotropic, while anisotropic diffusion is overestimated. It is clear from the graphs that the effect of the SNR on the measured anisotropy becomes greater as the SNR decreases. In this example, the measured anisotropy begins to increase rapidly at SNRs below about 20. At extremely low SNR (< 5), $tr(\boldsymbol{D})$ also became noticeably elevated. These findings are in good agreement with results obtained using Monte Carlo simulations (129).

The mean SNR of the patient diffusion-weighted images was calculated using Eq. 5.3 and found to be 36 ± 10 for normal appearing regions and 72 ± 21 for

ischaemic regions. Figure 5.1 would suggest that the SNR of the brain diffusion images, of around 36 for normal appearing regions and 72 for ischaemic regions, is sufficient such that the effects of noise on the calculated anisotropy will be small. This is because the anisotropy is only decreasing by negligible amounts if the SNR is further increased. The average measure of the SNR over the individual ischaemic and normal ROI measurements, rather than the SNR over the whole brain is considered because of the CSF present in the brain. CSF has a high diffusion coefficient and thus low signal in the diffusion-weighted images. However, the ROI measurements generally contained relatively small amounts of CSF and so the whole brain estimate of SNR would provide a conservative view of the SNR. Also, the SNR of an ischaemic region was always greater than that of the contralateral normal region and so this simple analysis would suggest that any small residual noise contamination would be more pronounced in the normal region. A residual contamination of this type would reduce the anisotropy contrast between stroke and normal regions.

However, the SNR analysis presented here is far too simplistic to provide reliable reassurances as to the accuracy of diffusion measurements in the presence of noise. The SNR of the diffusion-weighted images as has been used, conventionally, to describe noise contamination in diffusion experiments (115,129) has a high dependence on the imaging parameters used, particularly the diffusivity being measured. Therefore, the behaviour of the anisotropy with SNR observed in Fig. 5.1 cannot be directly transposed to another diffusion experiment with different experimental parameters, i.e. *b*-value, diffusivity, degree of anisotropy etc. This problem is discussed in more detail in Chapter 7, where a measure of noise is presented that is largely independent of experimental parameters.

5.3.2 Rotational variance

As discussed in §3.6.2, the orthogonal diffusion sampling strategy produces measurements of anisotropy that are rotationally variant (RV), i.e. that depend on the orientation of the sample relative to the orthogonal measurement axes.

$$\boldsymbol{b}_0 = \begin{pmatrix} 0.55 & 0.00 & -0.06 \\ 0.00 & 0.00 & 0.00 \\ -0.06 & 0.00 & 0.03 \end{pmatrix} \text{smm}^{-2} \qquad \boldsymbol{b}_x = \begin{pmatrix} 733.58 & 0.00 & -1.82 \\ 0.00 & 0.00 & 0.00 \\ -1.82 & 0.00 & 0.03 \end{pmatrix} \text{smm}^{-2}$$

$$\boldsymbol{b}_y = \begin{pmatrix} 0.55 & 16.65 & -0.06 \\ 16.65 & 699.73 & -1.75 \\ -0.06 & -1.75 & 0.03 \end{pmatrix} \text{smm}^{-2} \qquad \boldsymbol{b}_z = \begin{pmatrix} 0.55 & 0.00 & 16.59 \\ 0.00 & 0.00 & 0.00 \\ 16.59 & 0.00 & 696.25 \end{pmatrix} \text{smm}^{-2}$$

Figure 5.2 The *b*-matrices calculated for the baseline $\boldsymbol{b}_0$, and three orthogonal diffusion $\boldsymbol{b}_x$, $\boldsymbol{b}_y$ and $\boldsymbol{b}_z$ pulse sequences. The *b*-matrices were calculated numerically from Eq. 3.24 as described in §3.4.1.

Measurements of ADC_{av} obtained using the orthogonal imaging technique may also exhibit a small degree of rotational variance due to imaging and diffusion gradient interactions. These interactions are only fully accounted for by using the whole *b*-matrix and tensor formalism as in Eqs. 3.23 and 3.24. However, the off-diagonal elements of the *b*-matrix are small compared to the dominant diagonal element, as illustrated in Fig. 5.2. The error in ADC_{av} resulting from the use of Eq. 3.14a to evaluate the signal attenuation, rather than Eq. 3.23 is therefore negligible.

However, the rotational variance will introduce significant errors into anisotropy measurements obtained using the orthogonal technique, but the magnitude of these errors has rarely been considered. Pierpaoli and Basser, presented a series of anisotropy measurements obtained from the human brain using equivalent rotationally variant and invariant indices (115). They demonstrated that the anisotropy was considerably underestimated by using the orthogonal imaging technique. However, the RV measurements did show the general trends of increasing anisotropy that corresponded with the rotationally invariant (RI) measurements, except for the notable case of the anterior internal capsule, which was seen as isotropic using RV techniques but was highly anisotropic using the RI measure. This study suggested that the RV anisotropy could be used as a qualitative measure of anisotropy only, providing reduced anisotropy contrast and occasionally misrepresenting the anisotropy completely.

It has been previously demonstrated that using ipsilateral to contralateral ratios provides a method of measuring ADC changes in ischaemia that is independent of tissue orientation (168), although extreme care must be taken when positioning the ROIs if errors are not to be introduced. Similarly, due to the overall symmetry of the brain, anisotropy differences quantified using ipsilateral to contralateral ratios provide a means of measuring these changes that would also be largely independent of tissue orientation. So, while the absolute anisotropy may be underestimated using the three gradient orthogonal technique, using the stroke-to-normal ratios should be sufficient for identifying if the anisotropy is elevated or otherwise in the diseased state, provided that the contralateral side is also not diseased. However, this relies on the patients being identically reoriented in the scanner if serial comparisons are to be performed, a procedure that is difficult to achieve in a realistic clinical setting.

The rotational variance of the measured anisotropy obtained in this study, is likely to increase the variability between patients. Together with an underestimation of the anisotropy caused by not sampling the whole diffusion tensor, it would be recommended that the diffusion tensor technique be used for a fully quantitative analysis, particularly if multiple studies are to be performed. However, in using the three gradient orthogonal technique in a multi-patient study of sufficient size, the stroke-to-normal ratios would be expected to show equal amounts of increased and decreased anisotropy, if there was no underlying difference in anisotropy between the stroke and normal regions. Therefore, it would be expected that the orthogonal imaging technique could be used to evaluate changes in anisotropy between the normal and diseased states, however, the magnitude of these changes may not be accurately quantified.

5.3.3 The reliability of the navigator motion correction in vivo

The effectiveness of the three navigator phase correction algorithms; zero order; linear zero and first order; and independent zero and first order; were observed for the first 26 patients enrolled into the study. The effectiveness of these algorithms was assessed qualitatively by observing whether the corrected images were sufficiently artefact free to enable a quantitative evaluation to be made. The

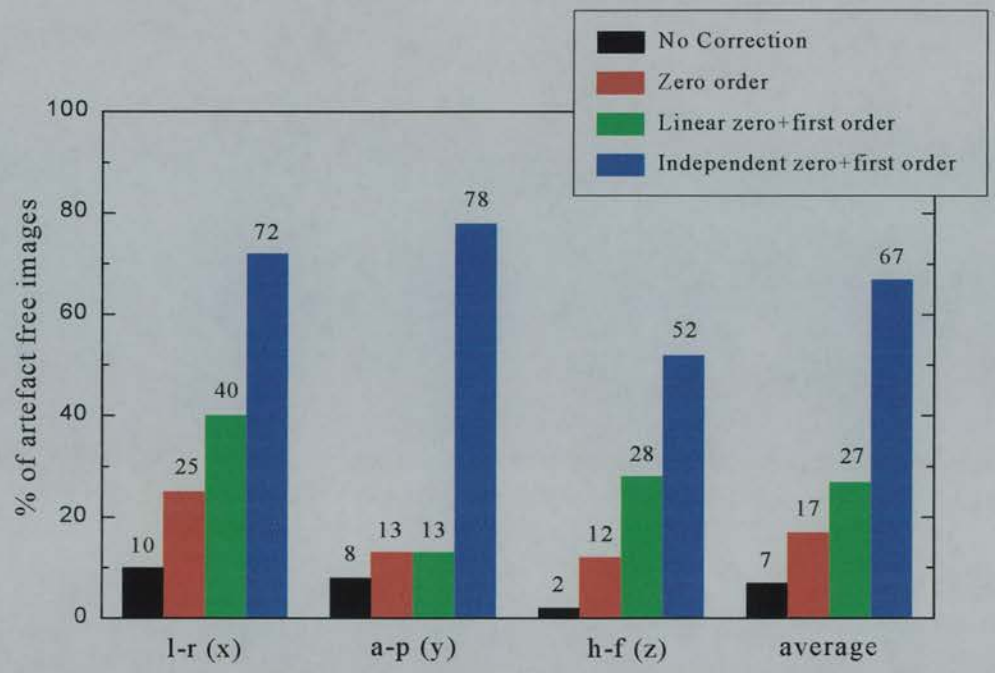


Figure 5.3 The Percentage of diffusion-weighted images judged to be acceptably artefact free after motion correction performed using the different navigator correction algorithms. Diffusion gradients were applied in the left-right (l-r), anterior-posterior (a-p) and head-feet (h-f) directions.

assessment was performed for all images and tabulated separately for the sequences with diffusion gradients applied along the three orthogonal directions. The results are summarised by the bar chart representation in Fig. 5.3.

This study demonstrated that the effectiveness of the navigator correction is extremely dependent on the motion correction algorithm used. With no correction applied, only 7% of the DWIs were artefact free, compared to 17% for the zero-order correction, 27% for the linear zero and first-order correction and 67% for the independent zero and first-order method. Typical examples of images corrected using the navigator technique are shown in Fig. 5.4. There was a significant difference between the success rate of the latter method depending on whether the diffusion gradient was applied along the phase encoding direction (G_x or G_y) or not (G_z). This result is anticipated from theoretical considerations because the navigator echo can only be used to correct for rotational motion when the diffusion gradient is applied along the phase encode direction (91), as discussed in §3.5.1.

It is interesting to observe the relative differences in the effectiveness of the different motion compensation algorithms. The algorithm performing zero-order

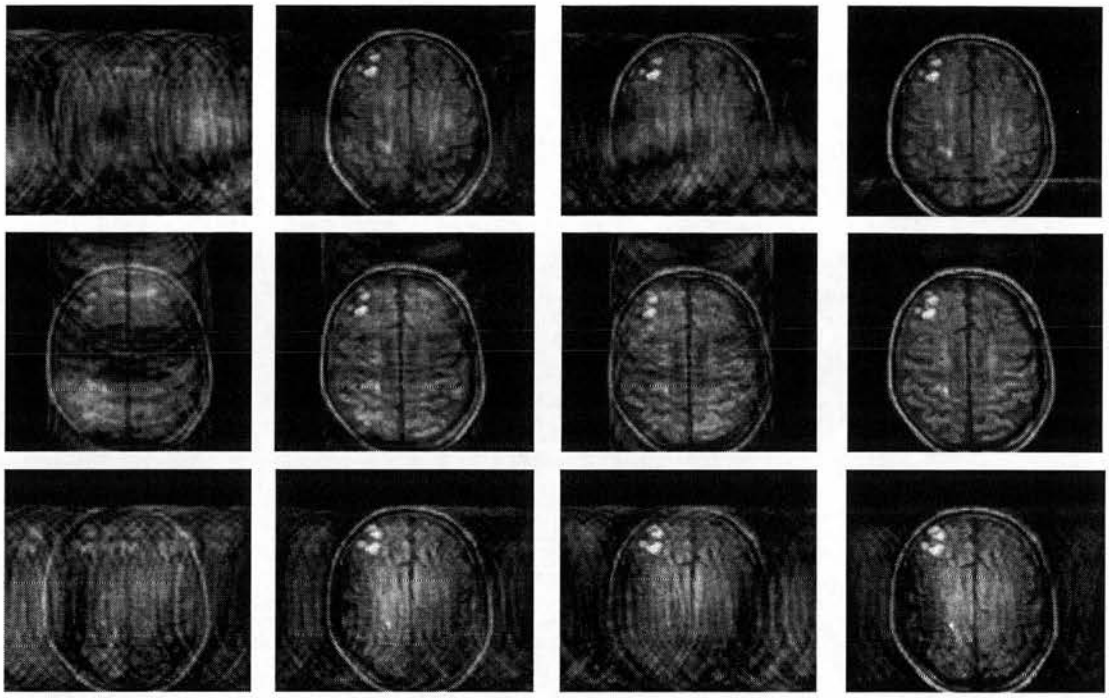


Figure 5.4 Examples of typical images corrected for motion artefacts using navigator echo methods. The top, middle and bottom rows show images with diffusion gradients applied along the l-r. a-p and h-f directions, respectively.

phase corrections, to allow for translational motion, results in only a marginal 10% improvement in the number of acceptable images. Similarly, the algorithm assuming zero and linear first-order phase changes, to correct for both rotational and translational rigid body motion, only provides a further 10% improvement in the number of acceptable images. The greatest improvement occurs when independent phase angles are calculated for each point along the navigator projection, as opposed to fitting a linear phase variation along the projection. This process effectively corrects for non-rigid body motion and results in an improvement of 60% compared to the standard non-corrected images. These results would suggest that non-rigid body motion provides the bulk of the motion artefact arising in diffusion-weighted spin-echo images, possibly arising from pulsatile brain motion.

Even after non-rigid body motion navigator correction has been performed, some residual motion artefacts exist at a substantial enough level to result in only about two-thirds of all images being artefact free. Even allowing for the limitations

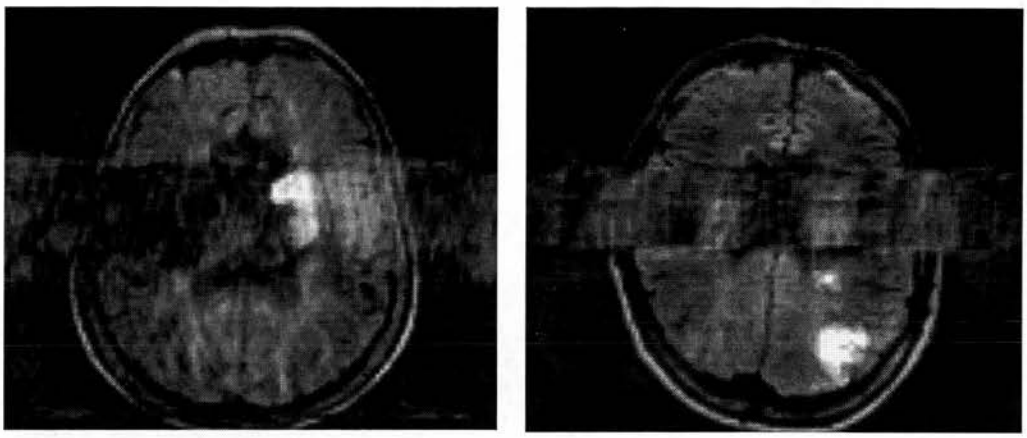


Figure 5.5 Illustration of the failure of the motion algorithm to correct for motion around the ventricles. This is likely to be due to the complex pulsatile motion in the brain and the low SNR of the navigator profile.

of the navigator correction when diffusion gradients are applied in non-phase encoding directions, the algorithm only corrects about 75% of images. The residual image artefacts may result from rotation angles that exceed the small angle approximations implicit in the correction algorithms. Also, for patients moving rapidly, or in the case of pulsatile brain motion, the implicit assumption that the form of motion is the same when both the navigator and image echoes are acquired may not be adequate. Another problem may result from noisy navigator profiles, which could result in a poor estimation of the motion induced phase errors. An attempt was made to overcome the problem of spurious phase errors, by applying smoothing functions to the navigator profiles. However, while these methods occasionally improved the visual appearance of an image, the effect was not substantial enough to reclassify poor images. Figure 5.5 demonstrates an example of an image corrected using the independent zero and first-order method which still shows significant residual blurring at the level of the ventricles. As discussed above, this blurring may result either from complex pulsatile motion occurring between the image and navigator echoes, or from noise contamination of these profiles resulting from the low signal of the CSF in the diffusion-weighted images.

The navigator echo correction procedure provides an obvious improvement in image quality, with a 97% probability that at least one of the three diffusion images will be artefact free for any given slice, almost guaranteeing the diagnostic usefulness

of the technique. However, there is only a 29% chance that all three diffusion images will be artefact free for any given slice. This means that the likelihood of being able to perform a quantitative evaluation using the orthogonal technique is considerably less. Therefore, quantitative diffusion measurements are only realistically achievable on a small number of slices for any given patient, making measurements encompassing the whole of an infarct extremely unlikely to be obtained. Indeed, it is quite conceivable that for a given patient, no lesion containing slices will be suitably free from artefact in all three directions. This was found to be the case in this study, where the minimum criterion of being able to measure at least one lesion containing slice only occurred in 12 out of 21 patients. This problem would be even worse if DTI was performed using the navigator technique, then six DWIs would be required to be artefact free for any given slice. If it is assumed, somewhat optimistically, that the average success rate of 67% for each image is maintained, then there is only a 9% chance of obtaining the six artefact free images for any given slice! The reality is worse because the tensor acquisition schemes require diffusion gradients to be applied along directions orthogonal to the phase encoding direction. Therefore, even if the scan time could be tolerated ($\approx$ 35 minutes), it is unlikely that enough artefact free images would be obtained to make the technique useful.

5.3.4 Image alignment and quantitative in vivo measurements

The twelve patients showed a mix of grey and white matter lesions, as classified by a neuroradiologist, with four patients having 'pure' white matter lesions, seven having mixed white and grey matter and one having a 'pure' grey matter lesion. On average, patients underwent imaging 4.0 $\pm$ 3.8 days after onset of stroke symptoms. Typical examples of the images and quantitative diffusion maps are shown in Fig. 5.6. ROI measurements, totalling 32, were obtained from the 12 patients, either from the same lesion extending into multiple slices or from a patient having multiple recent lesions. ROI measurements varied in size from 3 to 62 pixels (mean 18 $\pm$ 17); however no significant dependence was found between the size of the infarct and the measured ADC_{av} or anisotropy, as illustrated in Fig. 5.7. The marginally increased anisotropy for smaller lesions may well result from the

		Stroke Region	Normal Region	Stroke/Normal Ratio
ADC_{av}	Unaligned	$0.76 \pm 0.34^{\dagger}$	$1.14 \pm 0.45^{\dagger}$	0.75 ± 0.40
	Aligned	$0.60 \pm 0.23^{\dagger}$	$1.15 \pm 0.45^{\dagger}$	0.56 ± 0.23
A_{sd}	Unaligned	0.36 ± 0.23	0.23 ± 0.17	1.90 ± 2.38
	Aligned	0.27 ± 0.11	0.18 ± 0.11	1.78 ± 1.53
A_{vr}	Unaligned	0.70 ± 0.23	0.83 ± 0.17	0.90 ± 0.45
	Aligned	0.74 ± 0.23	0.86 ± 0.17	0.87 ± 0.28

$^{\dagger}ADC_{av}$ in $10^{-3}mm^2s^{-1}$

Table 5.1 Average ROI measurements obtained from stroke and contralateral normal areas of ADC_{av} and diffusion anisotropy maps from 12 patients with small cortical or lacunar infarcts. Results are shown for images calculated from unaligned and aligned sets of diffusion-weighted images.

increased proportion of border pixels, which are likely to be more susceptible to increased anisotropy due to image distortions and misalignment. The mean SNR of the patient diffusion-weighted images was found to be 36 ± 17 for normal appearing regions and 72 ± 34 for ischaemic regions, calculated using Eq. 5.3. The mean values for ADC_{av}, A_{vr} and A_{sd} are shown in Table 5.1, for the 12 patients. The values are shown for measurements taken from the SPM aligned and unaligned images.

One-sample t-tests of the mean stroke to normal ratios for the ADC and anisotropy measurements were performed. The ADC_{av} ratio gave 0.75 ± 0.40 ($p = 0.0008$) for the unaligned data and 0.56 ± 0.23 ($p < 0.0001$) for the aligned data. Similarly, the values for the A_{sd} test yielded 1.90 ± 2.38 ($p = 0.02$) for the unaligned data and 1.78 ± 1.53 ($p = 0.003$) for the aligned data. The A_{vr} test gave 0.90 ± 0.45 ($p = 0.12$) for the unregistered data and 0.87 ± 0.28 ($p = 0.005$) for the registered data.

The patient data, summarised in Table 5.1, suggest that aligning the original diffusion-weighted images influences the measurements obtained of ADC_{av} and anisotropy. The anisotropy appears to be overestimated when the unaligned images are used. This is not entirely surprising as in regions where two different media intersect, a small misregistration could cause the intersection to be seen as a highly anisotropic region. This is likely to be particularly noticeable in this study where many of the ROIs are small in size. ADC_{av} is also noticeably higher in the unaligned

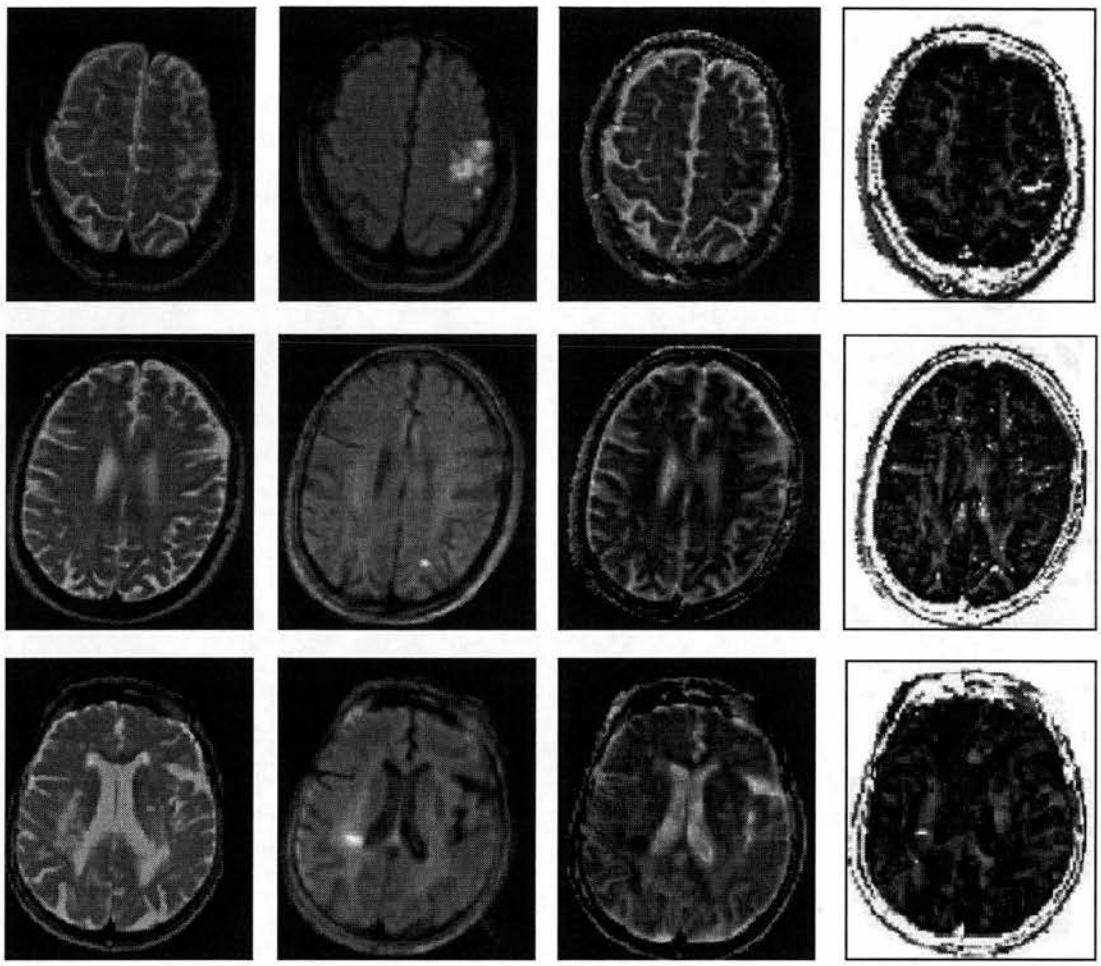


Figure 5.6 Examples of T_2-weighted (left); DWI_{av} (left middle); ADC_{av} (right middle); and A_{sd} (right); images for; a 38 year old patient presenting with symptoms of a left-sided partial anterior circulation infarct (PACI) (top); a 66 year old patient with symptoms of a left-sided PACI (middle); and a 87 year old patient with a right sided lacunar infarct and also a left sided haemorrhage (bottom).

stroke regions, this is probably due to misregistered diffusion-weighted images causing some normal tissue to be included in the ischaemic ROI therefore increasing the measured ADC. Mean translations of 3.2 ± 1.9 mm and mean rotations of 4.6 ± 3.9° were obtained from SPM alignment of the 12 patients, suggesting a possible misregistration of 1 or 2 pixels between diffusion-weighted images. So, while the navigator echo can correct for rotations and translations of this order during the image acquisition, the mean position of the image may be shifted relative to the other DWIs in the data set, and thus require further alignment before calculating ADC_{av} and anisotropy images.

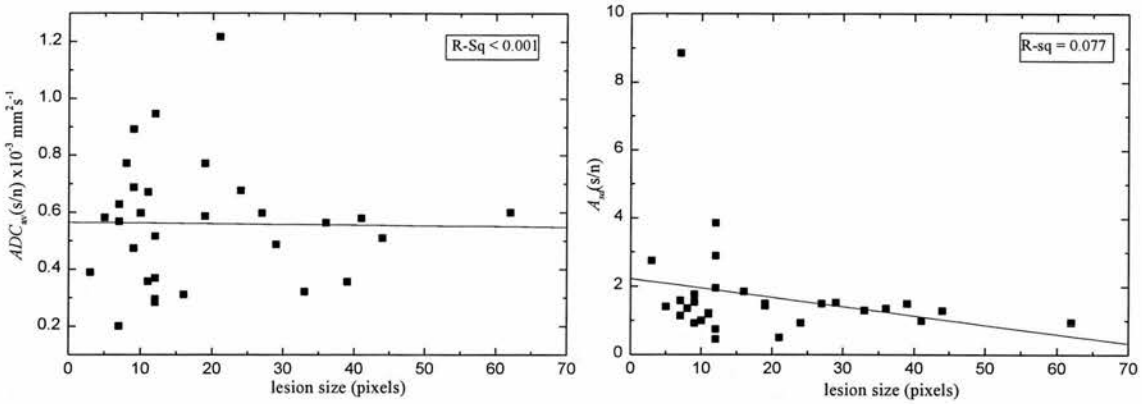


Figure 5.7 The correlation between the size of the infarct with (a) ADC_{av} and (b) A_{sd}. The poor correlation is indicated by the low R-Sq fitting paramter obtained from linear regression.

The effect of these rotations and translations on the visual appearance of the images after navigator correction is illustrated in Fig. 5.8 for the 12 patients studied quantitatively. For each patient, all constituent DW images were scored on a scale ranging from 1 for totally unusable images to 10 for images with no visible artefacts. The totals were scored for each patient and an average taken to give an 'artefact score' for that patient. The rotation and translation required to realign the images in the slice used for the ROI measurement was then obtained using SPM95, to give an indication as to the degree of patient movement during the scan period. The SPM parameters were calculated from the ROI slices because they were free from artefact. Using all of the slices to estimate the motion would have provided useless information, due to the spurious rotations and translations obtained from attempting to align artefactual images. The results show a general trend of increased artefact for patients who appeared to move more between scans. The correlation is also greater for patients undergoing translational, rather than rotational motion, a result which is somewhat surprising, as the theoretical limitations of the navigator echo chiefly involve the correction of rotational motion.

The ADC measurements obtained from the 12 patients using the aligned images suggest an ADC_{av} decrease of about 40% in the ischaemic regions which agrees with previous studies of ADC changes in acute stroke (21,67). It is clear from the data that the ADC_{av} decrease and anisotropy increase are highly significant ($p << 0.05$) when the aligned images are used. The changes are much less significant if the calculated images are not obtained from an aligned set of diffusion-weighted images

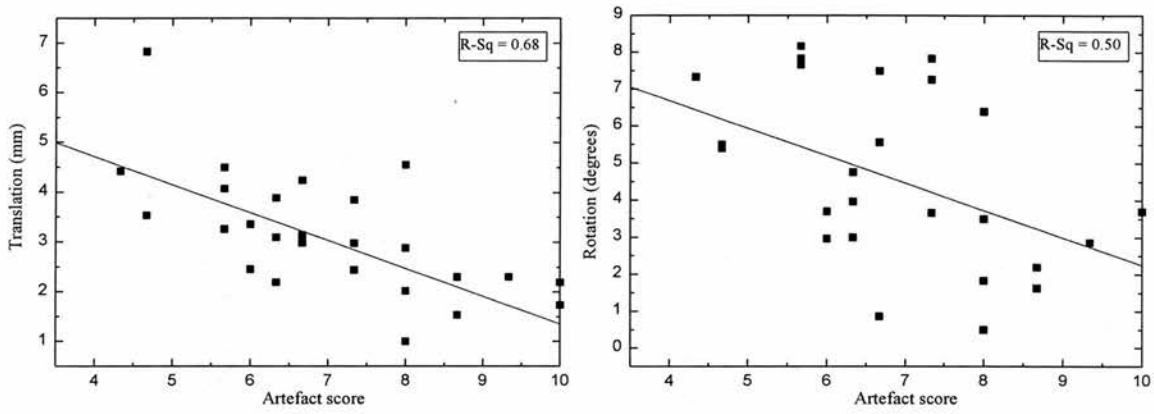


Figure 5.8 The correlation between the degree of motion artefact measured using an 'artefact score' ranging from 1 (unusable) - 10 (no visible artefact) and the measured rotation and translation of the patient during the scan.

This demonstrates the importance of image alignment when calculating ADC and anisotropy maps, if quantitative results are to be obtained. This would apply even for rapidly-acquired images (e.g. EPI) because patient motion between scans may still occur.

A further complication to the process of image alignment arises because of the phase encoding direction being swapped, to improve the effectiveness of the navigator correction for the two in-plane diffusion measurements. The chemical shift artefact between water and fat is prominent along the frequency encoding direction, due to the application of the readout gradient during acquisition. Therefore, the water-fat shift will be dependent on the direction of the frequency encoding gradient and so images obtained with diffusion gradients applied along the left-right and anterior-posterior directions will have different chemical shifts (as the phase/frequency encoding direction is swapped between these two acquisitions). This effect can be observed by looking at the shift between the brain tissue and the fat surrounding the scalp. Figure 5.9 illustrates this shift for the images obtained with diffusion gradients applied along the left-right (l-r), anterior-posterior (a-p) and head-feet (h-f) directions. The phase encoding direction is horizontal on the l-r and h-f images, but vertical on the a-p images. This small shift of around a pixel may result in alignment errors because the images are no longer identical. However, this error is likely to be small compared to other image artefacts due to motion.

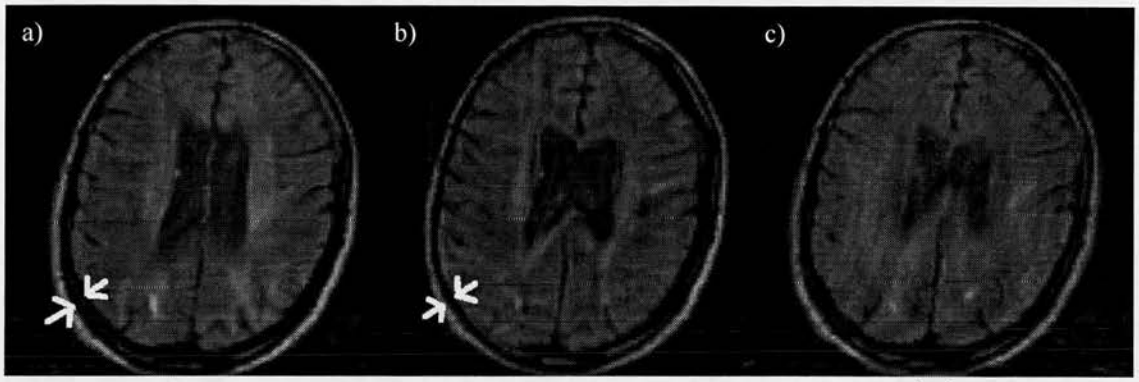


Figure 5.9 The differential fat-water chemical shift arising from swapping the phase encode direction. Images (a) and (c) have the phase encoding along the horizontal direction, while image (b) has the phase encoding along the vertical direction. The direction of the diffusion gradient is indicated below each figure.

The increased anisotropy measured in this study seems to support the idea of the cell swelling in cytotoxic oedema causing an increased restriction of the extracellular water, as described in §3.7.1, which would be accompanied by a reduced ADC and an increase in diffusion anisotropy. The biophysical mechanism for the anisotropy increase is uncertain, but a possible explanation arises if an arrangement of aligned fibre bundles are considered, where diffusion occurs preferentially along the fibre direction. Swelling of the fibres perpendicular to their axis will lead to a reduced space between the fibres. This will increase the restriction of the extracellular water, transverse to the fibre direction, thus increasing the measured anisotropy (126). Diffusion imaging shows great promise for enabling the extent of this swelling to be observed in vivo by measuring the changes of ADC_{av} and diffusion anisotropy. The large variability in ADC and anisotropy between patients can be assigned to the different tissue structures and the degree of cell swelling, suggesting that classification of infarcts into pure white matter and mixed grey/white matter may be more revealing, for a patient group of sufficient size. This is discussed in more detail in Chapter 9.

5.4 Conclusions

Navigated spin-echo diffusion imaging was performed on 36 patients, presenting with symptoms of acute ischaemic stroke (1 - 6 days post-ictus) and quantification was undertaken on 12 of the patients. ADC_{av} measurements showed a significant decrease of around 40% in acute ischaemic regions. Diffusion anisotropy changes were also demonstrated and were significantly increased in the lesions. The methodology was assessed in terms of its likelihood of producing reliable quantitative measurements. The effects of rotational variance were considered and some of the limitations of the orthogonal sampling technique were discussed. Experimental noise was demonstrated to have a significant effect on the measured anisotropy and a simple attempt was made to account for noise contamination in the images. The limitations of the navigator correction procedure were also considered, with only around a 30% chance of obtaining sufficiently artefact free images to attempt a quantitative analysis. The results suggest that only a limited quantitative evaluation of diffusion is possible in the clinical setting, using the low specification hardware and methodology described. However, as a diagnostic, non-quantitative tool, the technique was demonstrated to be extremely successful. The following chapters attempt to address the problems hindering the reliable acquisition of quantitative diffusion images, in particular, the degrading effects of rotational variance and experimental noise.

6

The Limitations of Using Spherical or Cylindrical Diffusion Models

6.1 Introduction

Some of the potential problems arising from using the orthogonal diffusion acquisition scheme were discussed in Chapter 5. One of the principal limitations of this technique is its susceptibility to sample orientation, so called rotational variance (115). This arises chiefly because the model assumes that the diffusion environment is spherically symmetric. A further acquisition scheme has been recently proposed based on the application of a tetrahedral arrangement of diffusion gradients. This method is derived using a mathematical diffusion formalism that assumes cylindrical symmetry (85). This model was suggested because many of the regions studied in neuro-imaging are believed to exhibit cylindrical symmetry (128,171). The reasons for using these simplified models, rather than sampling the full diffusion tensor, is that they allow for the collection of a reduced number of images. This is an important consideration in the clinical setting, where scanner time is valuable and sick patients may be unable to co-operate for long periods. The aim of this chapter is to discuss the limitations of these techniques for performing accurate diffusion quantification in realistic situations.

To describe fully a general diffusion situation, the entire diffusion tensor ($\boldsymbol{D}$) must be sampled, requiring at least seven non-collinear diffusion measurements,

enabling the six independent elements of the symmetric diffusion tensor and the T_2-weighted baseline signal intensity to be evaluated (24). The orthogonal encoding technique samples the three diagonal elements of the diffusion tensor (D_{xx}, D_{yy}, D_{zz}), which enables a full characterisation of the diffusion ellipsoid only when the diffusion is isotropic or when the measurement and principal diffusion axes are aligned (115). The tetrahedral encoding scheme samples the diffusion environment using four diffusion gradient vectors in a tetrahedral arrangement, which enables calculation of $tr(\boldsymbol{D})$ and the three independent off-diagonal elements. This allows for a complete description of diffusion when the diffusing spins are in an axisymmetric environment (85).

Computer simulations were performed to evaluate the effectiveness of the orthogonal and tetrahedral schemes at characterising adequately the diffusion environment. The simulations were performed such that the user defines a diffusion ellipsoid (86) and its orientation with respect to the measurement axes. The 'measured' diffusion tensor is then calculated, enabling the orthogonal and tetrahedral diffusion measurements to be obtained and compared to the tensor analysis. Only the theoretical drawbacks of the different encoding schemes and not the practical implications for implementing them are discussed. Therefore, the values entered into the simulation are diffusivities that are assumed to be reliably measured, with b-values corrected for all gradient interactions, using the correct models of diffusion, and with no measurement noise.

6.2 Theoretical Considerations

6.2.1 The diffusion ellipsoid and coordinate transformations

The concept of the diffusion ellipsoid was introduced in §3.4.3 to provide an easily interpretable description of the diffusion environment in a voxel, where the distance from the origin to a point on the diffusion ellipsoid represents a scaled diffusion measurement in that direction (86). A full description of the symmetric diffusion tensor requires the six independent elements (D_{xx}, D_{yy}, D_{zz}, D_{xy}, D_{xz}, D_{yz}) to be sampled (83,172). This is tantamount to sampling enough points on the ellipsoid to characterise fully its size, shape and orientation. Clearly, in the case of isotropic

diffusion, one measurement alone (the radius) is enough to describe fully the diffusion sphere. Also, for axisymmetric ellipsoids, four measurements are sufficient to describe the ellipsoid and so characterise fully the diffusion. However, it is unlikely that the characteristics of the diffusion ellipsoid and its orientation will be known *a priori*, thus extensive tests may need to be carried out before models assuming spherical or cylindrical symmetry could be used reliably in a realistic situation, particularly if distortions due to noise are considered.

The relationship between the diffusion tensor obtained in the measurement (or magnet) and the principal diffusion coordinate systems was given in §3.4.3. These coordinate transformations enable the principal diffusivities to be obtained from the measured tensor, a process that involves calculating its eigenvalues and eigenvectors. The orthogonal imaging technique assumes that all off-diagonal elements are zero, i.e. that the diagonal elements are the principal diffusivities, while the tetrahedral technique assumes that the eigenvalues of the tensor are cylindrically symmetric i.e. $\lambda_1 \neq \lambda_2 = \lambda_3$. The methods required to extract the diffusion information from the measurements are therefore somewhat different for the different acquisition methods and are described below.

6.2.2 Diffusion in a general anisotropic environment

The tensor diffusion acquisition scheme involves the application of at least seven gradient vectors enabling the most general diffusion case to be sampled as described in §3.4. The gradients were conventionally applied along vectors (1 0 0), (0 1 0), (0 0 1), (1 1 0), (1 0 1) and (0 1 1), along with a baseline image with no diffusion gradients applied. Many other gradient configurations have since been suggested to give more efficient sampling of the ellipsoid (83,173). The diffusion tensor can be derived from the seven diffusion measurements by using a multivariate linear regression algorithm (24). The matrix rotation given by Eqs. 3.31-4 is then performed to evaluate the principal diffusion coefficients and directions, a process which is equivalent to calculating the eigenvalues and eigenvectors of $\boldsymbol{D}$. Thus, the six measurements performed in the tensor encoding scheme enable the measurement of all six independent elements of the diffusion tensor.

The diffusion environment can then be characterised using the various scalar invariant quantities obtained from the matrix operations suggested in §3.6.2. These scalar quantities provide rotationally invariant, physically meaningful 'stains' which describe the magnitude of diffusion, the degree of anisotropy and the level of fibre tract organisation (84). The general diffusion situation can be described fully when all six tensor elements are known, enabling the eigenvalues and eigenvectors which characterise the diffusion ellipsoid to be obtained. An alternative mathematical approach to characterising the diffusion ellipsoid from the measured tensor was recently proposed (85). This procedure involved using matrix decomposition to separate the reduced diffusion tensor (Λ) into isotropic and anisotropic components, such that

$$\Lambda = \bar{\lambda}\boldsymbol{I} + \eta\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 2 \end{pmatrix} + \varepsilon\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad [6.1]$$

where Λ is given by Eq. 3.32, η is the major anisotropy element, ε is the minor anisotropy element and $\boldsymbol{I}$ is the identity matrix. The quantities $tr(\boldsymbol{D}) = \bar{\lambda}$, η and ε are all invariant to rotation and characterise the size and shape of the diffusion ellipsoid. The anisotropy elements η and ε are calculated from Eq. 6.1 such that

$$\eta = \frac{1}{3}\left[\lambda_{z'''} - \frac{\left(\lambda_{x'''} + \lambda_{y'''}\right)}{2}\right] \qquad [6.2]$$

$$\varepsilon = \frac{\lambda_{x'''} - \lambda_{y'''}}{2} \qquad [6.3]$$

where the coordinate system (x''', y''', z''') represents the principal diffusion coordinate system, as in §3.4.3. Therefore, the size and shape of the diffusion ellipsoid, in the most general case, can be characterised by the parameters $\bar{\lambda}$, η and ε, while its orientation is given by the three Euler angles ϕ, θ and ψ obtained from Eq. 3.34. Several different notations have been used in the literature to represent the principal diffusivities, with (λ_1, λ_2, λ_3) generally representing the principal diffusivities ordered by magnitude, while the representation ($\lambda_{x'''}$, $\lambda_{y'''}$, $\lambda_{z'''}$) describes principal diffusivities that correspond to the magnet (x, y, z) system and are not necessarily ordered

consistently after the coordinate rotation in Eqs. 3.31 and 3.34 has been applied (see §3.4.3).

A complete description of the diffusion ellipsoid enables many rotationally invariant anisotropy indices to be derived. The standard deviation (A_{sd}) and volume ratio (A_{vr}) indices are two common examples, utilising the full information content of the diffusion tensor to provide rotationally invariant measures (115), given by

$$A_{sd} = \frac{1}{\sqrt{6\bar{\lambda}}} \sum_{i=x''',y''',z'''} \sqrt{\left(\lambda_i - \bar{\lambda}\right)^2} \qquad [6.4a]$$

$$= \frac{1}{\sqrt{6}D_{av}} \sum_{i=x,y,z} \sqrt{\left(D_{ii} - D_{av}\right)^2 + 2\left(D_{xy}^2 + D_{xz}^2 + D_{yz}^2\right)} \qquad [6.4b]$$

$$A_{vr} = \frac{\lambda_{x'''} \times \lambda_{y'''} \times \lambda_{z'''}}{\left(\bar{\lambda}\right)^3}. \qquad [6.5]$$

The definition of the standard deviation in Eq. [6.4b] is particularly useful as it does not require the calculation of the eigenvalues (85). The major and minor anisotropy indices (A_{major} and A_{minor}) can be obtained from η and ε, by dividing by $\bar{\lambda}$, as shown by Conturo *et al.* (85). A_{major} is the axial anisotropy and shows the deviation between the axial and average non-axial diffusion coefficients and A_{minor} is the equatorial anisotropy which shows the deviation between the two minor or equatorial diffusivities.

6.2.3 Diffusion in a spherically symmetric environment

The orthogonal diffusion encoding technique involves three separate diffusion measurements, obtained by applying diffusion-encoding gradients along orthogonal x, y and z axes in the magnet coordinate system. This technique obtains estimates of D_{xx}, D_{yy} and D_{zz}, from which the trace of the diffusion tensor can be calculated

$$\frac{1}{3}tr(\boldsymbol{D}) = \frac{D_{xx} + D_{yy} + D_{zz}}{3} = D_{av}. \qquad [6.6]$$

In this expression, it is assumed that the three orthogonal measurements ADC_x, ADC_y and ADC_z are accurately measured, with gradient interactions being accounted for, then $ADC_x = D_{xx}$, $ADC_y = D_{yy}$ and $ADC_z = D_{zz}$ and the quantity D_{av} is invariant upon

rotation of the coordinate system (86). It is clear that this technique does not sample any of the off-diagonal tensor elements and so D_{xx}, D_{yy} and D_{zz} only provide reliable estimates of the individual principal diffusivities when the measurement and principal diffusion axes are aligned. Obviously, in the case of isotropic diffusion, when there is no preferred principal diffusion direction, diffusion can be described completely using the orthogonal imaging technique (with surplus information as only one of D_{xx}, D_{yy} and D_{zz} is required), again provided that the diffusivities are accurately measured. A number of diffusion anisotropy indices have been derived from the orthogonal technique, all of which provide a measure of the variation of the diagonal elements of the diffusion tensor. These DAIs are rotationally variant and depend upon the orientation of the principal diffusion and measurement axes. The standard deviation index (A_{sd}) (21) and the volume ratio index (A_{vr}) (115), were used to provide anisotropy measurements for the study in Chapter 5 and are shown below

$$A_{sd} = \frac{1}{\sqrt{6}D_{av}} \sum_{i=x,y,z} \sqrt{\left(D_{ii} - D_{av}\right)^2}, \qquad [6.7]$$

$$A_{vr} = \frac{D_{xx} \times D_{yy} \times D_{zz}}{\left(D_{av}\right)^3}. \qquad [6.8]$$

It is clear that Eqs. 6.7 and 6.8 will only provide an identical anisotropy measurement to the rotationally invariant equivalents in Eqs. 6.4 and 6.5, when $\lambda_{x''} = D_{xx}$, $\lambda_{y''} = D_{yy}$ and $\lambda_{z''} = D_{zz}$, i.e. when the measurement and principal diffusion coordinate systems are aligned.

6.2.4 Diffusion in an axisymmetric environment

The tetrahedral encoding technique involves four simultaneous applications of diffusion gradients along the x, y and z axes. The gradients are applied in a tetrahedral arrangement, one each along vectors (1 1 1), (-1 -1 1), (1 -1 -1) and (-1 1 -1). The four acquisitions, $D^{(1)}$, $D^{(2)}$, $D^{(3)}$ and $D^{(4)}$, allow the measurement of the off-diagonal tensor elements D_{xy}, D_{xz}, D_{yz} and the average diffusion coefficient, D_{av}, which is rotationally invariant and equivalent to the trace of the diffusion tensor, again provided that there are no interactions between imaging and diffusion gradients (85). These four parameters allow the diffusion tensor (or ellipsoid surface) to be

described fully, subject to the diffusion being axisymmetric. So, for an axisymmetric diffusion environment, the four diffusion measurements enable D_{av}, η, θ and ϕ to be determined, which completely describe the axisymmetric diffusion ellipsoid (85). Only four parameters are required to characterise the axisymmetric ellipsoid because the third Euler rotation ψ is redundant due to the axial symmetry (see Fig. 3.2) and the value of ε is zero because the two minor elements, $\lambda_{x''}$ and $\lambda_{y''}$, must be the same.

To enable the parameters η, θ and ϕ to be determined from the off-diagonal elements, the matrix operation described in Eq. 3.31 must be performed with Λ in its decomposed form i.e. separated into isotropic and anisotropic components (85). By setting $\varepsilon = 0$, expressions can then be derived for η, θ and ϕ, as follows

$$\boldsymbol{D} = \boldsymbol{E}\Lambda\boldsymbol{E}^T = \boldsymbol{R}^T\Lambda\boldsymbol{R} = \bar{\lambda}\left(\boldsymbol{R}^T\boldsymbol{I}\boldsymbol{R}\right) + \eta\boldsymbol{R}^T\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 2 \end{pmatrix}\boldsymbol{R} \qquad [6.9],$$

where $\boldsymbol{R}^T\boldsymbol{I}\boldsymbol{R} = \boldsymbol{I}$ because $\boldsymbol{R}$ is an orthogonal matrix. Performing the transformation on the second term then yields

$$\boldsymbol{D} = \bar{\lambda}\boldsymbol{I} + \eta\begin{pmatrix} 3\cos^2\phi\sin^2\theta - 1 & 3\cos\phi\sin\phi\sin^2\theta & 3\cos\phi\cos\theta\sin\theta \\ 3\cos\phi\sin\phi\sin^2\theta & 3\sin^2\phi\sin^2\theta - 1 & 3\sin\phi\cos\theta\sin\theta \\ 3\cos\phi\cos\theta\sin\theta & 3\sin\phi\cos\theta\sin\theta & 3\cos^2\theta - 1 \end{pmatrix}. \qquad [6.10]$$

Notice that ψ does not appear in the expression for $\boldsymbol{D}$ above and thus is redundant for axial diffusion. The off-diagonal elements can then be calculated from Eq. 6.10, such that

$$D_{xy} = 3\eta\cos\phi\sin\phi\sin^2\theta \qquad [6.11]$$

$$D_{xz} = 3\eta\cos\phi\cos\theta\sin\theta \qquad [6.12]$$

$$D_{yz} = 3\eta\sin\phi\cos\theta\sin\theta \qquad [6.13]$$

and these equations can be solved to obtain expressions for η, θ and ϕ.

$$\eta = \frac{1}{3}\left[\frac{D_{xy}D_{xz}}{D_{yz}} + \frac{D_{xy}D_{yz}}{D_{xz}} + \frac{D_{xz}D_{yz}}{D_{xy}}\right] \qquad [6.14]$$

$$\theta = \tan^{-1}\left(\frac{D_{xy}\sqrt{D_{xz}^2 + D_{yz}^2}}{D_{xz}D_{yz}}\right) \qquad [6.15]$$

$$\phi = \tan^{-1}\left(\frac{D_{yz}}{D_{xz}}\right). \qquad [6.16]$$

From these parameters the diffusion parallel ($D_{//}$) and perpendicular ($D_{\perp}$) to the principal diffusion axis can be determined from Eq. 6.1, such that

$$D_{//} = \lambda_{z'''} = D_{av} + 2\eta \qquad [6.17]$$

$$D_{\perp} = \lambda_{x'''} = \lambda_{y'''} = D_{av} - \eta \qquad [6.18]$$

The tetrahedral anisotropy A_{tet} is then evaluated, such that

$$A_{tet} = \frac{\eta}{D_{av}} = \frac{D_{//} - D_{\perp}}{D_{//} + 2D_{\perp}} \qquad [6.19]$$

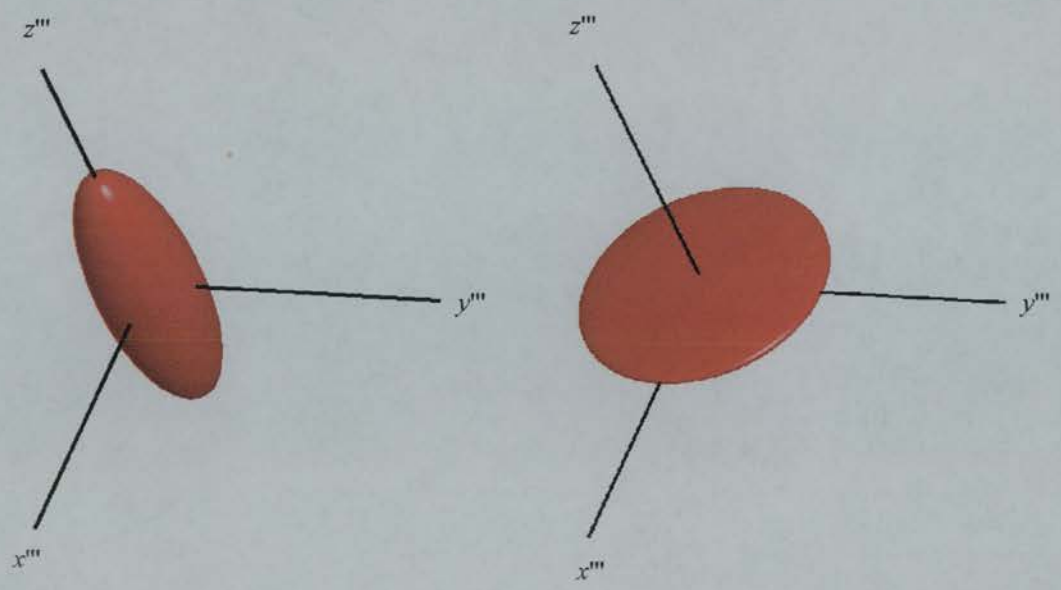
It is interesting to note that the standard deviation index given in Eq. 6.4 reduces to the tetrahedral anisotropy in the case when $\lambda_{x''} = \lambda_{y''}$, although the sign of the square root must be chosen correctly to indicate whether the diffusion is prolate or oblate (see below). In the case of axisymmetric diffusion, with $D_{//}$ and $D_{\perp}$ known, the measured tensor $\boldsymbol{D}$ can be generated using the inverse transformation of Eq. 3.31. A further tetrahedral anisotropy index (A_{tw}) has recently been proposed (174), which can be used in conjunction with the orthogonal A_{sd} (Eq. 6.7), to give a fully quantitative measure of anisotropy, where

$$A_{tw} = \frac{1}{\sqrt{6}D_{av}}\sqrt{2\left(D_{xy}^2 + D_{xz}^2 + D_{yz}^2\right)} \qquad [6.20]$$

Equation 6.4b illustrates how the orthogonal A_{sd} and A_{tw} will add in quadrature to give the tensor standard deviation index (174).

6.2.5 Prolateness

Conventional diffusion anisotropy indices, such as A_{sd} and A_{vr} describe the variation of three orthogonal diffusivities, but do not distinguish between prolate ($\lambda_{x''} \approx \lambda_{y''} << \lambda_{z''}$) and oblate ($\lambda_{x''} \approx \lambda_{y''} >> \lambda_{z''}$) diffusion, as illustrated in Fig. 6.1. The major and minor anisotropy indices can be used to describe the diffusion in this case, but they require that the diffusion coefficients are sorted correctly (85). They are also not uniquely scaled (or normalised) and so it can be difficult to interpret the degree of deviation from the axisymmetric prolate or oblate conditions, for any given



$\lambda_{x'} = 240$; $\lambda_{y'} = 240$; $\lambda_{z'} = 720$.
$A_{sd} = 0.400$
$A_{fa} = 0.603$

$\lambda_{x'} = 560$; $\lambda_{y'} = 560$; $\lambda_{z'} = 80$.
$A_{sd} = 0.400$
$A_{fa} = 0.603$

Figure 6.1 Illustration of (a) prolate ($\lambda_{x'} \approx \lambda_{y'} << \lambda_{z'}$) and (b) oblate ($\lambda_{x'} \approx \lambda_{y'} >> \lambda_{z'}$) diffusion ellipsoids, demonstrating how a single conventional anisotropy index does not necessarily distinguish between these cases.

ellipsoid. To overcome these ambiguities a quantity called the prolateness is defined, from the standard deviation and volume ratio anisotropy indices, scaled such that axisymmetric prolate diffusion is represented by zero and axisymmetric oblate diffusion is represented by one. Any other form of diffusion asymmetry can then be represented as a combination of prolate and oblate diffusion and so take the intermediate values.

The concept of the prolateness arises from the different sensitivities of A_{vr} and A_{sd} to prolate and oblate diffusion. A_{vr} is found to scale both prolate and oblate diffusion into the range between zero and one, whereas A_{sd} scales prolate diffusion between zero and one but scales oblate diffusion between zero and 0.5. The plot of A_{vr} against A_{sd} is shown in Fig. 6.2. The upper and lower curves correspond to axisymmetric prolate and oblate diffusion, respectively, for varying degrees of

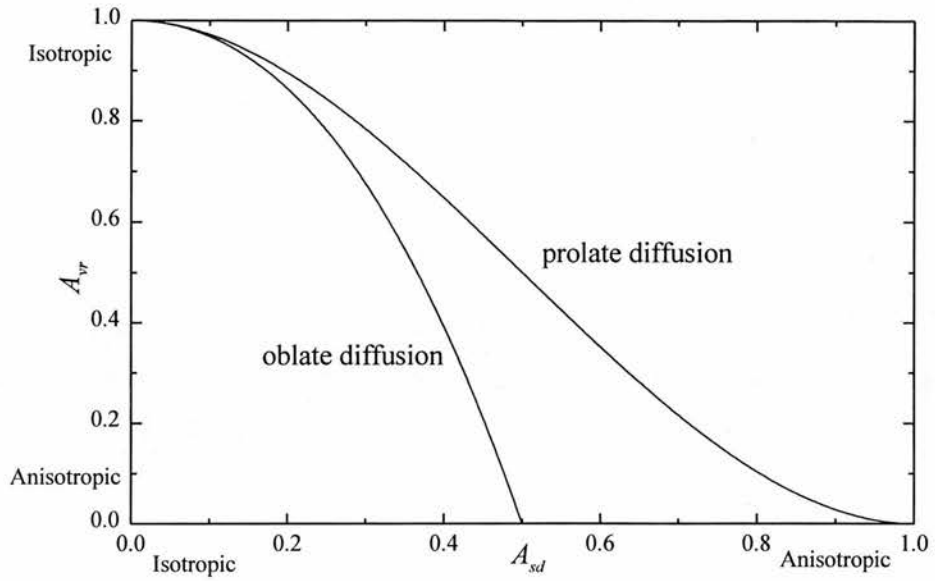


Figure 6.2 The relationship between the standard deviation (A_{sd}) and volume ratio (A_{vr}) anisotropy indices for axisymmetric prolate and oblate diffusion. Any real diffusion ellipsoid can only take values for A_{sd} and A_{vr} that lie on or between the two curves. A given asymmetric diffusion environment can therefore be thought of as being a combination of prolate and oblate diffusion.

anisotropy. Any other form of non-axisymmetric diffusion will fall between these curves and as such can be considered to be a combination of prolate and oblate diffusion. The bounding curves can be derived in terms of A_{vr} and A_{sd}, for the axisymmetric prolate and oblate cases, such that

$$A_{vr}(prolate) = 1 - 3A_{sd}^2 + 2A_{sd}^3 \qquad [6.21a]$$

$$A_{vr}(oblate) = 1 - 3A_{sd}^2 - 2A_{sd}^3 \qquad [6.21b]$$

These equations were derived by performing a polynomial fit of the curves in Fig. 6.2 and were subsequently verified analytically by substituting the expressions for A_{sd} and A_{vr} in Eqs 6.4 and 6.5 into Eq. 6.21. The prolateness (P) can then be defined from Eq. 6.21 above as the ratio of the deviation from the prolate case to the overall deviation between the prolate and oblate cases, such that

$$P = \frac{A_{vr}(prolate) - A_{vr}}{A_{vr}(prolate) - A_{vr}(oblate)} \qquad [6.22]$$

Substituting Eq. 6.21 into Eq. 6.22 gives

$$P = \frac{1 - 3A_{sd}^2 + 2A_{sd}^3 - A_{vr}}{4A_{sd}^3} \qquad [6.23]$$

The prolateness can then be calculated on a pixel-by-pixel basis using Eq. 6.23, from the calculated A_{sd} and A_{vr} obtained from Eqs. 6.7 and 6.8 for the orthogonal case and Eqs. 6.4 and 6.5 for the tensor case. The prolateness is scaled such that axisymmetric prolate diffusion is represented by zero and axisymmetric oblate diffusion is represented by one. The 'transition' between prolate and oblate diffusion is then represented by a value of 0.5. The exception is for isotropic diffusion where $A_{sd} = 0$ and $A_{vr} = 1$, then P becomes infinite because the diffusion is neither prolate nor oblate.

This index used in conjunction with one of the anisotropy indices, either A_{vr} or A_{sd} provides sufficient information to characterise completely the shape of the diffusion ellipsoid, with the anisotropy indicating the variation between the diffusivities and the prolateness indicating their form or symmetry. While knowledge of both A_{vr} and A_{sd} would be sufficient to characterise the shape of the diffusion ellipsoid, using P and A_{sd} (or A_{vr}) seems to be more physically intuitive. The prolateness can also be utilised to provide the correct sign for A_{sd} such that it conforms with the chemical definition of anisotropy (see §8.3). The chemical definition is scaled from -0.5 to 1.0 (as is A_{tet}), so the sign of the square root in Eqs. 6.4 or 6.7 must be chosen correctly such that prolate diffusion takes the positive solution, while oblate diffusion takes the negative solution. It is a simple matter to obtain this sign from P, rather than go through a lengthy eigenvalue sorting procedure. In practice, the chemical definition of anisotropy is difficult to display as an easily interpretable grey scale image, which is why the positive solution to A_{sd} is generally used to give an overall magnitude of anisotropy. An evaluation of the effectiveness of the prolateness index at providing a visual representation of ellipsoid symmetry is presented in Chapter 8.

6.3 Computer simulations

Measurements obtained using the orthogonal, tetrahedral and tensor encoding schemes were simulated using computer programs written in Matlab (The

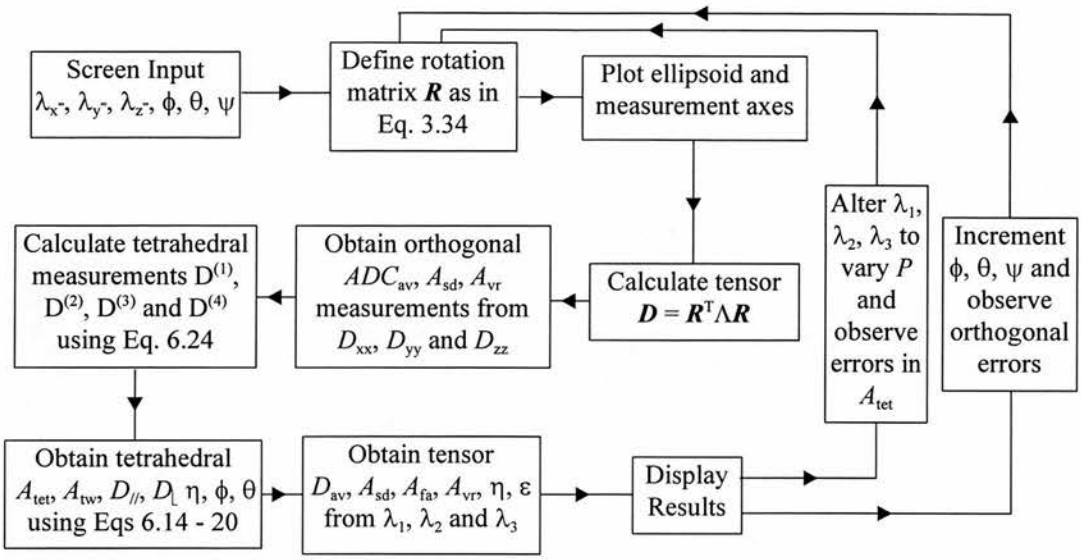


Figure 6.3 A flow chart illustrating the principal functions involved with performing the Matlab computer simulations of the orthogonal, tetrahedral and tensor diffusion encoding schemes.

Mathworks, Natick, MA., USA) on a Sun Ultrasparc workstation (Sun Microsystems, Mountain View, CA., USA). The basic functions of the programs are illustrated by the flow chart in Fig. 6.3. Principal diffusion coefficients ($\lambda_{x'}$, $\lambda_{y'}$, $\lambda_{z'}$) were inputted by the user and the orientation of the diffusion ellipsoid relative to the measurement axes was determined by entering values for ϕ, θ and ψ. The principal diffusion coefficients are then used to define the reduced diffusion tensor (Λ) using Eq. 3.32. The simulations perform the reverse transformation to that given by Eq. 3.31, to determine all six independent diffusion tensor elements, as would be obtained in the measurement or magnet coordinate system - the 'measured' diffusion tensor $\boldsymbol{D}$. The simulation displays the ellipsoid, defined by the inputted parameters $\lambda_{x'}$, $\lambda_{y'}$, $\lambda_{z'}$, ϕ, θ and ψ, relative to the measurement axes and the anisotropy information is given for the three encoding techniques. A typical output window from the simulation is shown in Fig. 6.4. The orthogonal encoding scheme was simulated by taking the diagonal elements of the 'measured' diffusion tensor, D_{xx}, D_{yy} and D_{zz}, as the three orthogonal diffusion measurements. D_{av} was then calculated according to Eq. 6.6 and the anisotropy evaluated using the standard deviation and volume ratio indices according to Eq. 6.7 and 6.8.

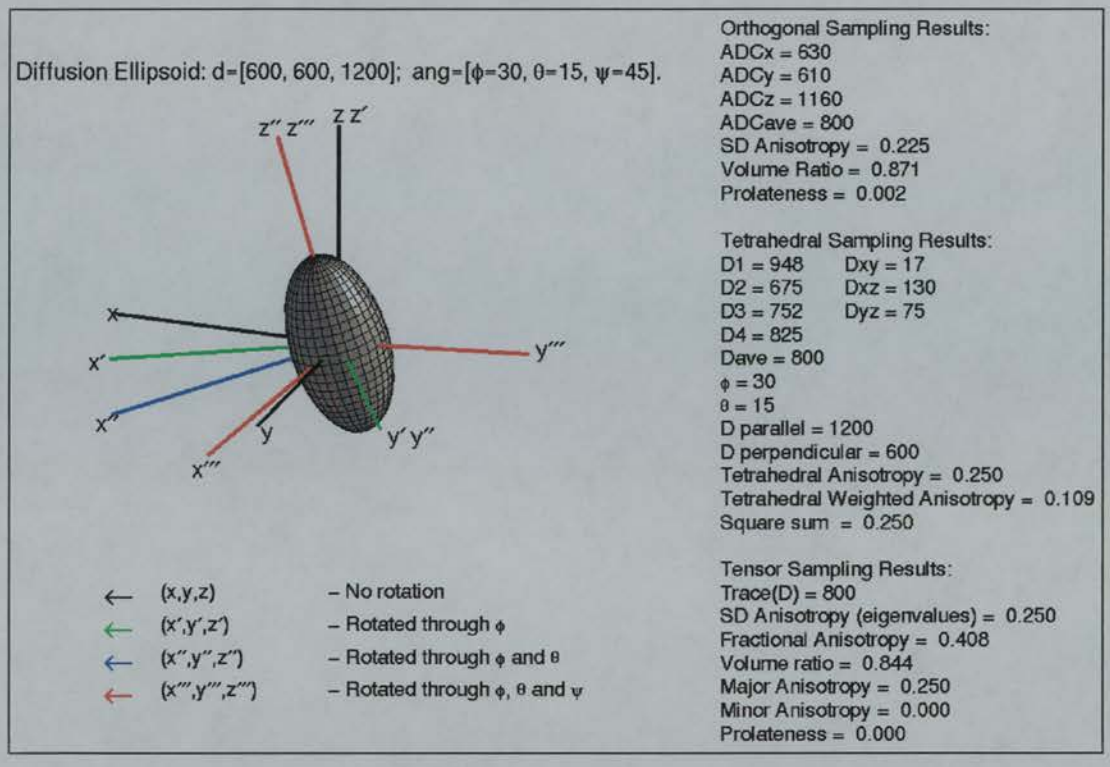


Figure 6.4 A typical output window obtained from the simulation for a single diffusion ellipsoid.

The tetrahedral encoding scheme was simulated by calculating $D^{(1)}$, $D^{(2)}$, $D^{(3)}$ and $D^{(4)}$ from the 'measured' diffusion tensor, given the geometrical arrangement of the tetrahedral gradient vectors, such that (2),

$$D^{(1)} = \tfrac{1}{3}\left[\left(D_{xx} + D_{yy} + D_{zz}\right) + 2\left(D_{xy} + D_{xz} + D_{yz}\right)\right] \qquad [6.24a]$$

$$D^{(2)} = \tfrac{1}{3}\left[\left(D_{xx} + D_{yy} + D_{zz}\right) + 2\left(D_{xy} - D_{xz} - D_{yz}\right)\right] \qquad [6.24b]$$

$$D^{(3)} = \tfrac{1}{3}\left[\left(D_{xx} + D_{yy} + D_{zz}\right) + 2\left(-D_{xy} - D_{xz} + D_{yz}\right)\right] \qquad [6.24c]$$

$$D^{(4)} = \tfrac{1}{3}\left[\left(D_{xx} + D_{yy} + D_{zz}\right) + 2\left(-D_{xy} + D_{xz} - D_{yz}\right)\right] \qquad [6.24d]$$

D_{av} and the off-diagonal tensor elements D_{xy}, D_{xz} and D_{yz} can then be calculated using the expressions given in Eq. 6.25 below (2).

$$D_{av} = \tfrac{1}{4}\left[D^{(1)} + D^{(2)} + D^{(3)} + D^{(4)}\right] \qquad [6.25a]$$

$$D_{xy} = \tfrac{3}{8}\left[D^{(1)} + D^{(2)} - D^{(3)} - D^{(4)}\right] \qquad [6.25b]$$

$$D_{xz} = \tfrac{3}{8}\left[D^{(1)} - D^{(2)} - D^{(3)} + D^{(4)}\right] \qquad [6.25c]$$

$$D_{yz} = \tfrac{3}{8}\left[D^{(1)} - D^{(2)} + D^{(3)} - D^{(4)}\right] \qquad [6.25d]$$

The values obtained for D_{av}, D_{xy}, D_{xz} and D_{yz} can then be used to calculate η, θ, ϕ, $D_{//}$, $D_{\perp}$, A_{tw} and A_{tet} from Eqs 6.14 - 6.20.

The tensor encoding scheme was simulated by using the specified principal diffusivities ($\lambda_{1\text{-}3}$) to define the reduced diffusion tensor (Λ) from Eq. 3.32. The average of the diagonal elements of Λ is calculated from Eq. 6.6 to give D_{av}. The anisotropy elements η and ε can be calculated as in Eqs. 6.2 and 6.3, enabling the axial and equatorial anisotropy indices to be obtained. The anisotropy indices, A_{sd} and A_{vr}, are obtained using Eqs. 6.4a and 6.5. Finally, A_{sd} can be calculated using the measured diffusion tensor elements according to Eq. 6.4b and can be compared to the quadrature sum (174) of the orthogonal and tetrahedral measures given in Eqs. 6.7 and 6.20.

The simulated diffusion ellipsoid's size and shape can be altered by varying the input parameters $\lambda_{x'}$, $\lambda_{y'}$, $\lambda_{z'}$, ϕ, θ and ψ, so that the effectiveness at characterising the diffusion environments of the different anisotropy indices and encoding schemes can be assessed. The tensor scheme is known to describe the diffusion ellipsoid fully in the general case and so can be used as a 'gold standard' (86). The orthogonal scheme only provides a correct description of anisotropy if the measurement and principal diffusion axes are aligned and so its deviation from the 'correct' tensor values can be analysed upon relative rotation of the coordinate systems, which can be achieved by varying ϕ, θ and ψ. This can be viewed simply by plotting the orthogonal anisotropy against the Euler rotation angles. The tensor anisotropy should remain constant, for comparison, as it is rotationally invariant.

The tetrahedral scheme only provides a correct description of anisotropy in the case of axisymmetric diffusion. The deviation from the tensor value can be determined by altering the prolateness (or equatorial anisotropy) i.e. by increasing the deviation between the two minor elements $\lambda_{x'}$ and $\lambda_{y'}$, which increases the asymmetry of the ellipsoid. Again, the tensor anisotropy should give the correct value as this does not assume axisymmetric diffusion when its indices are derived. Therefore, to view this deviation graphs can be plotted showing the tensor and tetrahedral anisotropy indices against the prolateness. The equatorial anisotropy

could be used for these plots, but the prolateness, P, provides a unique scaling between 0 and 1, which is easier to interpret, as a value of 0.5 represents the 'transition' from prolate to oblate diffusion. If the axial and equatorial anisotropy indices were used, the sign of the axial anisotropy would be required to indicate whether the diffusion is prolate (+ve) or oblate (-ve) and then the equatorial anisotropy indicates the deviation between the two equatorial elements. However, the range of values that the equatorial anisotropy can take is not unique and is highly dependent on the shape of the diffusion ellipsoid.

6.4 Quantifying the rotational variance of orthogonal data

For the orthogonal encoding technique, the measured diffusion coefficients, D_{xx}, D_{yy} and D_{zz} agreed with the principal diffusion coefficients, $\lambda_{x'}$, $\lambda_{y'}$ and $\lambda_{z'}$ when the principal diffusion and measurement axes coincided i.e. when the Euler angles were zero or multiples of 90 degrees. The orthogonal anisotropy indices also agreed with their tensor equivalents in this case. The average of the three orthogonal measurements was always equivalent to the trace of the diffusion tensor and was independent of the orientation of the ellipsoid. This verifies the theoretical expectation that the orthogonal imaging technique provides a rotationally invariant measure of the trace of the diffusion tensor (provided that D_{xx}, D_{yy} and D_{zz} are reliably measured).

To illustrate the variation in anisotropy likely to be produced from misalignment of the principal diffusion and measurement axes, graphs were plotted of A_{sd} and A_{vr} against the Euler angles ϕ, θ and ψ, for both the orthogonal and tensor simulations. Two-dimensional graphs were also plotted of ϕ and θ against the anisotropy indices, for different fixed values of ψ, to illustrate typical variations. All of the graphs showed similar features, but their exact form was dependent on the shape of the diffusion ellipsoid. Examples are given in Fig. 6.5 showing the form of the orthogonal $A_{sd}(\phi, \theta)$ surfaces for a number of different ellipsoid examples and for different values of ψ. Similar results were obtained for A_{vr}, but are not illustrated here.

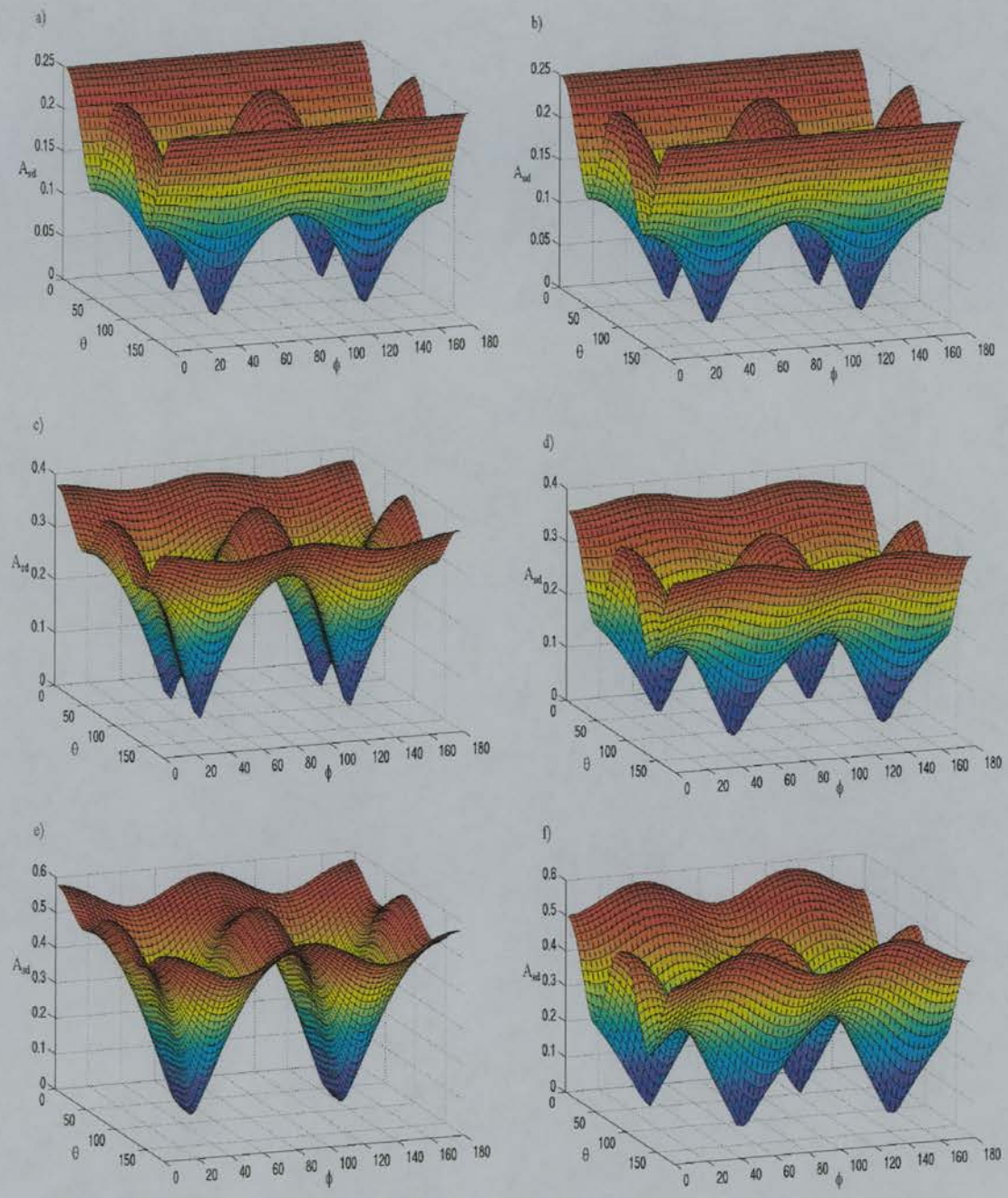


Figure 6.5 The effect of rotation on the orthogonal A_{sd}, illustrated for various shapes of diffusion ellipsoids. The anisotropy is plotted against Euler angles ϕ and θ for different principal diffusivities and fixed values of ψ. The examples illustrated are for principal diffusivities $\lambda = [\lambda_{x'}, \lambda_{y'}, \lambda_{z'}]$ (arbitrary units (a.u.)) and angles ψ, such that a) $\lambda = [600, 600, 1200]$, $\psi = 0°$; b) $\lambda = [600, 600, 1200]$, $\psi = 45°$; c) $\lambda = [600, 300, 1200]$, $\psi = 0°$; d) $\lambda = [600, 300, 1200]$, $\psi = 45°$; e) $\lambda = [600, 0, 1200]$, $\psi = 0°$; f) $\lambda = [600, 0, 1200]$, $\psi = 45°$.

The tensor indices were found to be of constant value for all orientations, equivalent to the orthogonal indices at Euler angles of zero or multiples of 90 degrees. However, for other orientations, the orthogonal indices varied considerably depending on the relative orientation of the axes. The orientation at which the maximum error in the anisotropy measurement occurred was highly dependent on the shape of the diffusion ellipsoid. The orthogonal technique always underestimated the anisotropy relative to the tensor case (except for the special cases previously discussed), for any given orientation and it was observed that in the worst case, a highly anisotropic ellipsoid could be visualised as being isotropic. The maximum error in the anisotropy was defined by ϕ and θ, such that varying ψ did not increase the maximum error, it only altered the values of ϕ and θ at which this error occurred, as illustrated in Fig. 6.5. For the axisymmetric ellipsoid represented in Fig. 6.5a and 6.5b, the third Euler rotation ψ, made no difference to the appearance of the graph, as is expected from theory (85). It is also evident from these graphs that at certain points, e.g. near to the bottom of a trough, a small change in orientation will lead to a large change in anisotropy. This suggests that extremely consistent sample orientation will be necessary when using the orthogonal technique, if studies are to be compared from separate scan sessions

λ_x"	λ_y"	λ_z"	A_{sd}(Tensor)	$\overline{A}_{sd} \pm \sigma$	% error (-ve)
900	0	0	1.000	0.621 ± 0.221	38
800	100	0	0.839	0.526 ± 0.186	37
700	200	0	0.694	0.440 ± 0.159	37
700	100	100	0.667	0.414 ± 0.147	38
600	300	0	0.577	0.371 ± 0.140	36
600	200	100	0.509	0.321 ± 0.114	37
500	400	0	0.509	0.334 ± 0.122	34
500	300	100	0.385	0.247 ± 0.094	36
500	200	200	0.333	0.207 ± 0.074	38
400	300	200	0.192	0.124 ± 0.047	36
300	300	300	0.000	0.000 ± 0.000	0

Table 6.1 The average error incurred by using the orthogonal encoding scheme for describing anisotropy compared to the tensor encoding scheme. The average anisotropy, $\overline{A}_{sd}$, is calculated over all three Euler angles in the range {0°, 90°}. All diffusion coefficients are measured in arbitrary units.

To gain a further insight into the errors likely to be incurred if the orthogonal technique is used to sample diffusion anisotropy, the anisotropy was calculated for all values of ϕ, θ and ψ in the range 0° to 90°, increment 1°, for a given ellipsoid. It can be seen from Fig. 6.5, that this range of angles is sufficient for calculating the mean anisotropy error, as subsequent orientations just repeat the observations occurring in this range. The mean, $\overline{A}_{sd}$, and standard deviation, σ, of A_{sd} was then calculated over this range, to give an indication of the likely error incurred for an 'average' misalignment of the principal diffusion and measurement axes. These values are shown for several diffusion ellipsoids, with constant trace, in Table 6.1. On average, an underestimation of about 35% will be incurred if the orthogonal A_{sd} is used rather than the tensor equivalent. This error seems to be fairly independent of the shape of the diffusion ellipsoid, provided that the ellipsoid is not exactly spherical. It may be that the slight differences found can be accounted for because of the non-continuous sampling resulting from using only 1° increments, when calculating $\overline{A}_{sd}$. However, using discreet sampling at 1° increments was considered to be sufficient to give an indication of the likely approximate error because of the continuous (smooth) nature of the functions in Fig. 6.5.

6.5 The effect of ellipsoid symmetry on tetrahedral data

The tetrahedral scheme produced spurious results in several cases, the most notable being when the principal diffusion and measurement axes are aligned, including the unique case of isotropic diffusion. The anisotropy information could not be obtained for these examples, using the tetrahedral scheme, because the formulae for η, ϕ and θ (Eqs. 6.14-6) all have undefined (zero divided by zero) terms, due to the zero valued off-diagonal tensor elements. To overcome this problem for isotropic diffusion, a condition was put into the simulation, such that if the three principal diffusion coefficients are the same then the simulations give default 'isotropic' values for the anisotropy information. To avoid the undefined terms arising in the case of anisotropic diffusion with aligned coordinate systems, i.e. with Euler angles of zero or multiples of 90°, these angles were set minutely above their specified values and the evaluated anisotropy information was found to be negligibly

different from the true values. Another problem is that the tetrahedral scheme is derived such that, for prolate diffusion ellipsoids λ_z- must be the largest term and λ_x- must be greater than λ_y-, in the general case. Similarly, for oblate diffusion ellipsoids λ_z- must be the smallest element and λ_x- must still be greater than λ_y- (85). The initial input must be specified in this order if spurious results are to be avoided. This could clearly lead to problems when implementing the tetrahedral scheme in a real situation because the principal diffusion coefficients and their orientations are not known *a priori*.

The average diffusion coefficient (D_{av}) calculated from the tetrahedral measurements (Eq. 6.25a) gave identical results to the tensor and orthogonal equivalent measures, for any orientation of the ellipsoid and even in the general case when the ellipsoid is not axisymmetric. The Euler angles ϕ and θ agree with the user inputted values provided that the diffusion is axisymmetric and the principal diffusion coefficients are ordered correctly. There are, however, some discrepancies as occasionally the calculated angles appear in a different quadrant to that inputted, but this can be explained by considering the symmetry of the ellipsoid. The tetrahedral anisotropy A_{tet} (Eq. 6.19) agrees with the tensor A_{sd} in magnitude for all orientations of axisymmetric diffusion ellipsoids. However, A_{tet} has the advantage that it distinguishes between prolate and oblate diffusion, with prolate diffusion having positive values between 0 and 1, and oblate diffusion having negative values between 0 and 0.5. A_{tet} gave results that were quite different from the tensor A_{sd} or axial anisotropy, when the diffusion ellipsoid was not axisymmetric.

To quantify the errors arising due to the deviation of the ellipsoid from the axisymmetric condition, graphs of A_{tet} and the tensor A_{sd} indices were plotted against P (or A_{minor}). For simplicity, the graphs are illustrated only for prolate ellipsoids ($0 < P < 0.5$), but similar results were found for oblate diffusion ($0.5 < P < 1$). The graphs plotted in Fig. 6.6 clearly show how A_{tet} can take values up to $\pm\infty$ in the worst case, which will give meaningless information on the true extent of the anisotropy, since A_{tet} is only defined for values between -0.5 and 1.0. Figure 6.6 also shows how the errors arising are highly dependent on the ordering of the eigenvalues inputted into the simulation and of the orientation of the principal diffusion axes, for non-axisymmetric ellipsoids. This is a fundamental problem because the tetrahedral

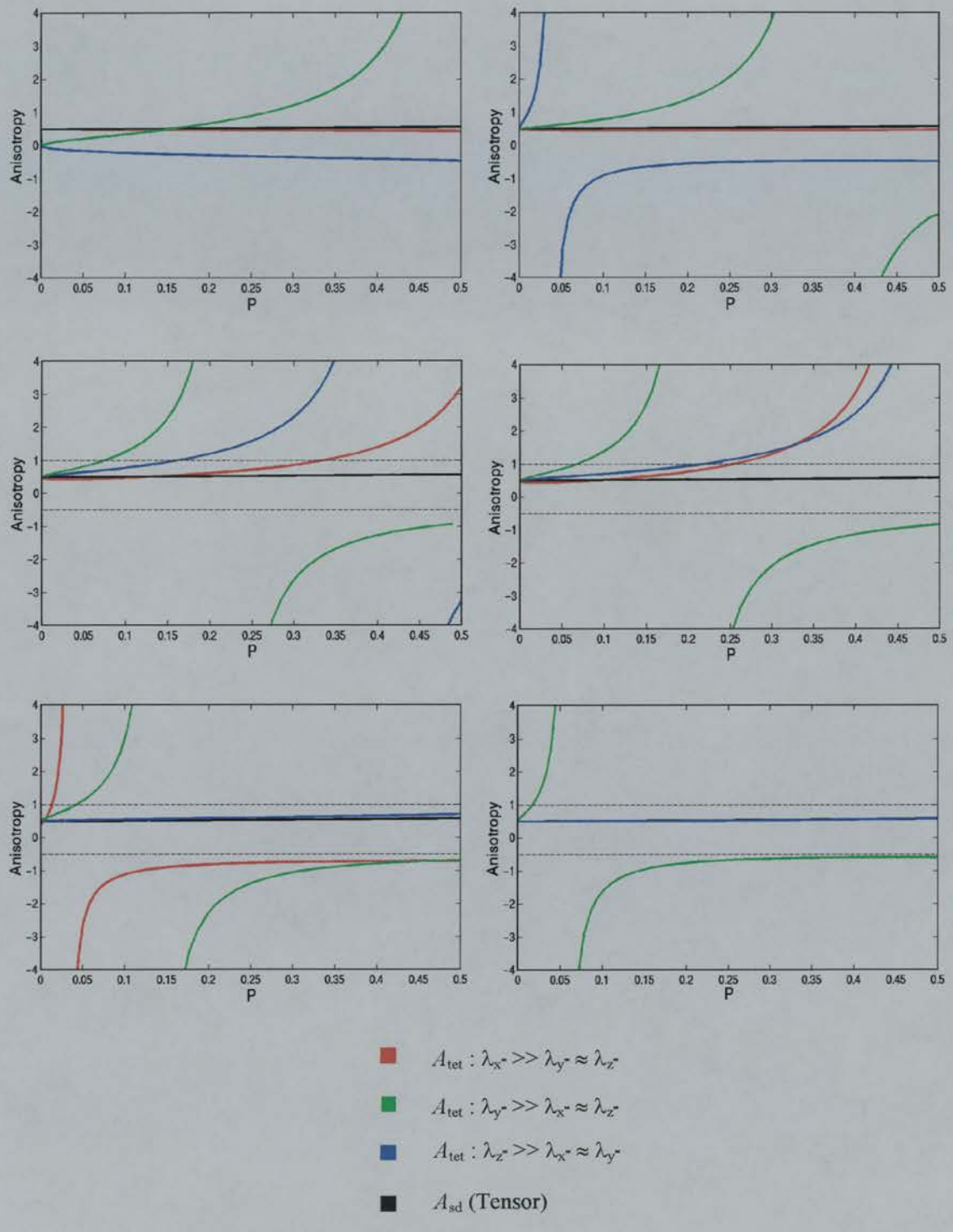


Figure 6.6 The variation of the tensor A_{sd} and tetrahedral A_{tet} anisotropy indices with prolateness (P) for a diffusion ellipsoid with fixes major anisotropy. Each plot illustrates the variation in A_{tet} for different permutations of the principal diffusion coordinate system. Examples are given for different Euler rotations with $\phi = \theta = \psi$, for all angles a) 0°; b) 25°; c) 35°; d) 36°; e) 40°; f) 45°. The dashed horizontal lines represent the extent to with A_{tet} is defined in the axisymmetric case.

technique only samples D_{av} and not the individual diagonal elements and so the assignment of the coordinate system that produces the minimum error cannot be known without further information being obtained. These errors arise because the expression for η in Eq. 6.14 is derived from Eqs. 6.11 - 6.13 under the conditions of axisymmetric diffusion. It is found that these expressions are no longer valid in the general case of non-axisymmetric diffusion as terms need to be added that involve ε and ψ.

There are certain situations where the tetrahedral anisotropy gives a reasonable approximation to the 'true' tensor anisotropy, even though the diffusion may be non-axisymmetric. There are points on the graphs in Fig. 6.6 where the tetrahedral anisotropy and the tensor anisotropy are equivalent, even though the diffusion is not axisymmetric, i.e. the tensor and tetrahedral curves intersect. Clearly, at and around these points, the anisotropy will be well approximated by the tetrahedral method. The example given by Conturo et al. (85), with $\lambda_{x'} = 0.8090$, $\lambda_{y'} = 0.7228$ and $\lambda_{z'} = 1.061$ which give $A_{major} = 0.114$ and $A_{minor} = 0.05$, is found to yield good results for the measured tetrahedral anisotropy with $A_{tet} = 0.101$ compared to the tensor $A_{sd} = 0.117$. This is not surprising because the tetrahedral anisotropy is close to its intersection with the tensor anisotropy and so this is a case where the tetrahedral technique will give a reasonable approximation to the true anisotropy. However, if the equivalent 'nearly' axisymmetric case of $\lambda_{x'} = 0.7669$, $\lambda_{y'} = 0.7649$ and $\lambda_{z'} = 1.061$ is considered which give $A_{major} = 0.114$ and $A_{minor} = 0.001$, then the measured $A_{tet} = 0.002$ compared to the tensor $A_{sd} = 0.114$, i.e. the diffusion is described as being nearly isotropic, using the tetrahedral technique!

The tetrahedral anisotropy plotted in Fig. 6.6 becomes highly dependent on the relative orientation of the principal diffusion and measurement axes, when the diffusion is not axisymmetric. The exact form of these curves is not too critical, but it is important to realise that large errors can result if the tetrahedral technique is used and the diffusion is not *exactly* axisymmetric. Indeed, Fig. 6.6 indicates that, in general, the error in measuring anisotropy using the tetrahedral technique does not increase uniformly as the diffusion becomes less axisymmetric and the tetrahedral anisotropy can provide a particularly poor estimate if the diffusion is 'just' not

axisymmetric, as shown above and in Fig. 6.6a. It is interesting to observe that when the ellipsoid is oriented at $\phi = \theta = \psi = 45°$ (i.e. along the tetrahedral directions) and the eigenvalues are inputted in the correct order, the anisotropy is correctly evaluated for all diffusion ellipsoid shapes.

Several studies have suggested that certain regions such as the spinal cord (171), corpus callosum and pyramidal tracts (128) may exhibit cylindrical symmetry and as such be suitable for a reliable diffusion evaluation, using the tetrahedral encoding scheme. However, this simulation suggests that accurate diffusion anisotropy measurements are unlikely to be obtained using the tetrahedral technique in a realistic in vivo situation, where fibres are unlikely to be exactly axisymmetric, particularly when the effects of noise are considered (115).

6.6 Conclusions

These simulations illustrated that if anisotropy information is not required and only an average diffusion coefficient is of interest, then either of the orthogonal, tetrahedral or tensor schemes can be used to measure accurately D_{av} or $tr(\boldsymbol{D})$. However, in practice, the orthogonal and tetrahedral techniques will only provide reliable D_{av} measurements when imaging gradient cross-terms are negligible. If cross-terms can be minimised then it may be desirable to use the tetrahedral scheme for sampling the ADC because the simultaneous application of all three orthogonal gradients potentially results in a three-fold increase in b-value. This could be valuable when trying to measure small diffusivities using conventional clinical scanners, where a high b-value is required to enable sufficient signal attenuation to provide an accurate measurement.

This study also confirms that the tensor encoding scheme must be used, to provide a full and reliable description of diffusion anisotropy. The orthogonal anisotropy will only produce a correct estimate of anisotropy if the measurement and principal diffusion coordinate systems are aligned, which is highly unlikely to occur in all voxels in an in vivo situation, where there are highly complex fibre arrangements. On average, the anisotropy is likely to be underestimated by around 35%. However, at certain orientations, highly anisotropic structures can be viewed

as being totally isotropic, but the orientation at which this maximum error occurs is highly variable and dependent on the shape of the diffusion ellipsoid. These findings would appear to mirror the experimental observations of Pierpaoli et al. (128), who observed a general underestimation of the anisotropy using orthogonal sampling compared to the tensor scheme. However, they occasionally observed a larger degree of error, whereby highly anisotropic structures were misrepresented as being almost isotropic.

Anisotropy measurements obtained using the tetrahedral encoding scheme were shown to produce large errors in many situations. These errors are dependent on the assignment of the principal diffusion coordinate system and also on the relative orientation of the principal diffusion and measurement axes, for non-axisymmetric diffusion. The errors arising are difficult to predict because they do not always increase with increasing deviation from the axisymmetric condition. In some cases, anisotropic diffusion can be assigned isotropic values when the diffusion is minutely different from the axisymmetric condition.

The simulated data suggests that the tetrahedral encoding scheme is likely to cause large and unpredictable errors in many realistic situations (e.g. in the presence of noise) and cannot be recommended as a reliable method for describing diffusion anisotropy. The orthogonal encoding scheme is also likely to cause large errors in the measured anisotropy, but is probably more useful than the tetrahedral scheme because the anisotropy is always underestimated, and fewer images need to be collected. As expected from theory, the tensor encoding scheme provided reliable anisotropy measurements that characterised the shape of the diffusion ellipsoid and were independent of errors due to ellipsoid symmetry and orientation.

7

Optimisation of Diffusion-Weighting Acquisition Schemes

7.1 Introduction

The accurate quantification of diffusion coefficients using MR requires a suitable level of diffusion-weighting to be chosen. The diffusion-weighted image must show sufficient attenuation, relative to a 'baseline' image, for the desired range of diffusion coefficients, without reducing the signal to the noise level. Therefore, the accuracy of a diffusion measurement is likely to be determined by the choice of *b*-value and the degree of noise contamination in the image. Some of the problems associated with noise contamination of diffusion images were discussed in §5.3.1, although a rather simplistic approach was adopted. The effect of noise on measurements of diffusion anisotropy was observed and an increase in anisotropy was demonstrated with increasing noise contamination. To date, measurements of noise in diffusion images have largely been obtained by measuring the SNR of the diffusion-weighted image (115,129), as in §5.3.1. However, this approach is shown to provide a measurement of noise contamination that is highly dependent on experimental parameters, such as *b*-value and diffusivity. A more reliable approach is therefore presented in this section, providing a measure that is largely independent of the experimental parameters chosen. This method is based on obtaining the diffusion-to-noise ratio (DNR), which is measured as the SNR in the calculated ADC

or ADC_{av} maps. The degree of noise contamination that enables acceptable diffusion anisotropy measurements to be obtained can then be found and expressed as a DNR value (as was done in §5.3.1 with the SNR). This DNR value is shown to provide a noise measurement that is applicable in most experimental situations because of the lack of dependence on experimental parameters.

A procedure is then developed that enables optimisation of the *b*-value and number of image acquisitions to produce sufficient DNR such that accurate diffusion measurements can be obtained over the range of diffusivities typically encountered in the human brain. This procedure is based on an extension of the scheme developed by Xing et al. (175). They demonstrated that the DNR of the resultant ADC image, obtained from a single baseline and diffusion-weighted image, is dependent upon the SNR of the baseline image, the diffusivity being measured, the *b*-values used and the number of acquisitions at each *b*-value. In this work, the optimisation procedure was generalised using a simple isotropic model to make it applicable for acquisition schemes involving multiple sampling directions. The limitations of these assumptions are discussed in the context of typical applications in the human brain. The original scheme (175) optimised the signal acquisition procedure to obtain the maximum DNR for single diffusivity. For an in vivo situation, this is far from ideal as it is required to measure a range of diffusivities. As such, the optimisation scheme was also extended to provide optimised acquisition parameters that provide the required DNR over the range of diffusivities encountered in human brain imaging. The effectiveness of the optimised acquisition scheme was evaluated using Monte-Carlo simulations, in vitro phantom studies and finally in vivo on a normal human subject.

7.2 Noise considerations for a single diffusion measurement

For a single diffusion measurement, the signal attenuation in the diffusion-weighted image can be given, simply, by the Stejskal-Tanner relationship (16) in Eq. 3.16. For a two-point ADC measurement, obtained from images acquired with *b*-values b_1 and b_2 and assuming a mono-exponential decay of the signal attenuation with *b*-value (69), the ADC can be calculated from Eq. 4.3. The 'baseline' image has

been traditionally acquired with $b_1 = 0$, to minimise acquisition time as this acquisition is directionally independent because no diffusion gradients are applied. If a non-zero b-valued 'baseline' image is acquired in an anisotropic medium, then the measured signal will depend on the direction of the applied gradient (71). In this case, it is necessary to calculate the ADC from Eq. 4.3 using b_1 and b_2 images acquired along the same direction. Thus, for any given direction, the ADC can be calculated from Eq. 7.1 below, using two different b-values,

$$ADC = \frac{1}{b_2 - b_1}\left[\ln\left(S_{b_1}\right) - \ln\left(S_{b_2}\right)\right] \qquad [7.1]$$

where S_{b_1} and S_{b_2} are the signals measured from the images acquired with b-values b_1 and b_2 respectively (16).

In a realistic MR experiment, measurements of S_{b_1} and S_{b_2} will be contaminated with noise. The SNR of the b_1 image will be given by

$$SNR_{b_1} = S_{b_1} / N \qquad [7.2]$$

where N is the standard deviation of the noise. In order to observe the noise contamination resulting in the calculated ADC image, it is required to know the contamination in the measurements of $\ln(S_{b_1})$ and $\ln(S_{b_2})$. This can be obtained by calculating the standard deviation (σ) of $\ln(S_{b_1})$ and $\ln(S_{b_2})$, such that (175)

$$\sigma_{\ln\left(S_{b_1}\right)} = \frac{1}{SNR_{b_1}} \qquad [7.3]$$

$$\sigma_{\ln\left(S_{b_2}\right)} = \exp\left[ADC\left(b_2 - b_1\right)\right]\frac{N}{S_{b_1}} = \frac{\exp\left(\xi\right)}{SNR_{b_1}} \qquad [7.4]$$

where

$$\xi = ADC\left(b_2 - b_1\right) = ADC(\Delta b) \qquad [7.5]$$

and ξ can be considered to be a non-dimensional parameter for diffusion-weighting between b_1 and b_2 (175). The variances of $\ln(S_{b_1})$ and $\ln(S_{b_2})$ are just the square of their respective standard deviations and enable a simple evaluation of the variance of the ADC using Eq. 7.1,

$$\begin{aligned}\sigma^2_{ADC} &= \mathrm{var}(ADC) \\ &= \frac{1}{(b_2 - b_1)^2}\left[\frac{\mathrm{var}\left(\ln\left(S_{b_1}\right)\right)}{n_1} + \frac{\mathrm{var}\left(\ln\left(S_{b_2}\right)\right)}{n_2}\right] \\ &= \left(\frac{1}{SNR_{b_1}}\frac{ADC}{\xi}\right)^2\left(\frac{1}{n_1} + \frac{\exp(2\xi)}{n_2}\right)\end{aligned} \quad [7.6]$$

where n_1 and n_2 represent the number of acquisitions of the b_1 and b_2 images, respectively. The *DNR* is then simply calculated from the SNR of the ADC, to give a measure of the degree of noise contamination in the resulting calculated diffusivity, such that

$$DNR = \frac{ADC}{\sigma_{ADC}} = SNR_{b_1}\kappa_D \quad [7.7]$$

where

$$\kappa_D = \frac{\xi}{\sqrt{\frac{1}{n_1} + \frac{\exp(2\xi)}{n_2}}} \quad [7.8]$$

Equation 7.7 demonstrates how the resultant DNR depends on the 'baseline' signal-to-noise ratio (SNR_{b_1}) and the 'sensitivity' of the diffusion-weighting (κ_D) (175). The term κ_D depends on the chosen *b*-values, the number of acquisitions of each image and the diffusivity measured. In effect, the sensitivity κ_D indicates the range of diffusivities over which accurate measurements can be obtained (i.e. satisfactory DNR is achieved) with the chosen acquisition parameters b_1, b_2, n_1 and n_2.

Optimisation of the diffusion-weighting scheme can then be achieved, as described by Xing et al. (175), by altering n_1 and n_2 for the experimental *b*-values used, to maximise the DNR for a given ADC. It can be seen from Eq. 7.7 that the DNR is also dependent on SNR_{b_1}. However, this term is defined by the T_1 and T_2 relaxation times of the imaging subject, the echo time of the imaging sequence and so is generally fixed for a given imaging sequence. Therefore, it is not considered to be a user variable parameter in the optimisation scheme, except for the fact that SNR_{b_1} generally varies with the chosen baseline b_1 value, but is independent of the Δb value used.

7.3 Implementation of the optimisation scheme

7.3.1 Maximising the diffusion sensitivity κ_D

Selecting the optimised acquisition parameters is not a straightforward matter as there are many variables in the optimisation scheme, namely *DNR*, SNR_{b_1}, b_1, b_2, n_1, n_2 and *ADC*. There are many different ways of optimising these parameters using Eqs. 7.5, 7.7 and 7.8, and the procedure adopted by Xing et al. (175) is discussed here. The principal aim of any optimisation scheme is to achieve a high DNR, such that a reliable diffusion measurement can be obtained. Equation 7.7 demonstrates how the DNR is dependent on SNR_{b_1} and κ_D and so maximising these two parameters is a priority for obtaining high DNR. As discussed in §7.2, SNR_{b_1} is largely fixed by the imaging sequence, hardware and subject, resulting in the maximisation of κ_D being the priority.

It can be seen from Eq. 7.8 that κ_D depends on n_1, n_2 and the non-dimensional diffusion-weighting parameter ξ. The variation of κ_D can be observed for different values of ξ, as shown in Fig. 7.1, illustrated for the case with $n_1 = 1$ and $n_2 = 1$. A maximum κ_D value (κ_{max}) in this curve is reached when $\xi_{max} = ADC(b_2 - b_1) \approx 1.1$.

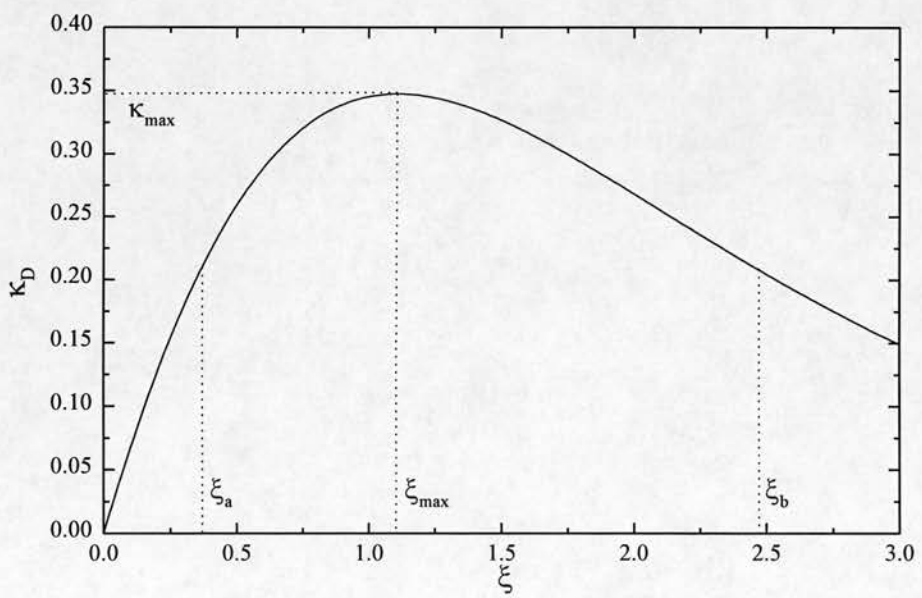


Figure 7.1 The diffusion-weighting sensitivity κ_D, plotted as a function of the non-dimensional diffusion weighting parameter $\xi = \mathrm{ADC}(b_2 - b_1)$, for the case when $n_1 = n_2 = 1$. The maximum κ_D value occurs when $\xi \approx 1.1$. The interval $\{\xi_a, \xi_b\}$ is the 60% interval, defined for ξ at 60% of κ_{max}.

Below this value for ξ_{max} ($\xi < \xi_{max}$), a rise in ξ increases the degree of diffusion attenuation, which improves the accuracy of the diffusion measurement. However, above ξ_{max} ($\xi > \xi_{max}$) the diffusion attenuation becomes too large and the measurement becomes corrupted by noise, reducing the DNR and hence the accuracy of the measurement. Thus, the accuracy of the diffusion measurement becomes optimal (maximum DNR) for a given *ADC* by selecting a value of Δb in Eq. 7.5 such that $\xi = \xi_{max}$. This procedure only optimises the measurement for an individual *ADC* value and so requires *a priori* information on this value. However, as long as a rough idea of the *ADC* is known, then the measurement will be reasonable over a small interval in the range $\xi_a < \xi_{max} < \xi_b$ as indicated by the relatively broad maximum in Fig. 7.1. Xing et al. (175) suggest defining an arbitrary interval $\{\xi_a, \xi_b\}$ with the values of ξ obtained at 60% of κ_{max}, the so called 60% interval. The limitations of this arbitrarily chosen range are discussed in more detail in §7.4 and §7.5.

7.3.2 Optimising the signal averaging scheme

So far, only the case of a single acquisition has been considered ($n_1 = n_2 = 1$). Clearly, altering the values of n_1 and n_2 in Eq. 7.8 will change the form of the curve in Fig. 7.1. It is logical to assume that by signal averaging (i.e. increasing n_1 and n_2), the resultant DNR will be increased. As SNR_{b_1} is defined as the SNR in a *single* image acquired at b-value b_1, increasing n_1 and n_2 must increase the value of κ_D (for a given ξ). It is also logical to perform more acquisitions on the b_2 image (i.e. $n_2 > n_1$) as this has a lower SNR (because $b_2 > b_1$). The second stage of the optimisation procedure is, therefore, to determine the optimum combinations of n_1 and n_2 that achieve the maximum κ_D value for a given imaging time. The imaging time (T_{im}) for a single ADC measurement will be given by

$$T_{im} = (n_1 + n_2)T_{aq} = mT_{aq} \qquad [7.9]$$

where T_{aq} is the time taken to acquire a single diffusion-weighted image. Thus, for a given m, it is required to find the combination of n_1 and n_2 that produces the maximum value of κ_{max}. The optimal combinations of n_1 and n_2 have been found (175) and are displayed for the first 14 values of m in Table 7.1. The corresponding

m	n_1	n_2	ξ_a	ξ_{max}	ξ_b	κ_{max}
2	1	1	0.37	1.11	2.47	0.347
3	1	2	0.41	1.19	2.54	0.470
4	1	3	0.45	1.25	2.59	0.556
5	1	4	0.48	1.30	2.64	0.622
6	1	5	0.51	1.34	2.68	0.677
7	2	5	0.43	1.19	2.57	0.729
8	2	6	0.45	1.25	2.59	0.786
9	2	7	0.47	1.25	2.62	0.835
10	2	8	0.48	1.30	2.64	0.880
11	2	9	0.49	1.30	2.66	0.920
12	3	9	0.45	1.25	2.59	0.962
13	3	10	0.46	1.25	2.61	1.000
14	3	11	0.47	1.25	2.62	1.040
15	3	12	0.48	1.30	2.64	1.080

Table 7.1 Optimised signal averaging schemes to achieve the maximum sensitivity κ_{max} (and hence DNR) for a given total number of images (m). [From Xing et al. (175)].

variation of κ_D with ξ is shown in Fig. 7.2 for the first four of these optimal solutions with even m.

It can be seen from Table 7.1 and Fig. 7.2 that for the optimal solutions of n_1 and n_2, the maximum value of κ_D increases as m increases, as expected. Also, the range $\{\xi_a, \xi_b\}$ defined by the 60% interval remains reasonably constant for all optimal solutions (175). There is a small variation in the values ξ_a, ξ_{max} and ξ_b, but this is only likely to effect the accuracy of ADC measurements corresponding to the edge of the defined 60% interval. Therefore, experimental implementation is facilitated, whereby for insufficient DNR, the number of acquisitions can be increased, following the optimised scheme in Table 7.1 until a sufficiently high DNR is reached. The range of ADCs which are covered by the 60% interval $\{ADC_a, ADC_b\}$ will remain roughly the same, meaning that no adjustment of the b-value is necessary after signal averaging (175).

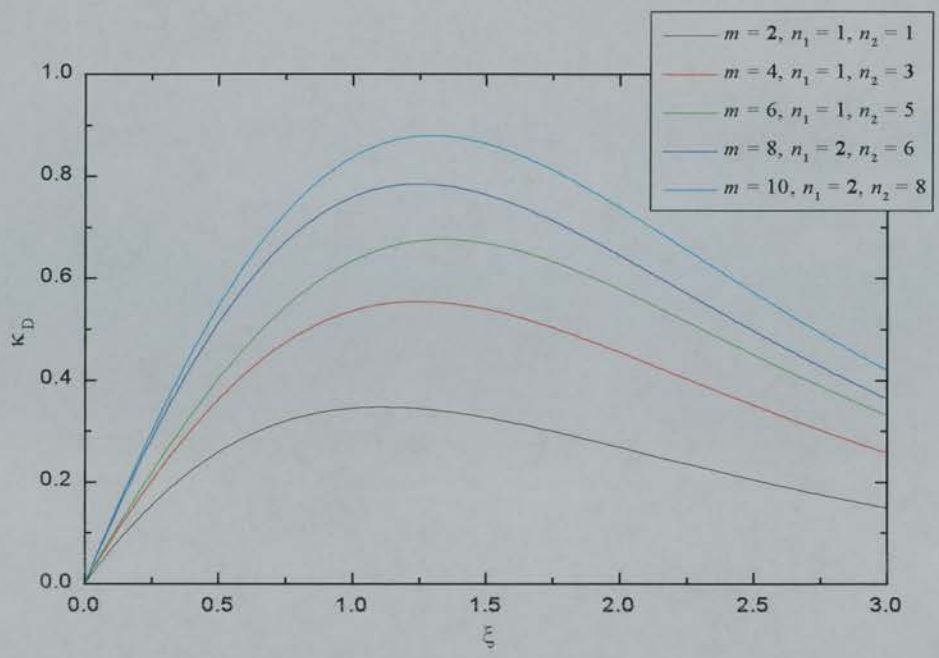


Figure 7.2 The diffusion-weighting sensitivity κ_D, plotted as a function of the non-dimensional diffusion weighting parameter $\xi = ADC(b_2 - b_1)$, for different optimum combinations of n_1 and n_2 as given by Table 7.1. Examples are given for the first five even values of m.

7.3.3 Practical implementation of the optimisation scheme

Implementation of the optimisation scheme to find the ideal effective b-value (Δb) and number of acquisitions (n_1 & n_2), is summarised by flow chart in Fig. 7.3 (175). The first step of the procedure is to calculate κ_D from Eq. 7.7 as the ratio of the required DNR to SNR_{b_1}. The DNR required such that acceptable measurements are obtained was not discussed as part of the original study (175). This is somewhat surprising as it is an essential input parameter for the whole optimisation procedure! The value for SNR_{b_1} is measured from a typical experiment obtained using a single acquisition at b-value b_1. If the calculated κ_D is greater than κ_{max} for the chosen n_1 and n_2 (Table 7.1), then further signal averaging is required to achieve sufficient DNR. When m has been increased sufficiently such that the calculated κ_D is less than κ_{max}, verification that the required range of ADCs $\{ADC_1, ADC_2\}$ fall into the 60% interval is required. Firstly, Eq. 7.8 must be solved for ξ_a and ξ_b, given the values of κ_D, n_1 and n_2. The effective b-value (Δb) can then be calculated, given ADC_1, such that

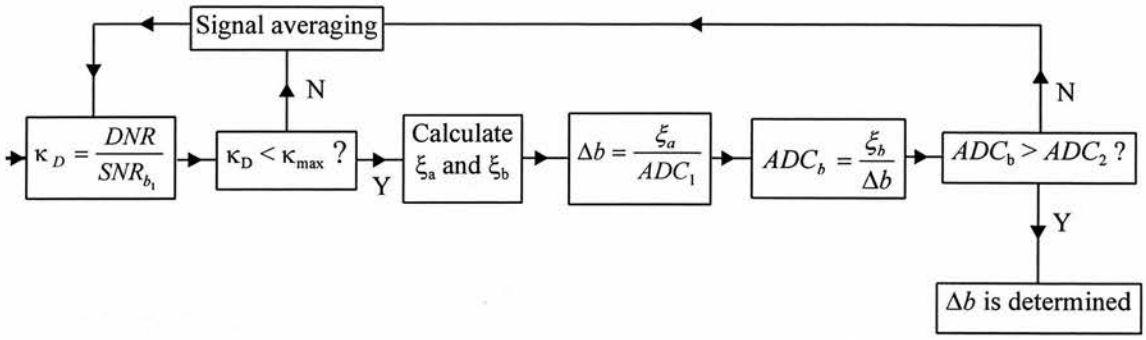


Figure 7.3 Schematic flow diagram illustrating the processes involved in optimising the *b*-values used in a two-point ADC measurement, following the procedures of Xing et al. (175).

$$\Delta b = \xi_a / ADC_1 \qquad [7.10]$$

and the highest diffusion coefficient that falls into the 60% interval is calculated from

$$ADC_b = \xi_b / \Delta b \qquad [7.11]$$

If $ADC_b > ADC_2$, then the required range of ADCs will be covered by the 60% interval and the optimum b_{eff}, n_1 and n_2 have been obtained. If $ADC_b < ADC_2$, then further signal averaging will be required until ADC_2 falls into the 60% interval. However, while following this procedure finds the optimum *b*-value for the given measurement, it does not account for whether this *b*-value is practically achievable or not, a problem that is discussed in more detail in §7.5.

7.4 Achieving an acceptable DNR

7.4.1 The DNR requirement

The first stage of the optimisation scheme summarised in Fig. 7.3 requires the calculation of κ_D from the required DNR and SNR_{b_1}. The value of SNR_{b_1} is considered to be fixed, determined by the hardware and imaging sequence used. However, the DNR selected is purely an arbitrary choice, as the minimum DNR (DNR_{min}) that enables acceptable diffusion measurements to be obtained is not known. Furthermore, the method of Xing et al. (175) uses an arbitrary 60% interval, to define the range of acceptable DNR. Clearly, before such an arbitrary range can be defined, the actual value of DNR_{min} must be known, to ensure that all diffusivities falling into the 60% interval have $DNR > DNR_{min}$. Previous studies have considered

the degree of noise contamination in terms of an average SNR (SNR_{av}) for the experiment (115,129). It was suggested that a minimum value for SNR_{av} (SNR_{min}) of 20 and ideally of 40 is required to obtain acceptable noise independent diffusion measurements (129). Therefore, the aims of this section are to establish DNR_{min} for typical imaging situations, to explore its variability with experimental parameters, such as b and ADC, and to compare this with the equivalent SNR_{av} analysis.

7.4.2 Monte Carlo simulations

The effects of noise on typical DTI experiments were observed by using Monte Carlo simulations, written in Matlab. A set of N synthetic, noise free 256 × 128 images of a numerical phantom were generated to mimic typical two-point DTI experiments. For a two-point fit with a zero-valued b_1 'baseline' image, $N = 7$, and diffusion gradients were simulated using the 'uniform' sampling scheme with a single amplitude along vectors $(1/\sqrt{2},0,1/\sqrt{2})$, $(-1/\sqrt{2},0,1/\sqrt{2})$, $(0,1/\sqrt{2},1/\sqrt{2})$, $(0,1/\sqrt{2},-1/\sqrt{2})$, $(1/\sqrt{2},1/\sqrt{2},0)$ and $(-1/\sqrt{2},1/\sqrt{2},0)$, along with the baseline image with no diffusion gradients applied (83). For a non-zero 'baseline' b-value, $N = 12$, and diffusion gradients are simulated with two different amplitudes along the same vectors as above. The alternative 'conventional' sampling scheme of (1,0,0), (0,1,0), (0,0,1) $(1/\sqrt{2},1/\sqrt{2},0)$, $(1/\sqrt{2},0,1/\sqrt{2})$, $(0,1/\sqrt{2},1/\sqrt{2})$ was also investigated (24,85). The imaging parameters used to construct these simulated data sets were based on those used in the EPI diffusion sequence discussed in §4.3.2. Values for the b-matrix elements used in the various simulations were calculated numerically from these parameters (81,82).

The numerical phantom consisted of a large circle of radius 30 voxels (area 2821 voxels), placed in the centre of the FOV. The elements of $\boldsymbol{D}$ simulated in the numerical phantom can be altered to mimic different diffusion situations, i.e. to vary the anisotropy or $tr(\boldsymbol{D})$ etc. The signal intensities in each voxel of the individual diffusion-weighted images $S(\boldsymbol{b})$ can then be simulated from Eq. 3.23, rearranged such that

$$S(\boldsymbol{b}) = S(0)\exp\begin{pmatrix} -b_{xx}D_{xx} - b_{yy}D_{yy} - b_{zz}D_{zz} \\ -2b_{xy}D_{xy} - 2b_{xz}D_{xz} - 2b_{yz}D_{yz} \end{pmatrix} \quad [7.12]$$

where the baseline signal intensity $S(0)$ is assumed to be take an arbitrary value of 1000. The simulated DWIs can then be used to calculate the diffusion tensor by multivariate linear regression and the various quantitative measurements can be obtained.

To establish the effects of noise on the diffusion measurements, the simulation procedure described above must be repeated with the individual DWIs being corrupted by increasing levels of simulated noise. Therefore, random measurement errors on the voxel intensities in each of the DWIs which constitute the DTI experiment were calculated by adding the noise free signal $S(\boldsymbol{b})$ to the real part of a set of normally distributed, complex random numbers with zero mean and standard deviation scaled to the desired root mean squared noise level. The noisy amplitude signal is then obtained by taking the magnitude of this new complex number (115,129). The SNR can then be calculated for each of the DWIs that constitute the DTI experiment and an average SNR taken for the experiment (SNR_{av}) (129). The DNR can then be calculated as the SNR in the calculated $tr(\boldsymbol{D})$ image (DNR_{av}). Due to the uniformity of the phantom, the SNR in any of the images can just be estimated by taking the ratio of the mean signal to the standard deviation within the synthetic phantom.

7.4.3 The effects of noise on measurements of diffusion

The Monte Carlo simulations described in §7.4.2 were performed to observe the effects of noise on measurements of ADC_{av} and anisotropy. The effect of noise on ADC_{av} and A_{sd} was observed for different diffusivities in the range 0.1 - 5.0 × 10^{-3} mm^2s^{-1} with a fixed b-value of 900 smm^{-2} and also for different b-values in the range 500 - 1500 smm^{-2} with a fixed diffusivity of 1.0 × 10^{-3} mm^2s^{-1}. The range of diffusivities chosen more than adequately covers the full range likely to be encountered in human brain imaging, while the b-value range covers typical values used in clinical diffusion imaging. Also, the effect of noise contamination on A_{sd} was observed for different shapes of cylindrical diffusion environments (i.e. different

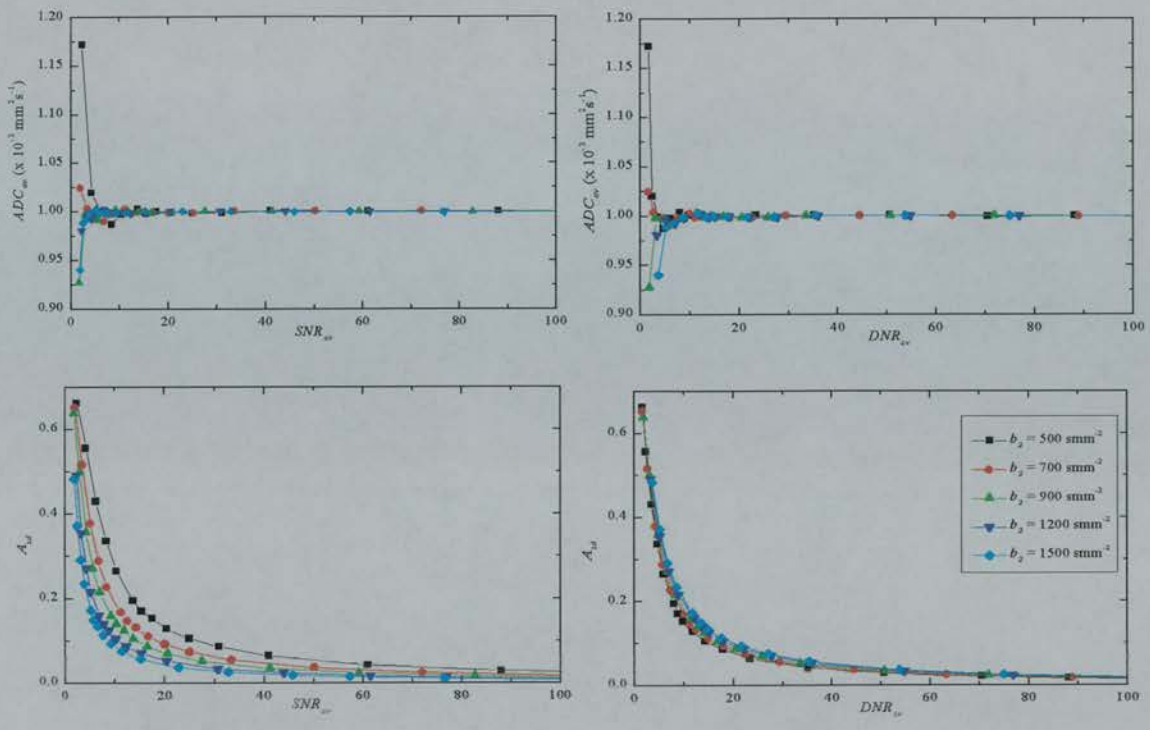


Figure 7.4 The effects of noise on measurements of ADC_{av} (top row) and A_{sd} (bottom row) for simulated images with $tr(\boldsymbol{D}) / 3 = 1.0 \times 10^{-3}$ mm^{2}s^{-1} and $A_{sd} = 0$, acquired at different b_2-values, with $b_1 = 0$. The noise contamination is described in terms of SNR_{av} (left) and the DNR_{av} (right). Error bars have been omitted for clarity.

degrees of anisotropy). The procedures were repeated for the different tensor acquisition schemes described in §7.4.2 and the noise was considered in terms of both DNR_{av} and SNR_{av} in each of the situations.

The effect of altering the b-value on measurements of ADC_{av} and A_{sd} is illustrated in Fig. 7.4. The graphs illustrate that typical measurements of ADC_{av} are relatively insensitive to noise contamination, except at very high degrees of noise contamination ($DNR_{av} \approx SNR_{av} < 5$). However, measurements of A_{sd} are highly susceptible to noise contamination and become markedly increased, even for relatively small degrees of noise contamination. Furthermore, the value of SNR_{av} at which the anisotropy is significantly increased is highly dependent on the b-value used, whereas DNR_{av} provides a measurement that is largely independent of the b-value, over the typical range of parameters studied.

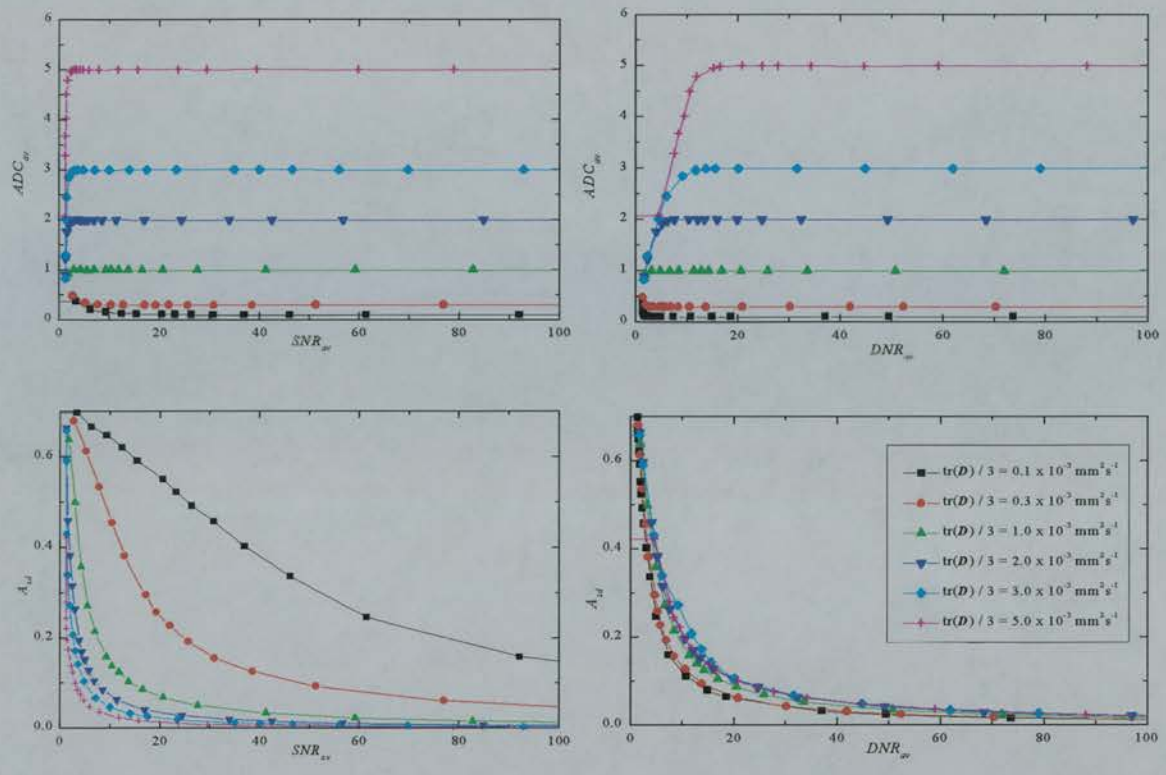


Figure 7.5 The effects of noise on measurements of ADC_{av} (top row) and A_{sd} (bottom row) for simulated images with variable $tr(\boldsymbol{D})$ / 3, acquired at $b_1 = 0$ and $b_2 = 900$ smm^{-2} with $A_{sd} = 0$. The noise contamination is described in terms of SNR_{av} (left) and the DNR_{av} (right). Error bars have been omitted for clarity.

It is interesting to observe the measurement of ADC_{av} at high levels of noise contamination. In §7.3.1 it was demonstrated that the optimum diffusion-weighting occurs when $\xi_{max} = ADC\Delta b \approx 1.1$. In this simulation $tr(\boldsymbol{D})$ / 3 is fixed at 1.0×10^{-3} mm^{2}s^{-1} and the optimum effective b-value therefore occurs at around 1100 smm^{-2}. For b-values much greater than this, there will be significant attenuation of the b_2 image and the signal will approach the level of the background noise (as for S_{CSF} in Fig. 7.6). At high noise levels, the measurement errors are then likely to lead to an increase in the signal of the b_2 image because the signal cannot take negative values and an underestimation of ADC_{av} therefore results (as in Case 1 & 2 in Fig. 7.6). Similarly for b-values much lower than optimum, there will be insufficient signal attenuation of the b_2 image relative to the b_1 image. If the measurement errors are sufficient such that some of the b_2 image pixels have higher signal than the b_1 image (see Fig. 7.6, Case 1 & 3), then a negative ADC may result, which is physically

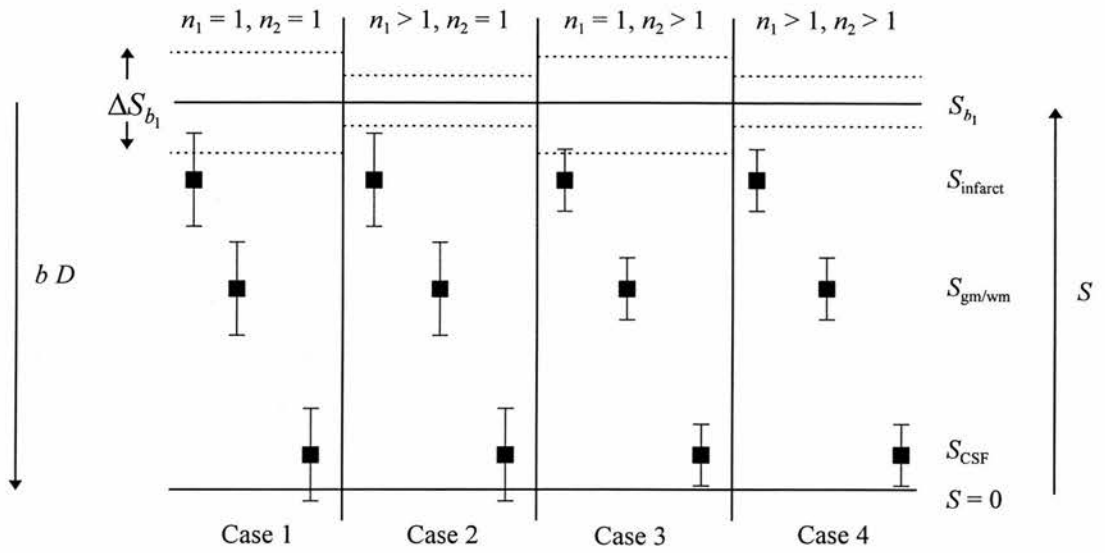


Figure 7.6 A schematic illustration of how the diffusion attenuation and signal averaging affect the reliable measurement of diffusivities. A two-point scheme is illustrated with n_1 acquisitions at b_1 and n_2 acquisitions at b_2. The diagram illustrates the effects of different signal averaging of the b_1 and b_2 images on the measurements of signals from simulated ischaemic infarcts (S_{infarct}), grey or white matter ($S_{\text{gm/wm}}$) and CSF (S_{CSF}), assuming a constant S_{b_1}. The horizontal dotted lines represent the error in the measurement of S_{b_1}, which can be reduced by signal averaging, i.e. increasing n_1. Similarly, the error bars on S_{infarct}, $S_{\text{gm/wm}}$ and S_{CSF} represent the noise induced errors for the measurement obtained at b_2 which can be reduced by increasing n_2.

meaningless. These pixels are therefore ignored in the ADC calculation and so the average signal in the b_2 image will be lower than expected, resulting in an over-estimation of the resulting ADC_{av}. These trends are observed in Fig. 7.5 for high degrees of noise contamination, where the measurement errors are likely to be sufficient to cause these effects.

Figure 7.5 illustrates the effect of noise on measurements of ADC_{av} and A_{sd} for different isotropic diffusivities at a fixed b-value of 900 smm^{-2}. Again, ADC_{av} is relatively insensitive to noise contamination, except for very low SNR_{av} and DNR_{av}, whereas A_{sd} shows a high dependence on the level of noise. The measurements of ADC_{av} are either over or underestimated at low SNR_{av} or DNR_{av} as before, governed by the degree of signal attenuation relative to the optimum obtained from Eq. 7.5. It is also interesting to observe that the value for DNR_{av} at which a reliable diffusion measurement is obtained is relatively dependent on $tr(\boldsymbol{D})$, when compared to SNR_{av}.

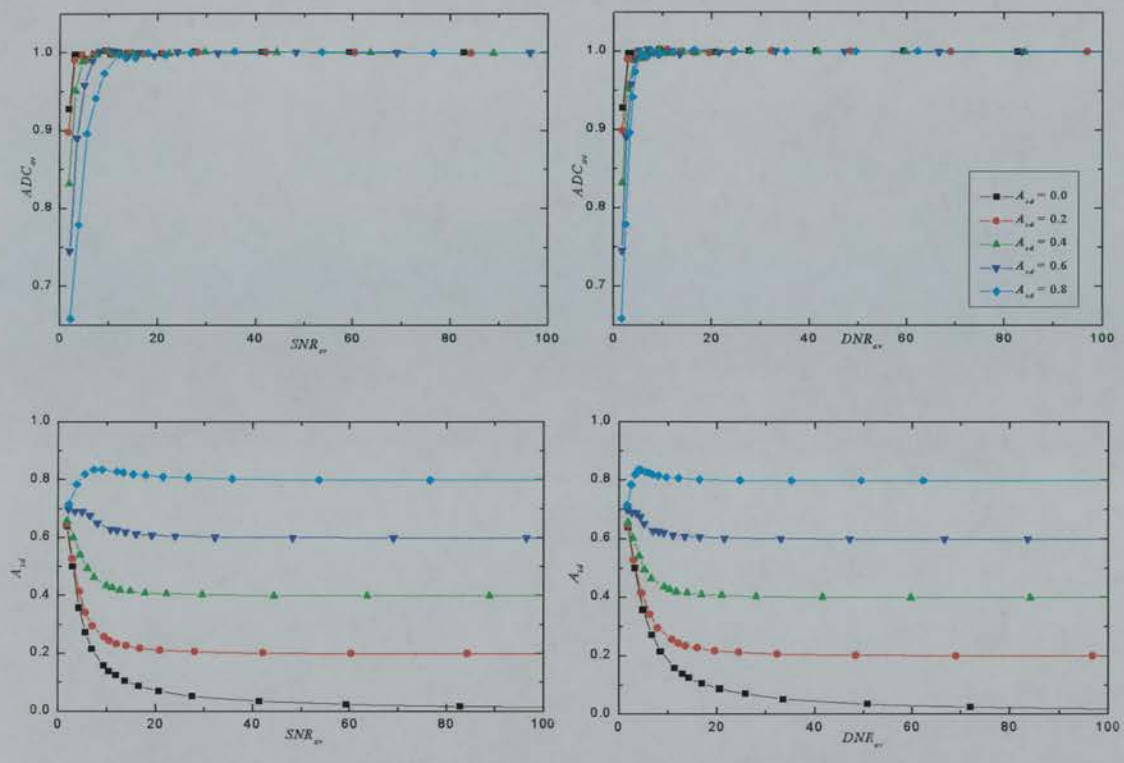


Figure 7.7 The effects of noise on measurements of ADC_{av} (top row) and A_{sd} (bottom row) for simulated images with $tr(\boldsymbol{D}) / 3 = 1.0 \times 10^{-3}$ mm^{2}s^{-1} and variable anisotropy, acquired at $b_1 = 0$ and $b_2 = 900$ smm^{-2}. The noise contamination is described in terms of SNR_{av} (left) and DNR_{av} (right). Error bars have been omitted for clarity.

However, this dependency only becomes noticeable when $DNR_{av} < 10$, which is considerably below the value at which acceptable noise independent anisotropy measurements are obtained. The most important feature of Fig. 7.5 is the extremely high dependency of the A_{sd} vs. SNR_{av} relationship on $tr(\boldsymbol{D})$. This is extremely important because it precludes the definition of a generally applicable value for SNR_{min}, as any chosen value will be highly dependent on $tr(\boldsymbol{D})$, even over typical values obtained in vivo. In contrast, the A_{sd} vs. DNR_{av} relationship shows little variation with $tr(\boldsymbol{D})$.

The effect of noise on measurements of ADC_{av} and A_{sd} are also observed for different axisymmetric ellipsoids, in Fig 7.7. Again, ADC_{av} is relatively insensitive to the effects of noise contamination, except at very high degrees of noise contamination ($DNR_{av} \approx SNR_{av} < 10$), while A_{sd} shows a much higher sensitivity to the effects of noise. A slightly greater variability with anisotropy is shown for the

ADC_{av} vs. SNR_{av} relationship than the equivalent DNR relationship, although this occurs because the b-value and ADC_{av} are close to the optimum. As can be seen in Fig. 7.4 and 7.5, the ADC_{av} vs. DNR_{av} relationship becomes more variable, if the chosen b-value and $tr(\boldsymbol{D})$ provide much greater signal attenuation than the optimum, given by Eq. 7.5. However, as before, A_{sd} shows a greater susceptibility to noise contamination than ADC_{av} and so the value for SNR_{min} and DNR_{min} is determined by the minimum value that produces acceptable noise independent anisotropy measurements. It is interesting to observe that measurements of isotropic diffusion ($A_{sd} = 0$) show a greater sensitivity to noise than measurements of anisotropic diffusion ($A_{sd} > 0$). This suggests that the relationships investigated in Figs. 7.4 and 7.5 will explore the 'worst case' of $A_{sd} = 0$ and therefore be applicable for finding an appropriate value for DNR_{min}.

The examples given in Figs. 7.4, 7.5 & 7.7 have all been illustrated for the case of uniform gradient sampling, with a zero b_1 value and non-zero b_2 value. However, the simulations were also performed using the conventional gradient sampling scheme and with non-zero b_1 values. No noticeable difference was observed in the form of the plots for the different sampling schemes. Also, the plots were found to be affected only by the Δb value chosen and not on the actual b_1 and b_2 values. So, even though the actual values for SNR_{av} and DNR_{av} are altered by varying b_1 and b_2, the degree of noise contamination remains independent of b_1 and b_2 provided that Δb remains constant.

7.4.4 The minimum DNR value for in vivo imaging

The Monte Carlo simulations performed in §7.4.3 demonstrate that measurements of A_{sd} are far more susceptible to the effects of noise contamination than measurements of ADC_{av}. Therefore, in order to obtain acceptable noise independent diffusion measurements (both magnitude and anisotropy), the value of SNR_{min} and DNR_{min} must be determined as the minimum value of SNR_{av} and DNR_{av} that produce a noise independent measurement of A_{sd}. However, as has been demonstrated, the A_{sd} vs. SNR_{av} relationship demonstrates a high degree of dependence on experimental parameters, particularly values for Δb and $tr(\boldsymbol{D})$.

Therefore, it is not possible to determine a value for SNR_{min} (as has been previously suggested (129)) that will be generally applicable, as a given value will only be valid for those specific experimental parameters. In contrast, the A_{sd} vs. DNR_{av} relationship shows very little dependence on b-value and $tr(\boldsymbol{D})$, at least over the typical range encountered in vivo. It can be seen from the relationships plotted in Figs. 7.4, 7.5 & 7.7, that a suitable value for DNR_{min} would be around 50, as any measurements having a greater DNR would be largely independent of noise contamination.

7.5 Generalising the optimisation scheme

The simple error analysis of the diffusion attenuation expression (Eq. 3.16) was shown in §7.2 and §7.3 to result in a method for optimising the acquisition scheme for a single diffusion measurement. The principal limitation of this method is that the acquisition scheme only maximises the DNR for an individual diffusivity. Also, the optimisation procedure only considers a single diffusion measurement and does not consider optimising a whole diffusion experiment, where several gradient directions are applied, as in the orthogonal, tetrahedral or tensor acquisition methods. This section aims to address these limitations to provide an optimisation scheme suitable for typical in vivo diffusion imaging protocols.

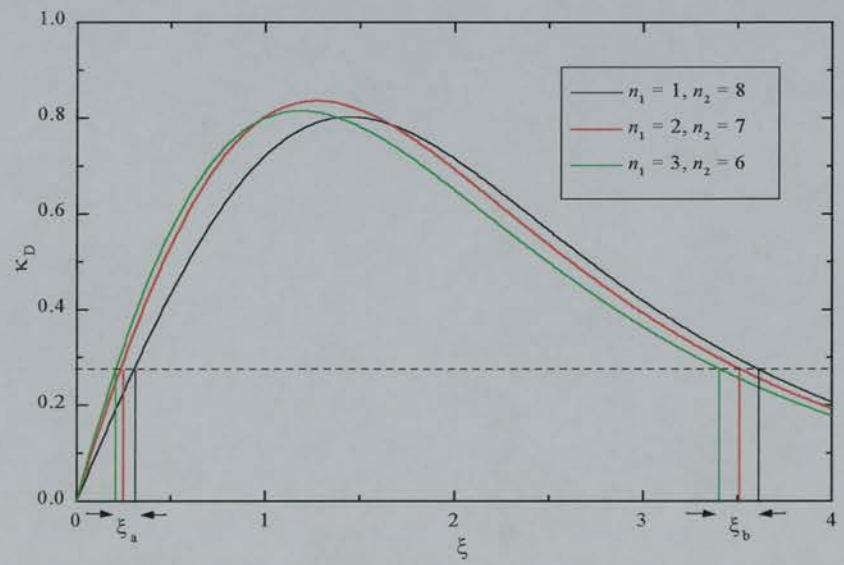


Figure 7.8 Illustration of the κ_D vs. ξ relationship for different n_1 and n_2 combinations with $m = 9$, demonstrating how n_1 and n_2 can be varied to accommodate a different range of diffusivities $\{ADC_a, ADC_b\}$ for a given m, where $ADC_a = \xi_a / \Delta b$ and $ADC_b = \xi_b / \Delta b$.

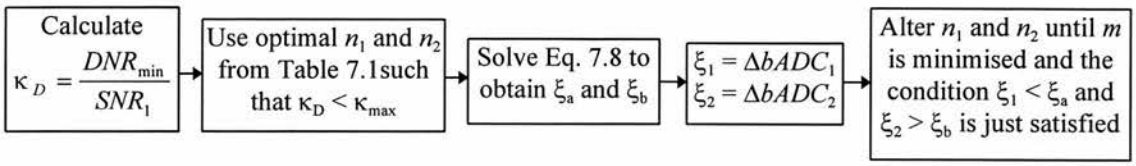


Figure 7.9 Flow chart illustrating the optimisation procedure to find the number of acquisitions n_1 and n_2 that minimise the overall imaging time (determined by $m = n_1 + n_2$), while providing acceptable diffusion measurements with $DNR > DNR_{min}$ in all voxels, over a chosen range $\{ADC_1, ADC_2\}$.

By following the procedure summarised in Fig. 7.3, an optimum value for Δb can be found such that the sensitivity κ_D can be 'optimised' to within 60% of κ_{max} for a range of diffusivities. However, the signal averaging scheme given in Table 7.1 was devised to produce the maximum κ_D value for a given m. Therefore, using this signal averaging scheme may not necessarily be the optimum averaging scheme for ensuring that a *range* of diffusivities have an acceptable DNR above DNR_{min}. This can be seen from Figs. 7.1 and 7.2, whereby the shape of the κ_D vs. ξ curves are altered as n_1 and n_2 are varied. It may be desirable, for example, to have a curve that rises and falls very steeply, to encompass a narrow range of diffusivities. Similarly, a greater sensitivity to higher diffusivities may be required resulting in a form of the curve that retains high κ_D sensitivity at large ξ values. Examples are given in Fig. 7.8 showing how slightly different n_1 and n_2 values alter the interval $\{\xi_a, \xi_b\}$.

The range of diffusivities typically found in human brain imaging of acute ischaemic stroke is $0.3 < tr(\boldsymbol{D}) / 3 < 3.0 \times 10^{-3}$ mm^{2}s^{-1} (127,167) and so it is desirable to find an optimised acquisition scheme that has $DNR_{min} > 50$ over this range of diffusivities. In practice, on many clinical scanners the b-value is fixed because clinical groups do not have access to pulse sequences, or the expertise to modify them on site. Therefore, in this study the optimised acquisition scheme is obtained for typical b-values used in many clinical studies, although the optimum b-value is also discussed. So, with the range of diffusivities selected and a fixed b-value chosen, n_1 and n_2 can be optimised such that the combination of n_1 and n_2 is found that minimises m, but still enables a $DNR_{min} > 50$ over $0.3 < tr(\boldsymbol{D}) / 3 < 3.0 \times 10^{-3}$ mm^{2}s^{-1}. The procedure adopted is summarised in Fig. 7.9 and the optimised acquisition schemes are presented for various b-values in Table 7.2.

Δb	SNR_1	κ_D	m	n_1	n_2	ADC_a	ADC_b
700	61	0.133	2	1	1	0.30	4.52
	50	0.164	4	2	2	0.26	4.81
	40	0.204	5	3	2	0.30	4.36
	30	0.272	9	3	6	0.30	4.90
	20	0.408	19	8	11	0.30	4.68
	10	0.816	73	29	44	0.30	4.68
900	50	0.164	2	1	1	0.29	3.21
	40	0.204	4	1	3	0.28	3.71
	30	0.272	6	2	4	0.29	3.48
	20	0.408	12	5	7	0.30	3.28
	10	0.816	38	18	30	0.30	3.33
1200	84	0.098	2	1	1	0.13	3.01
	50	0.164	4	1	3	0.17	3.04
	40	0.204	6	1	5	0.19	3.08
	30	0.272	9	1	8	0.25	3.03
	20	0.408	20	2	18	0.27	3.03
	10	0.816	76	7	69	0.28	3.00

Table 7.2 Optimised acquisition scheme with a non-zero b_1, and Δb = 700, 900 and 1200 smm^{-2}, given for different values of SNR_1. It can be seen that the optimum solutions for n_1 and n_2 are highly dependent on the b-value used and that these solutions are often considerably different to those given in Table 1.

It can be clearly seen from Table 7.2 how the optimum choice of n_1 and n_2 can be considerably different from the values given in Table 7.1, for a given m. This was illustrated in Fig. 7.8, for m = 9, which corresponds to the cases in Table 7.2 where Δb = 700 smm^{-2}, Δb = 1200 smm^{-2} and SNR_1 = 30. For these examples κ_D = 0.272, which can be seen to be considerably lower than 60% of $\kappa_{max} \approx 0.8$. However, satisfactory anisotropy measurements will be obtained because the signal averaging scheme ensures that $DNR > DNR_{min}$ for all ADCs in the range $\{ADC_a, ADC_b\}$.

The optimisation scheme presented in Table 7.2 uses a simple isotropic model to account for a diffusion tensor acquisition protocol. For an in vivo two-point diffusion-weighting scheme, a non-zero baseline b_1 value must be used to overcome IVIM perfusion contamination (19,116,175). This is necessary so that $\ln\left(S_{b_1} / S_{b_2}\right)$ displays a linear relationship with Δb (Eq. 7.1), which is essential for the error analysis presented in §7.2 and hence the optimisation scheme to be valid (175).

Therefore, it is assumed that twelve diffusion images are obtained, six at b_1 and six at b_2. The effective SNR_1 of the diffusion experiment (SNR_{eff}) is then assumed to be

$$SNR_{eff} = \sqrt{6} SNR_1 \qquad [7.13]$$

because the six diffusion images acquired at b_1 are assumed to lead to a $\sqrt{6}$ improvement in 'baseline' SNR. Similarly, there are six images acquired with b-value b_2 and so the six individual ADCs calculated from these six images are effectively summed to give the $tr(\boldsymbol{D})$ map. Therefore, it is assumed that the effective DNR_{min} required (DNR_{eff}) is

$$DNR_{eff} = DNR_{min} / \sqrt{6} \qquad [7.14]$$

Therefore, the κ_D value required to obtain a $tr(\boldsymbol{D})$ / 3 image with a $DNR > DNR_{min}$, using this acquisition scheme is

$$\kappa_D = \frac{DNR_{eff}}{SNR_{eff}} = \frac{DNR_{min}}{6SNR_1} \qquad [7.15]$$

The signal averaging assumptions above, effectively assume an isotropic model, whereby signal intensities only differ between b_1 and b_2 images and do not depend on the direction of the diffusion gradient. However, provided that the minimum ADC in any given measurement is greater than ADC_a, and the maximum is less than ADC_b, then errors in the diffusion measurements would not be expected to result. These assumptions are tested with in vitro and in vivo data in §7.6 and the limitations discussed in more detail.

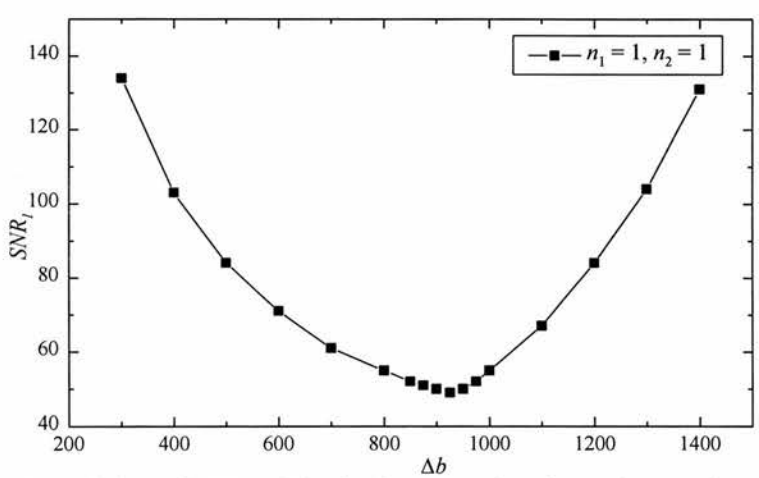


Figure 7.10 The variation of SNR_1 with Δb, demonstrating the optimum $\Delta b \approx 900$ smm^{-2} which requires the minimum SNR_1, while achieving the required $DNR_{av} > 50$ over the range of human brain diffusivities.

As was discussed previously, it is often the case that the b-value cannot be altered, in a clinical environment. Therefore, the optimisation scheme was presented in Table 7.2 for a typical range of b-values used in clinical studies. The results are also presented for a variety of SNR_1 values, to account for hardware and pulse sequence induced variations. However, it can be seen that using a Δb of 900 smm^{-2} is more optimum, than a Δb of 700 or 1200 smm^{-2}, requiring fewer acquisitions to measure accurately the required range of diffusivities $\{ADC_1, ADC_2\}$, at a given SNR_1. Alternatively, the value of SNR_1 can be observed for $n_1 = n_2 = 1$, and the optimum b-value will result in the minimum SNR_1 value. The relationship between SNR_1 and Δb is shown for $n_1 = n_2 = 1$ in Fig. 7.10. The optimum Δb occurs at slightly over 900 smm^{-2}, for $0.3 < tr(\boldsymbol{D}) / 3 < 3.0 \times 10^{-3}$ mm^{2}s^{-1}, resulting in the minimum SNR_1 that still satisfies $DNR > DNR_{min}$. This finding is somewhat fortunate because this b-value is readily achievable using conventional clinical systems.

7.6 Experimental verification

7.6.1 In vitro evaluation

The theory and simulations performed in previous sections, suggest that the optimisation scheme should enable an accurate diffusion analysis, over the range of diffusivities encountered in human brain imaging, provided that the acquisition scheme in Table 7.2 is followed. The aims of this section are to verify the effectiveness of the optimisation scheme using a series of phantom studies. The images were acquired using the Elscint Prestige 1.9T clinical scanner, with the custom developed imaging sequences and post-processing procedures discussed in §4.3.2 and §4.5, respectively. The EPI diffusion pulse sequences were modified to perform both the conventional and uniform tensor sampling schemes, described in §7.4.2, by adding gradient pulses along appropriate directions (24,83,85). A range of b-values were also provided by altering the magnitude of the diffusion gradient pulses as determined by Eqs. 3.16. The b-matrix was then calculated numerically from the sequence timing parameters. The gradient pulses were applied with magnitudes $G_0 = 0$, 3.78, 6.55, 10.00, 10.69, 11.34, 11.95, 13.09, 13.63 mTm^{-1},

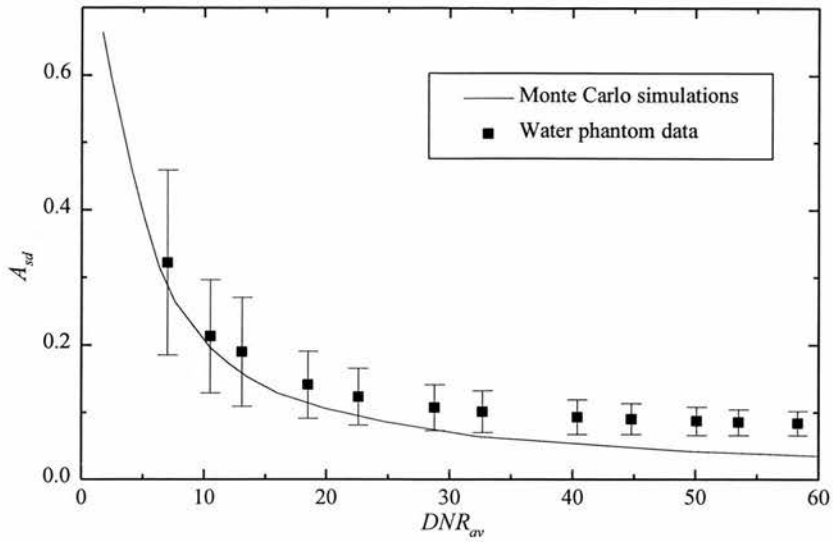


Figure 7.11 Isotropic water phantom data which provides experimental verification of the relationship between A_{sd} and DNR_{av}, demonstrating the hardware induced anisotropy offset.

leading to approximate scalar *b*-values of 0, 100, 300, 700, 800, 900, 1000, 1200 and 1300 smm^{-2}, respectively.

The first stage of the evaluation involves the verification of the relationships between A_{sd} and DNR_{av}. It is not realistic to verify all of the features discussed in §7.4.3, principally due to the difficulty in obtaining phantoms which mimic realistic in vivo diffusion environments. A water phantom was used to illustrate the A_{sd} vs. DNR_{av} relationship, for the 'worst case' of isotropic diffusion. The diffusion tensor was obtained from a series of diffusion-weighted images acquired with scalar *b*-values of 0 and 900 smm^{-2}, using the uniform sampling scheme (83). A region of interest was defined in the centre of the phantom image and an estimate of SNR_1 was obtained from the mean signal ($\bar{S}$) in the ROI, divided by the standard deviation (σ). A value for SNR_1 was obtained for all images acquired with $b = 0$ and the average was taken to be the SNR_1 of the experiment. The DNR was altered by varying n_1 and n_2 such that for each set of images DNR_{av} was calculated from the mean signal divided by the standard deviation in the $tr(\boldsymbol{D})$ / 3 image. The mean A_{sd} within the ROI is then calculated for each set of n_1 and n_2 values, and the plot of A_{sd} vs. DNR_{av} is obtained, as illustrated in Fig. 7.11. The experimental water phantom data is shown alongside the simulated curve for $tr(\boldsymbol{D})$ / 3 = 2.0 × 10^{-3} mm^{2}s^{-1} and A_{sd} = 0,

demonstrating a good correlation between the simulated and experimental data. However, an offset between the experimental and simulated A_{sd} occurs for higher DNR_{av} values. This anisotropy increase is likely to result from hardware induced artefacts, arising from incorrect gradient calibrations.

A second series of phantom studies was performed to establish whether the acquisition scheme presented in Table 7.2 succeeded in achieving DNR_{min} over the range of ADCs required. In order to test the acquisition method, a 'warm' water phantom was used at a temperature of 300K, with a corresponding diffusivity of 2.3 $\times 10^{-3}$ mm^2s^{-1} (67). It would be ideal to perform the measurements on phantoms with diffusivities at the extremes of the required range e.g. 0.3 and 3.0 $\times 10^{-3}$ mm^2s^{-1}. However, to produce these variations by temperature alone is highly impractical due to the difficulty in maintaining thermal equilibrium with the surroundings; a condition that is necessary to avoid temperature and convection current related artefacts (69,121). A gel phantom could be constructed to achieve the desired lower diffusivity, however, such phantoms are difficult to design (176). A simplified phantom was constructed using a mixture of gelatine and copper sulphate doped-water, but it was only possible to produce a minimum diffusivity of around 0.8 $\times 10^{-3}$ mm^2s^{-1} and the phantom images then displayed poor SNR, due to the low concentration of water. Nevertheless, using the 'warm' water phantom should provide a reasonable verification of the optimisation scheme, as the κ_D vs. ξ relationship (Fig. 7.8) is varying relatively slowly at high ξ. Therefore, the DNR measured at $tr(\boldsymbol{D}) / 3 = 2.3 \times 10^{-3}$ mm^2s^{-1} is likely to be only slightly higher than that at $tr(\boldsymbol{D}) / 3 = 3.0 \times 10^{-3}$ mm^2s^{-1} and so should provide a reasonable indication as to whether the DNR_{min} criterion is satisfied.

The experiments were performed with a fixed $b_1 = 100$ smm^{-2} and a variable $b_2 = 800, 1000, 1300$ smm^{-2}, to produce the values for Δb given in Table 7.2. The number of acquisitions n_1 and n_2 were then varied according to the optimised solutions in Table 7.2. A plot of A_{sd} vs. DNR_{av} was obtained, as described previously, to verify that the anisotropy measurements were independent of DNR_{av}, for values of m that are greater than the minimum value, given in Table 7.2. In principle, it would be possible to evaluate the optimisation scheme by acquiring a single DNR_{av} map with the optimum n_1 and n_2 combination and ensuring that $DNR_{av} > 50$. However,

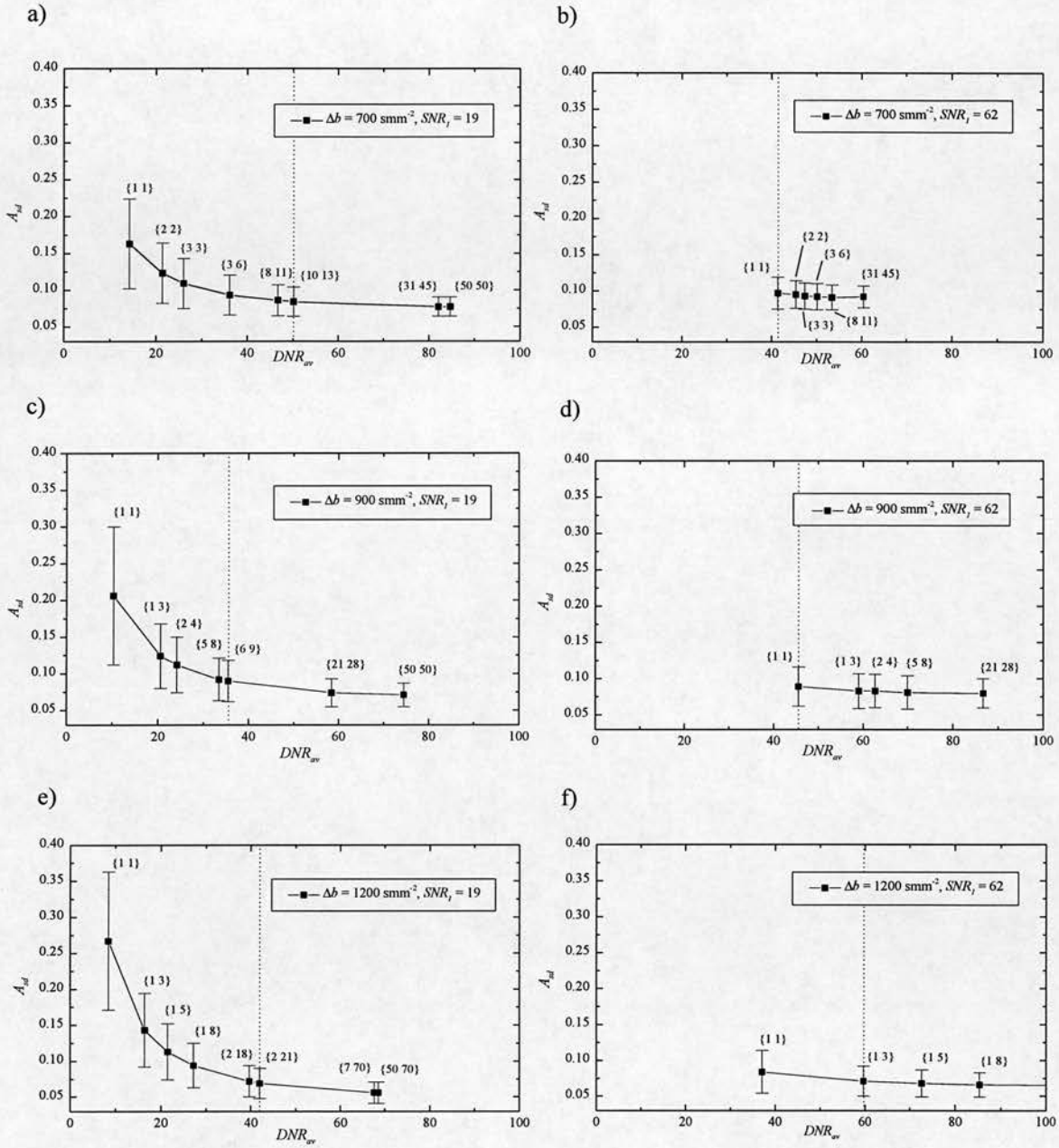


Figure 7.12 Experimental verification of the effectiveness of the optimisation scheme presented in Table 7.2. The DTI experiment was performed with Δb = 700 (top), 900 (middle) and 1200 smm^{-2} (bottom), for various acquisition combinations $\{n_1\ n_2\}$ and at two different SNR_1 values of 19 (left) and 62 (right). The optimised $\{n_1\ n_2\}$ for the given SNR_1 and Δb is indicated by the vertical dotted line. These optimum $\{n_1\ n_2\}$ combinations show a good correlation with the point at which A_{sd} becomes largely independent of DNR_{av}.

due to image artefacts resulting in signal variations within the images, the measured value for DNR_{av} is usually lower than expected. It is therefore more sensible to observe the plot of A_{sd} vs. DNR_{av} and ensure that further signal averaging causes no significant alteration in the measured value for A_{sd}.

The results from the phantom experiments are shown in Fig. 7.12, presented for two different 'baseline' SNR_1 values, of approximately 20 and 60. The SNR_1 value was altered by acquiring two different data sets with different slice thickness' of 3mm and 15mm, respectively. The DNR_{av} values measured using the optimum n_1 and n_2 combinations for the given SNR_1 and Δb are illustrated by the vertical dotted lines in Fig. 7.12. As described previously, the measured DNR_{min} is usually slightly less than 50, a feature which is likely to result from artefacts causing signal variation in the image that is not a result of noise contamination. However, the measured DNR_{av} at the optimum n_1 and n_2 correlates well with the DNR_{av} at which A_{sd} becomes independent of noise, i.e. the 'plateau' is reached. For the data obtained with SNR_1 = 19, the optimum $\{n_1\ n_2\}$ were calculated as being {10 13}, {6 9} and {2 21} for Δb = 700, 900 and 1200 smm^{-2}, respectively. This was performed because the optimum combination $\{n_1\ n_2\}$ varies greatly at low SNR_1, as indicated in Table 7.2. For SNR_1 = 62, this was not necessary because SNR_1 is sufficient such that $\{n_1\ n_2\}$ does not vary greatly with small changes in SNR_1. However, allowing for experimental errors in the measurement of SNR_1 and the relatively arbitrary choice of DNR_{min} = 50, Fig. 7.12 would suggest that it is sufficient to approximate SNR_1 with the nearest value given in Table 7.2. It is interesting to observe the effects of using non-optimum combinations of n_1 and n_2 on the measurements. The most striking example is in Fig. 7.12e, where the optimum combination with m = 77 of {7 70} produces a DNR_{av} of 67.6, while the non-optimum combination of {50 70} gives a DNR_{av} of 68.5, demonstrating how valuable imaging time can be saved by using optimum signal averaging schemes, rather that adopting the traditional schemes where $n_1 = n_2$.

7.6.2 In vivo evaluation

The optimisation scheme was tested in vivo on a normal volunteer, and the resulting images were assessed both qualitatively and quantitatively. The images

were acquired with b_1 = 100 and b_2 = 1000 smm^{-2}, to overcome IVIM perfusion contamination (19,116) and provide the optimum Δb. A slice thickness of 6mm was used, giving SNR_1 = 35, and the DTI experiment was performed for various optimum n_1 and n_2 values, given by Table 7.2. The values for DNR_{av} and SNR_1 cannot be calculated as described in the 'in vitro' section because the brain does not consist of a homogenous diffusion environment. The signal magnitude is estimated, as before, from the mean signal within a ROI in the brain ($\overline{S}$), but the noise cannot be estimated from the standard deviation (σ) of this ROI, due to the inhomogeneity of the structures in the region. Therefore, a subtraction image is produced such that the image of interest is subtracted on a pixel-by-pixel basis from an 'identical' image obtained with the same imaging parameters. The error σ is then obtained from the standard deviation within the ROI placed on this subtraction image. This method is used because the two images should be identical and any differences between them will be due to noise.

A quantitative evaluation as to the effectiveness of the optimisation scheme was performed by obtaining the relationship between A_{sd} and DNR_{av} for a ROI covering the whole brain, as illustrated in Fig. 7.13. It can be seen that the measured DNR_{av} values are considerably lower than those obtained in the in vitro study. However, this may reflect the difficulty in obtaining reliable DNR_{av} measurements in vivo, particularly given the poor reproducibility of images obtained using single shot EPI techniques. Also, the error bars in Fig. 7.13 are generally larger than those obtained for the in vitro study illustrated in Fig. 7.12. These error bars are determined from σ, and therefore indicate a higher variability in the in vivo measurements. This variability is believed to result from the poor reproducibility of in vivo EPI images as is discussed in more detail in §9.3.2. Nevertheless, Fig. 7.13 suggests that the optimum DNR_{av} is sufficient to produce reliable A_{sd} measurements because increasing m to increase DNR_{av} above the optimum value (vertical dotted line), does not appear to alter A_{sd}. In contrast, below the optimum m, A_{sd} begins to rise, suggesting noise contamination is beginning to become significant. Unfortunately, it was not possible to investigate a wider range of DNR_{av} values, due to time constraints because the imaging procedure must be repeated twice for each n_1

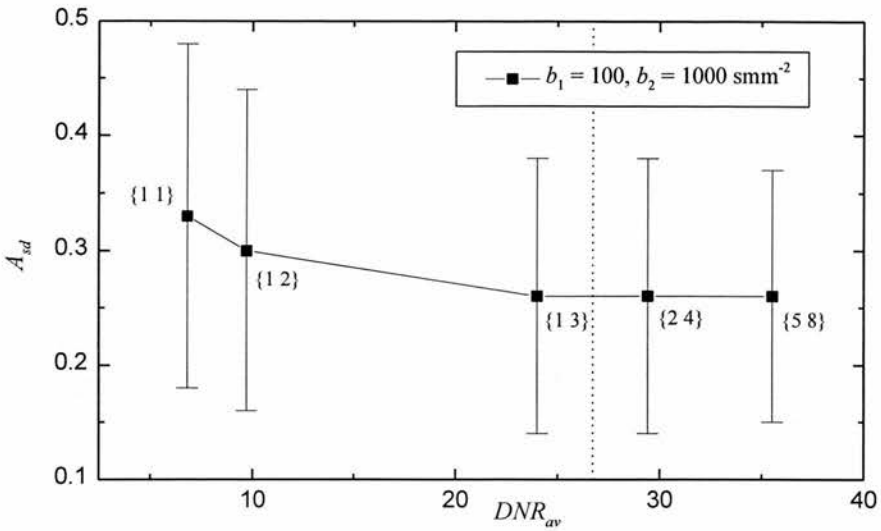


Figure 7.13 In vivo evaluation of the optimisation scheme presented in Table 7.2. The DTI experiment was performed on a normal volunteer, for various optimum $\{n_1\ n_2\}$ combinations and the plot of A_{sd} vs. DNR_{av} obtained. The baseline SNR_1 was 35, resulting in the minimum n_1 and n_2 combination occurring between {1 3} and {2 4}, as illustrated by the vertical dotted line.

and n_2 combination, in order that the subtraction image can be obtained and σ be reliably estimated.

Ideally, it would be desirable to obtain a verification of the optimisation scheme for the different regions of the brain individually, rather than taking an average measurement over the whole brain. This would verify that the scheme was providing a satisfactory DNR for different diffusion environments, such as CSF (high D, isotropic), white matter (intermediate D, anisotropic) and grey matter (intermediate D, fairly isotropic). Unfortunately, due to the poor reproducibility of the anisotropy measurements, in vivo, this was not possible as the measurement errors far outweighed the changes in anisotropy over these small regions. Regions of CSF may be expected to demonstrate poor DNR because they have $tr(\boldsymbol{D}) / 3 \approx 3.0 \times 10^{-3}$ mm^{2}s^{-1}, which corresponds to the upper limit of the optimisation scheme (ADC_2). However, for low b_1 values, CSF has a high SNR_1 value (relative to the whole brain average SNR_1 used in the optimisation scheme), so this should compensate for the low κ_D value and result in a reasonably high DNR. Similarly, ischaemic regions with a reduced ADC, may approach the lower limit ADC_1 of the optimisation scheme. This is unlikely to result in a poor DNR for lesions that are greater than 24 hours

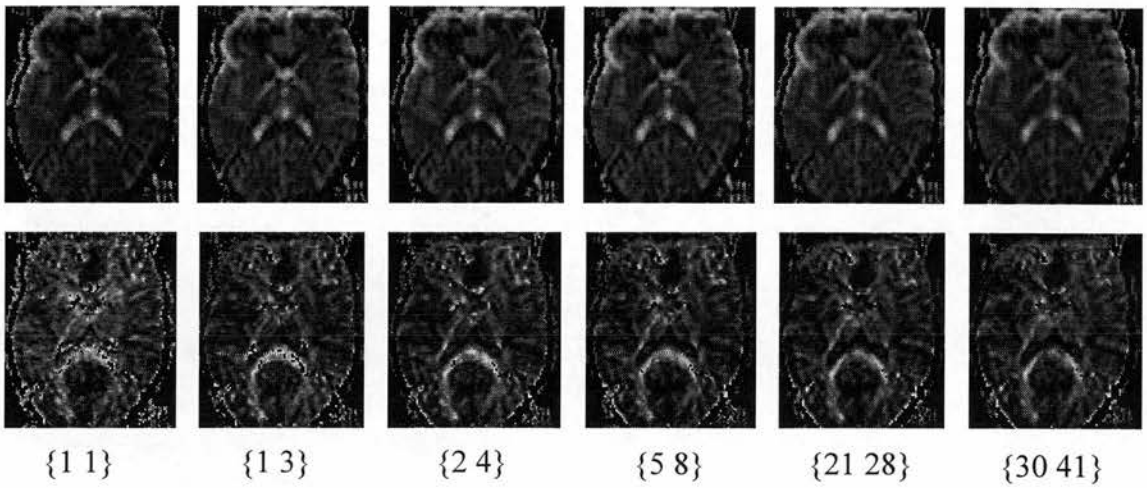


Figure 7.14 Qualitative evaluation of the effectiveness of the optimisation scheme at producing noise independent in vivo maps of ADC_{av} (top) and A_{sd} (bottom). ADC_{av} is seen to be relatively independent of $\{n_1\ n_2\}$. In contrast, the A_{sd} image acquired with {1 1} is brighter indicating a general elevation of A_{sd} indicative of noise contamination. However, by the {2 4} acquisition, further signal averaging has little effect on image quality, as predicted by Table 7.2 for $SNR_1 = 35$.

post-ictus, as these usually have elevated T_2 signal intensities and hence increased SNR_1 values. However, for highly acute lesions (< 24 hours post-ictus), the T_2 signal may not be elevated and therefore SNR_1 may be low and the resulting DNR may not meet the required condition of DNR_{min}. Similarly, for regions of highly anisotropic white matter, even though $tr(\boldsymbol{D}) / 3 \approx 1.0 \times 10^{-3}$ mm^{2}s^{-1}, an individual diffusion measurement along a given direction may be lower than ADC_1 and if SNR_1 is also low, the measurement may be susceptible to noise contamination. These limitations result from the isotropic model assumed in §7.5 and also by taking an average SNR_1 over the whole brain. Nevertheless, it is a relatively simple matter to alter the optimisation scheme to account for a greater range $\{ADC_1\ ADC_2\}$, at the expense of further signal averaging and a change in the optimum Δb. Table 7.2, illustrates how using a higher or lower b-value alters the range $\{ADC_1\ ADC_2\}$ that can be accurately measured.

To further verify the effectiveness of the optimisation scheme in vivo, a qualitative assessment of the images was also performed for maps of A_{sd} and ADC_{av}, calculated from sets of images acquired with different combinations of n_1 and n_2. The images are illustrated in Fig. 7.14, for $\{n_1\ n_2\}$ = {1 1}, {1 3}, {2 4}, {5 8}, {21

28}, {30 41}. The ADC_{av} images do not appear to be significantly altered by signal averaging, demonstrating the low sensitivity of ADC_{av} to noise contamination. However, the A_{sd} maps demonstrate significant improvements with signal averaging, as predicted by the theoretical considerations. It can be seen that the {1 1} image is of poor quality and is considerably brighter than the other images which is indicative of a noise induced increase in anisotropy. The {1 3} image is considerably improved and by the {2 4} image, there is little further improvement in image quality, even with signal averaging up to {30 41}. This is predicted by the optimisation scheme, which suggests that the DNR_{min} criterion is satisfied with a $\{n_1\ n_2\}$ combination between {1 3} and {2 4}. After this condition has been fulfilled, A_{sd} should be independent of noise and so little improvement in image quality would be expected with further signal averaging.

7.7 Conclusions

In this chapter, the theory and limitations of a previously derived diffusion-weighting optimisation scheme (175) have been discussed, resulting in recommendations being made to improve the procedure, which were implemented. The optimisation scheme requires the knowledge of DNR_{min}; a quantity that was not known or discussed in the original proposal. Therefore, in §7.4, Monte Carlo simulations were performed to enable DNR_{min} to be evaluated for typical in vivo situations. Previous studies have utilised the SNR as a measure of noise contamination (115,129), however this study demonstrated that the DNR provides a more reliable measure that is largely independent of the experimental parameters. One of the principal limitations of the original optimisation scheme was that the signal averaging scheme (Table 7.1) was optimised for a single diffusivity. Therefore, the procedure was altered to find the optimal signal averaging scheme for the typical range of diffusivities encountered in human brain imaging. The results were presented for several b-values and an optimum $\Delta b \approx 900$ smm^{-2} was found for human brain imaging. The theoretical predictions were then tested using a series of in vitro and in vivo experiments and good agreement was found between the theoretical and experimental data.

8

Techniques for Displaying Diffusion Data

8.1 Introduction

Chapters 6 and 7 have been concerned with investigating and developing acquisition strategies to enable the accurate evaluation of diffusion data in vivo. In diagnostic medical imaging, it is extremely important that the acquired data can be displayed clearly and efficiently, providing suitable contrast to highlight the necessary pathologies. Therefore, this chapter discusses some of the different image display techniques applicable to the visualisation of diffusion data. As has been shown in previous chapters, diffusion must be treated as a symmetric tensor quantity (24,25), resulting in the acquisition of six independent parameters which describe a diffusion ellipsoid (84,86). The visualisation process must somehow exploit these six parameters to provide the required physiological 'stain' (84) which highlights the pathological condition under investigation. This process often involves the calculation of scalar parameters from the tensor which exploit known physical characteristics, thereby reducing the six dimensional data set into a more manageable form, suitable for display on a computer screen, film or printout. Further techniques have also been developed using colour images, where the additive properties of primary RGB colours are exploited to display the multi-dimensional data set (125,177,178).

This chapter provides a summary of some of the conventionally used 'stains', such as ADC_{av} and A_{sd}, and demonstrates that these indices alone cannot characterise completely the size and shape of the diffusion ellipsoid. Therefore, the use of other indices to provide the necessary additional information is discussed, particularly the prolateness index (P) which was derived in §6.2.5 to provide an indication as to the degree of ellipsoid symmetry. The sensitivity and contrast of different anisotropy indices is also discussed, resulting in a method being derived for reducing the sensitivity to noise and improving contrast in diffusion anisotropy maps. The scalar parameters derived from the tensor (84) give an indication as to the size or shape of the diffusion ellipsoid, but give no indication as to its orientation. The development of fibre tract orientation maps and the potential for using DTI to map the main structural pathways within the brain are therefore discussed (177-181). The ability of diffusion ellipsoid images to provide a complete characterisation of the diffusion environment in a single image, is potentially of great value as a clinical aid (84,128). These ellipsoid images are presented and their usefulness in the clinical setting is discussed.

8.2 Two-dimensional diffusion maps

8.2.1 The magnitude of diffusion

The most simple images to display are the DWIs, which result directly from the signal attenuation given by Eq. 3.23. The contrast in these images will be dependent, to an extent, on the diffusivity, with higher D resulting in more signal attenuation. The image contrast is altered by varying the diffusion-weighting factor b. However, these images demonstrate contrast that is also determined by the original signal intensity (S_0) in the T_2-weighted 'baseline' image and the degree of diffusion attenuation is relative to this initial signal as determined by Eq. 3.23. For a high level of diffusion-weighting, these images do generally demonstrate lower signal intensities for higher diffusivities. However, there may still be substantial non-diffusion contrast, particularly for regions having a high T_2 signal and a low diffusivity. This 'T_2 shine through' results in an area of high signal in the diffusion-weighted images, that does not reflect the true diffusivity. This additional T_2 contrast

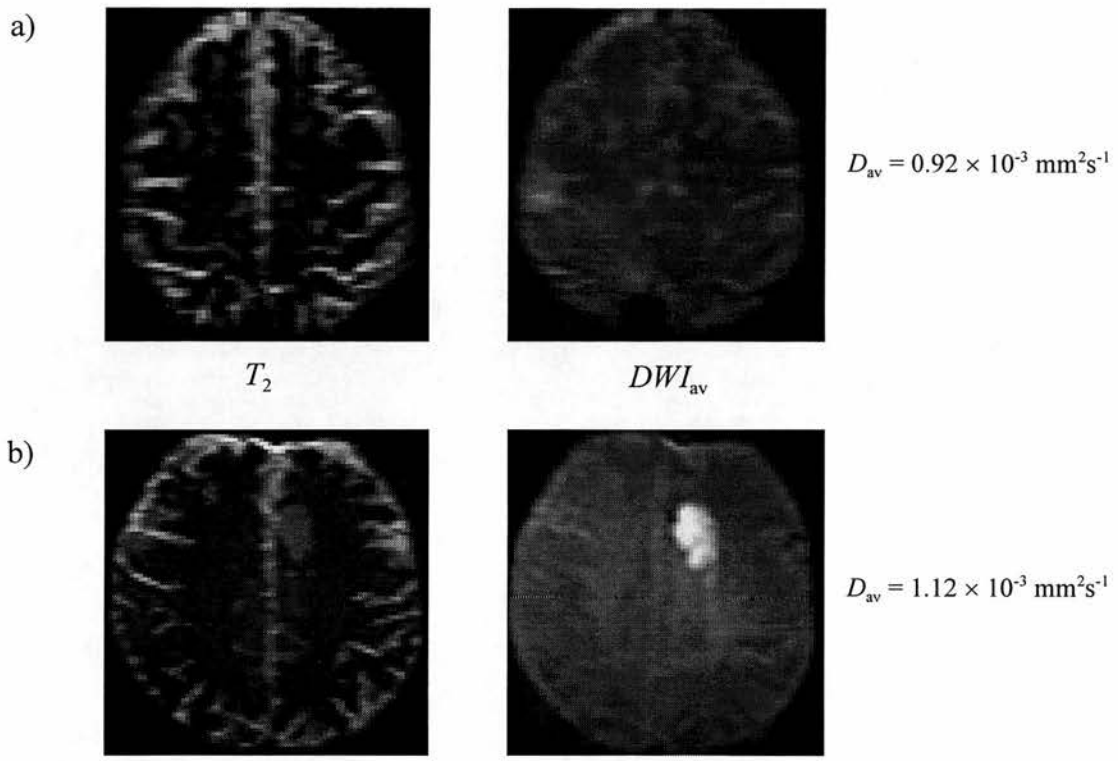


Figure 8.1 The 'T_2 shine through' effect, whereby the diffusion-weighted image (DWI_{av}) displays additional contrast due to T_2. The effect is illustrated for a) an acute lesion, with no noticeable elevation of T_2 signal and b) a later acute lesion, with a clearly elevated T_2 signal. The diffusivity (D_{av}) is slightly lower in a), so the signal should be higher in this image if the contrast were solely determined by the diffusion attenuation.

is useful for using DWIs to highlight acute stoke regions, which tend to have elevated T_2 signal intensities (except in the hyper-acute stages) and low diffusivities, as illustrated in Fig. 8.1. The combination of T_2 and diffusion contrast highlights these regions very effectively relative to the normal brain as was illustrated in Fig. 5.6.

While the DWIs are useful to provide a visualisation of the magnitude of diffusion, it must be remembered that the contrast is not exclusively determined by diffusion. It is therefore highly desirable to have diffusion maps where the contrast is determined solely by the value of D. These maps are obtained by calculating the diffusivity from two images acquired with different b-values, as in Eq. 4.3. The images are acquired with identical timing parameters, such that any difference in the signal intensities between the images is due to diffusion and the other contrast mechanisms can be eliminated. This allows for a quantitative assessment of the diffusivity, calculated along the direction of the diffusion gradient, which can be

evaluated in a given anatomical ROI. However, as a diagnostic image for identification of acute ischaemic lesions, maps of D may not be as useful as DWIs because the additional T_2 contrast is lost and the image displays poorer lesion-normal contrast. Nevertheless, the additional quantitative information is essential for verifying that the lesion corresponds to an area of real diffusion change and not just a 'T_2 shine through' effect.

8.2.2 The average size of the diffusion ellipsoid

To overcome the ambiguities arising from the directional dependence of the diffusion measurements, a quantitative measure of the average magnitude of the diffusion ellipsoid (D_{av}), is derived from the trace of the diffusion tensor. This is calculated from Eq. 3.44, and is often alternatively symbolised by $tr(\mathbf{D}) / 3$, ADC_{av} or $\overline{\lambda}$ (24,25,85). As was demonstrated in Chapter 6, D_{av} is a rotationally invariant quantity which can be acquired using either the orthogonal, tetrahedral or tensor encoding schemes. In theory, only measures of D_{av} derived using a DTI experiment will be rotationally invariant as they implicitly assume all gradient interactions (24). However, provided that these interactions are negligible for the imaging sequences used, then Eq. 3.14a will provide a good approximation to the true signal attenuation given by Eq. 3.23, resulting in a measure of D_{av} that is rotationally invariant, to

$$\boldsymbol{b}_0 = \begin{pmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.02 \\ 0.00 & 0.02 & 0.04 \end{pmatrix} \text{smm}^{-2} \qquad \boldsymbol{b}_x = \begin{pmatrix} 672.47 & 1.28 & 1.80 \\ 1.28 & 0.01 & 0.02 \\ 1.80 & 0.02 & 0.04 \end{pmatrix} \text{smm}^{-2}$$

$$\boldsymbol{b}_y = \begin{pmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 672.47 & 1.80 \\ 0.00 & 1.80 & 0.04 \end{pmatrix} \text{smm}^{-2} \qquad \boldsymbol{b}_z = \begin{pmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 0.01 & 1.29 \\ 0.00 & 1.29 & 673.52 \end{pmatrix} \text{smm}^{-2}$$

Figure 8.2 The b-matrices calculated for the baseline $\boldsymbol{b}_0$, and three orthogonal diffusion $\boldsymbol{b}_x$, $\boldsymbol{b}_y$ and $\boldsymbol{b}_z$ pulse sequences. The b-matrices were calculated numerically from Eq. 3.24 as described in §3.4.1.

within the bounds of experimental error. As was demonstrated in §5.3.2, this was the case for the Siemens orthogonal imaging sequences, where the *b*-matrix element used in Eq. 3.14a (either b_{xx}, b_{yy} or b_{zz}, to determine D_{xx}, D_{yy} or D_{zz}, respectively) dominates over all other tensor elements. This was also found to be the case for the Elscint EPI diffusion sequences as is demonstrated from the *b*-matrices shown in Fig. 8.2. Chapter 7 demonstrated how measurements of D_{av} were relatively insensitive to the effects of noise contamination. This further supports the widespread use of D_{av} in the clinical setting as reliable measurements can be obtained, even for techniques that produce images with poor SNR (129).

8.2.3 The shape of the diffusion ellipsoid

Various scalar parameters for describing the shape of the diffusion ellipsoid were presented in §3.6.2. These anisotropy indices provide a measure of the variation of the principal diffusivities. To enable the production of a reliable, rotationally invariant anisotropy 'stain', indices must be derived using the full information content of the diffusion tensor, as discussed in Chapter 6. However, while these DAIs provide an indication as to the degree of anisotropy, they do not provide an unambiguous representation of the shape of the diffusion ellipsoid. This was discussed in §6.2.5 and illustrated in Fig. 6.1, demonstrating how anisotropy indices do not distinguish between prolate and oblate diffusion. Therefore, two anisotropy indices are required to unambiguously determine the shape of the diffusion ellipsoids, these could be, for example, A_{sd} and P (as in §6.2.5), or A_{major} and A_{minor} (as in §6.2.2).

As the ellipsoid size and shape is described by three parameters, the eigenvalues λ_1, λ_2 and λ_3, it is logical to assume that two indices will be required to characterise the ellipsoid shape. Typically D_{av} is used to characterise the magnitude of the diffusion ellipsoid and two additional 'anisotropy' parameters are required to characterise the ellipsoid shape. As described in §6.2.5, the form of A_{sd} which conforms with the chemical definition such that $-0.5 < A_{sd} < 1.0$, provides a single index that uniquely characterises the shape of the diffusion ellipsoid. However, this form of A_{sd} still requires knowledge of a further parameter (e.g. P) to indicate

whether the positive or negative solution should be taken in Eq. 6.4. Furthermore, this form of A_{sd} does not provide such an easily interpretable quantitative physiological stain, as when its magnitude is taken.

8.3 Improving contrast and sensitivity in anisotropy maps

8.3.1 Introduction

The purpose of this section is to investigate the sensitivity of DAIs to changes in anisotropy, with a view to determining which indices are best suited for typical diffusion imaging situations, particularly in the human brain. The differences between the DAIs are illustrated using synthetic diffusion data and a mathematical concept for describing their sensitivity, leading to a methodology for producing diffusion anisotropy maps with improved contrast and sensitivity. It is well documented that measurements of diffusion anisotropy are extremely sensitive to noise, and its effect on the contrast in the different anisotropy maps is also investigated (115,129). The simulations resulted in the production of the gamma-variate anisotropy (A_{gv}) which was devised to provide improved contrast in anisotropy images of the human brain, while still retaining a low sensitivity to noise. Finally, maps of the different DAIs were obtained using in vivo human brain diffusion data to verify the theoretical predictions for anisotropy contrast and sensitivity. Throughout this study, the DAIs are compared relative to A_{sd} as this conforms to the chemical definition of anisotropy (182). As described above, this definition scales the anisotropy between -0.5 and 1.0 and in order that A_{sd} conforms with this definition, the sign of the square root in Eq. 6.4 must be chosen appropriately, such that prolate diffusion falls into the range {0 1} and oblate diffusion is in the range {0 -0.5}.

8.3.2 Theoretical and experimental methods

i) Diffusion Anisotropy Indices

The DAIs, A_{sd}, A_{vr}, A_{fa} and A_{er} were calculated on a pixel-by-pixel basis, in both simulated and in vivo data sets, using Eqs. 8.1 - 8.4, such that

$$A_{sd} = \frac{1}{\bar{\lambda}\sqrt{6}} \sum_{j=1}^{3} \sqrt{\left(\lambda_j - \bar{\lambda}\right)^2} \qquad [8.1]$$

$$A_{vr} = \frac{\lambda_1 \lambda_2 \lambda_3}{\bar{\lambda}^3} \qquad [8.2]$$

$$A_{fa} = \sqrt{\frac{3}{2\left(\lambda_1^2 + \lambda_2^2 + \lambda_3^2\right)}} \sum_{j=1}^{3} \sqrt{\left(\lambda_j - \bar{\lambda}\right)^2} \qquad [8.3]$$

$$A_{er} = \frac{\lambda_1 - \lambda_3}{\lambda_1} \qquad [8.4]$$

All of the above DAIs are nominally scaled into the range {0 1}. Furthermore, A_{sd}, A_{vr} and A_{fa} have the advantage that they are symmetrical with respect to the eigenvalues, whereas A_{er} requires sorting of the eigenvalues, which introduces an additional noise related bias, that must be corrected for (183,184).

ii) Simulations

Diffusion data were simulated with the eigenvalues of the diffusion tensor being chosen to represent axisymmetric diffusion covering the full range of anisotropy, for a diffusion tensor of constant trace. The DAIs were then calculated for each synthetic set of eigenvalues using Eqs. 8.1 - 8.4. The behaviour of the different DAIs relative to A_{sd} was then observed by plotting A_j as a function of A_{sd}, where j represents the subscripts *vr*, *fa*, *er* and *gv*. These plots can be interpreted such that the gradient, dA_j / dA_{sd}, represents a measure of the sensitivity of each of the indices to changes in A_{sd}, i.e. the change in measured anisotropy per unit change in A_{sd}. For A_{sd} values where the DAI has a gradient greater than one, the DAI has a higher sensitivity than A_{sd} and conversely for gradients less than one. To illustrate this further, the relationships between A_j and A_{sd} were numerically differentiated to produce plots of dA_j / dA_{sd} vs. A_{sd} which directly illustrate this measure of the sensitivity. The intersections between these curves enable the range of anisotropy values where each index shows maximum or minimum sensitivity to be found. This concept of the anisotropy sensitivity has been independently developed in a study which has been published simultaneously with this work (185).

Monte Carlo simulations were performed to investigate the effects of noise on the anisotropy measurements. The diffusion tensor experiment was simulated in the

presence of varying degrees of noise, using the methodology described previously in Chapter 7 (129). The DAIs were then calculated for varying levels of simulated noise contamination (115). The different DAIs, A_j, were plotted as functions of the DNR, for the different cases.

iii) The gamma-variate anisotropy

Using the information obtained from the simulations, a DAI was developed that has low sensitivity to noise, but that retains a good contrast range for typical anisotropy values encountered in human brain imaging. This DAI has a sensitivity that takes the form of a gamma-variate function (186) and as such is called the gamma-variate anisotropy A_{gv}

$$\frac{dA_{gv}}{dA_{sd}} = -aA_{sd}^{n}\, exp\left(-bA_{sd}\right). \qquad [8.5]$$

The reasons for selecting an index of this form will become apparent in later sections. Equation 8.5 is the general form for the sensitivity of A_{gv} (dA_{gv} / dA_{sd}), where a, b and n are constants to be determined, such that a and b are normalisation factors and n determines the steepness of the rising part of the curve and can be chosen depending on the required contrast (186). The maximum sensitivity is chosen to occur at a value of A_{sd}^{max} and the normalisation constants must be found to ensure that A_{gv} scales between 0 and 1. Equation 8.6 shows the solution of A_{gv} for $n = 2$, which is the case used throughout this study.

$$A_{gv} = \frac{a\left(b^2 A_{sd}^2\, exp\left(-bA_{sd}\right) + 2bA_{sd}\, exp\left(-bA_{sd}\right) + 2\, exp\left(-bA_{sd}\right) - 2\right)}{b^3}. \qquad [8.6]$$

The detailed derivation for this and other cases is presented below. To provide optimum contrast between grey and white matter, A_{sd}^{max} was chosen to have a value roughly coinciding with the transition between these two diffusion environments, which has been shown to occur at around $A_{sd} = 0.25$ (128,169). The solution for $A_{sd}^{max} = 0.25$ gives $a = -259.57$ and $b = 8$. A_{gv} can then be calculated on a pixel-by-pixel basis using Eq. 8.6 and compared to the other DAIs.

iv) General derivation of the gamma-variate anisotropy

A general expression for the sensitivity of the gamma-variate anisotropy, dA_{gv} / dA_{sd}, was given in Eq. 8.5. The normalisation constants a and b are found by satisfying the condition given in Eq. 8.7 such that A_{gv} is scaled between zero and one. The maximum of the curve is set to occur at A_{sd}^{max} and so this gives the further condition given by Eq. 8.8.

$$\int_0^1 \frac{dA_{gv}}{dA_{sd}} dA_{sd} = 1 \qquad [8.7]$$

$$\frac{d^2 A_{gv}}{dA_{sd}^2} = -a \exp\left(-bA_{sd}^{max}\right) + abA_{sd}^{max} \exp\left(-bA_{sd}^{max}\right) = 0 \qquad [8.8]$$

Values for n of 1 or 2 were found to give reasonable sensitivity ranges for most applications, as using higher values made the index sensitive to an extremely small range of anisotropy values. This is due to the fast rise of the curve, produced by the dominating high order polynomial, and the normalisation condition given in Eq. 8.7. Integrating Eq. 8.5 and using the conditions imposed by Eq. 8.7 enables A_{gv} to be obtained for $n = 1$ and $n = 2$ as in Eqs. 8.9 & 8.10.

$$A_{gv(n=1)} = \frac{a\left(bA_{sd} \exp\left(-bA_{sd}\right) + \exp\left(-bA_{sd}\right) - 1\right)}{b^2} \qquad [8.9]$$

$$A_{gv(n=2)} = \frac{a\left(b^2 A_{sd}^2 \exp\left(-bA_{sd}\right) + 2bA_{sd} \exp\left(-bA_{sd}\right) + 2\exp\left(-bA_{sd}\right) - 2\right)}{b^3} \qquad [8.10]$$

A value for A_{sd}^{max} is chosen and the constants a and b can be derived using Eqs. 8.5, 8.7 & 8.8 as given in Eqs. 8.11 - 8.13

$$b = \frac{n}{A_{sd}^{max}} \qquad [8.11]$$

$$a_{(n=1)} = \frac{1}{A_{sd}^{max}\left(\exp(-b) + A_{sd}^{max}\left(\exp(-b) - 1\right)\right)} \qquad [8.12]$$

$$a_{(n=2)} = \frac{4}{A_{sd}^{max}\left(2\exp(-b) + 2A_{sd}^{max} \exp(-b) + A_{sd}^{max\,2}\left(\exp(-b) - 1\right)\right)} \qquad [8.13]$$

Table 8.1 shows the constants a and b, calculated for different A_{sd}^{max} values. The procedure adopted in this section can equally be applied to different values of n,

$A_{sd}(max)$	$n=1$ a	$n=1$ b	$n=2$ a	$n=2$ b
0.1	-100.05	10.00	-4000.00	20.00
0.2	-26.05	5.00	-501.39	10.00
0.3	-13.14	3.33	-154.01	6.67
0.4	-8.77	2.50	-71.40	5.00
0.5	-6.73	2.00	-42.00	4.00
0.6	-5.60	1.67	-28.61	3.33
0.7	-4.88	1.43	-21.43	2.86
0.8	-4.40	1.25	-17.13	2.50
0.9	-4.05	1.11	-14.32	2.22

Table 8.1 The gamma-variate anisotropy normalisation constants, a and b, calculated for different values of $A_{sd}(max)$ for the cases where $n = 1$ and $n = 2$.

altering the steepness of the rising part of the curve. The more general method presented could also be used for defining other scaled anisotropy indices using the concept of the anisotropy sensitivity, dA_j / dA_{sd}.

v) MR Imaging

To illustrate the differences in contrast and sensitivity, maps of the DAIs were obtained from a typical transverse section of the human brain. The DTI acquisition was implemented on the Elscint Prestige 1.9T scanner, in which diffusion gradients were applied along 6 non-collinear directions (83). The sequences were implemented to follow the optimisation scheme presented in Chapter 7, such that two different scalar b-values of approximately 100 and 1000 smm^{-2} were used. Imaging parameters were: TE = 120 ms, TR = 1s image^{-1}, Δ = 43.9 ms, δ = 40 ms, 128 x 72 matrix (zero filled to 256 x 128), FOV = 460 x 230 mm, slice thickness = 6 mm, with 2 acquisitions for the b = 100 smm^{-2} and 4 acquisitions for the b = 1000 smm^{-2} data sets. All images were transferred to a Sun Ultrasparc workstation and aligned using SPM95 (Hammersmith Hospital, London, UK), to correct for patient motion between acquisitions, as discussed in §4.4. The data was processed using `map_ten7fsr` and saved in the Matlab readable .mdf format. Custom software was developed in Matlab to calculate and display the anisotropy maps using Eqs. 8.1 - 8.4 & 8.6. A

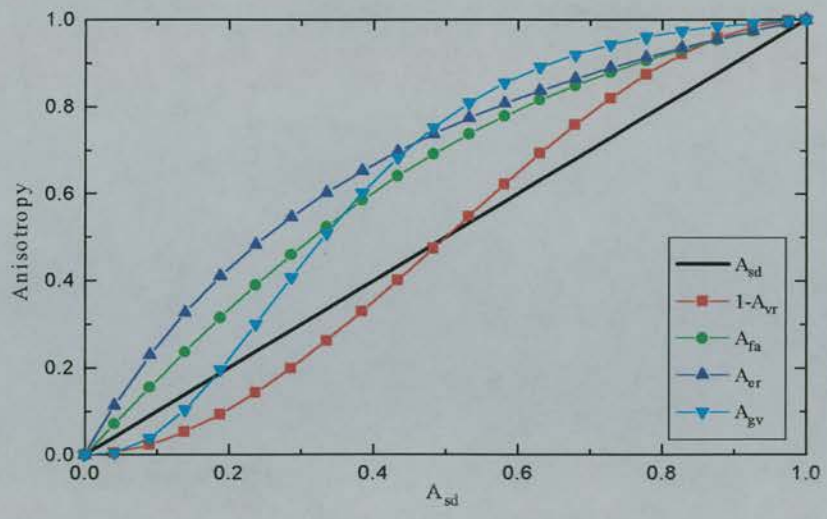


Figure 8.3 The standard deviation (A_{sd}), volume ratio (A_{vr}), fractional anisotropy (A_{fa}), eigenvalue ratio (A_{er}) and gamma-variate (A_{gv}) anisotropy indices, as a function of A_{sd}, plotted from synthetic diffusion data representing axisymmetric diffusion environments.

histogram of A_{sd} values found in the human brain was also plotted to evaluate the sensitivity and likely contrast produced by each of the DAIs, given a typical distribution of values found in human brain imaging.

8.3.3 Results

Fig. 8.3 shows the variation of the different DAIs, A_j, with A_{sd} for axisymmetric diffusion. The non-axisymmetric case complicates the interpretation by producing a 'spread' of the lines shown in Fig. 8.3 (see Fig. 6.2) and so is omitted for clarity (187). This plot demonstrates how all of the DAIs can be expressed as functions of A_{sd}, in some cases with simple analytical solutions as for A_{vr} and A_{sd} in Eq. 6.21. This feature has been exploited to develop A_{gv} in Eq. 8.6. Figure 8.4 shows the variation of the anisotropy sensitivity, dA_j / dA_{sd}, with A_{sd}, again for axisymmetric diffusion. It can be seen clearly how A_{er}, and to a lesser extent A_{fa}, show the greatest sensitivity to isotropic diffusion, whereas A_{gv} and A_{vr} show a very low sensitivity to isotropic diffusion. The maximum sensitivity of A_{gv} and A_{vr} occur at values for A_{sd} of 0.25 and 0.5, respectively. A high sensitivity to anisotropic diffusion is retained by A_{sd}, whereas all of the other DAIs show little sensitivity to highly anisotropic diffusion. Figure 7.7 demonstrated that noise has the greatest influences on

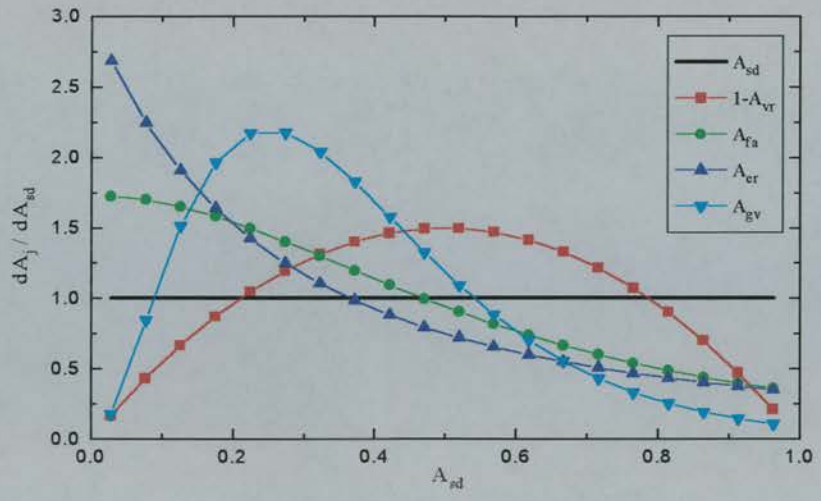


Figure 8.4 The anisotropy sensitivity dA_j / dA_{sd} (j = *sd*, *vr*, *fa*, *er*, *gv*), as a function of A_{sd}, plotted from synthetic diffusion data representing axisymmetric diffusion environments.

measurements of isotropic diffusion, suggesting that a desirable property for a DAI would be to have a relatively low sensitivity to this region, thereby reducing noise contamination. Figure 8.5 shows the dependency of the different DAIs on noise, for the 'worst case' of isotropic diffusion, clearly illustrating how A_{vr} and A_{gv} have a lower sensitivity to noise than A_{sd}, A_{fa} or A_{er}, at least for reasonably high DNR (> 10).

The histogram of A_{sd} values obtained from a typical transverse section of the human brain is shown in Fig. 8.6. The histogram has a maximum value at around A_{sd}

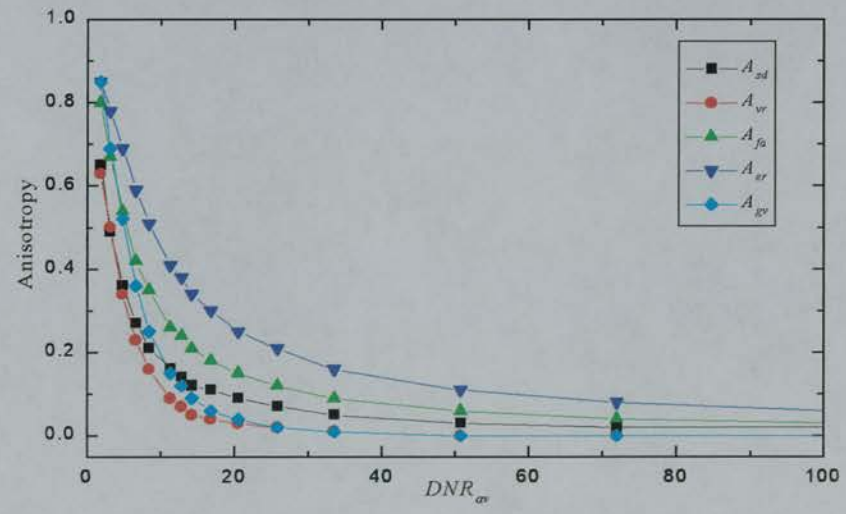


Figure 8.5 The anisotropy indices, A_{sd}, A_{vr}, A_{fa}, A_{er} and A_{gv}, plotted as a function of DNR_{av} for isotropic diffusion. The data was simulated using Monte Carlo methods and clearly illustrates the dependence of the different anisotropy indices on noise.

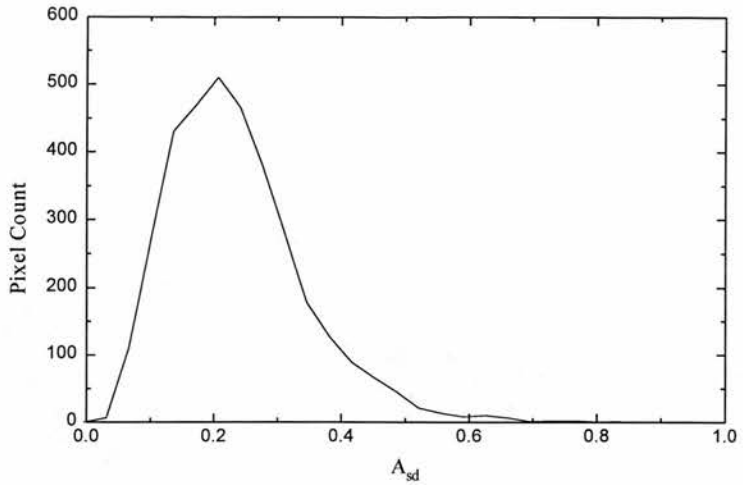


Figure 8.6 A histogram of A_{sd} values measured from a typical transverse section of the human brain, corresponding to that shown in Fig. 8.7.

= 0.25 and has a very low pixel count for values above 0.6 Anisotropy maps corresponding to the data in this histogram are shown in Fig. 8.7, along with a map of $\overline{\lambda}$. All of the anisotropy maps have identical grey scale windowing, such that a pixel with a value of zero is represented by black and a pixel with a value of one is represented by white.

8.3.4 Discussion

Figures 8.3 & 8.4 demonstrate that the regions where the DAIs have maximum sensitivity determine the range of anisotropy values that provide most of the image contrast. A_{er}, and to a lesser extent A_{fa}, show the greatest sensitivity to isotropic diffusion, as is illustrated by the overall brightness of these images in Figs. 8.7d & 8.7e. As such, they are extremely sensitive to the effects of noise contamination, resulting in CSF appearing relatively bright, even though diffusion in CSF is isotropic. This effect is exaggerated because CSF has a high diffusion coefficient and so has a particularly low SNR in the diffusion weighted image (23).

A_{vr} shows CSF to be very dark, because it has a low sensitivity to isotropic diffusion and hence noise. However, the contrast is relatively poor because, as is seen in Fig. 8.6, A_{sd} largely takes values of less than 0.5 and so A_{vr} retains a relatively high sensitivity to a range of anisotropies that have a low pixel count, in the human

brain. Similarly, A_{sd} retains a relatively high sensitivity to anisotropic diffusion and so produces an image that is also fairly dim, again due to the low pixel count at high anisotropy values.

The map of A_{gv} in Fig. 8.7f shows improved contrast in the human brain. This is because it has a low sensitivity to noise and so the CSF in the ventricles appears reasonably dark, while it has a high sensitivity to the majority of anisotropy values found in the human brain, therefore utilising the full contrast range. As expected, the image shown in Fig. 6f also displays improved grey-white matter contrast, due to the maximum sensitivity of A_{gv} occurring with an anisotropy value that is typical of the transition between grey and white matter structures (128,169). The anisotropy values observed in this study seem to be on a somewhat compressed scale when compared to that seen previously (128,169) i.e. isotropic diffusion is more anisotropic and anisotropic diffusion is more isotropic. This may be due to the relatively poor image quality of single-shot EPI techniques and the limited gradient capabilities of our scanner (20). The resulting degradation in image quality is likely

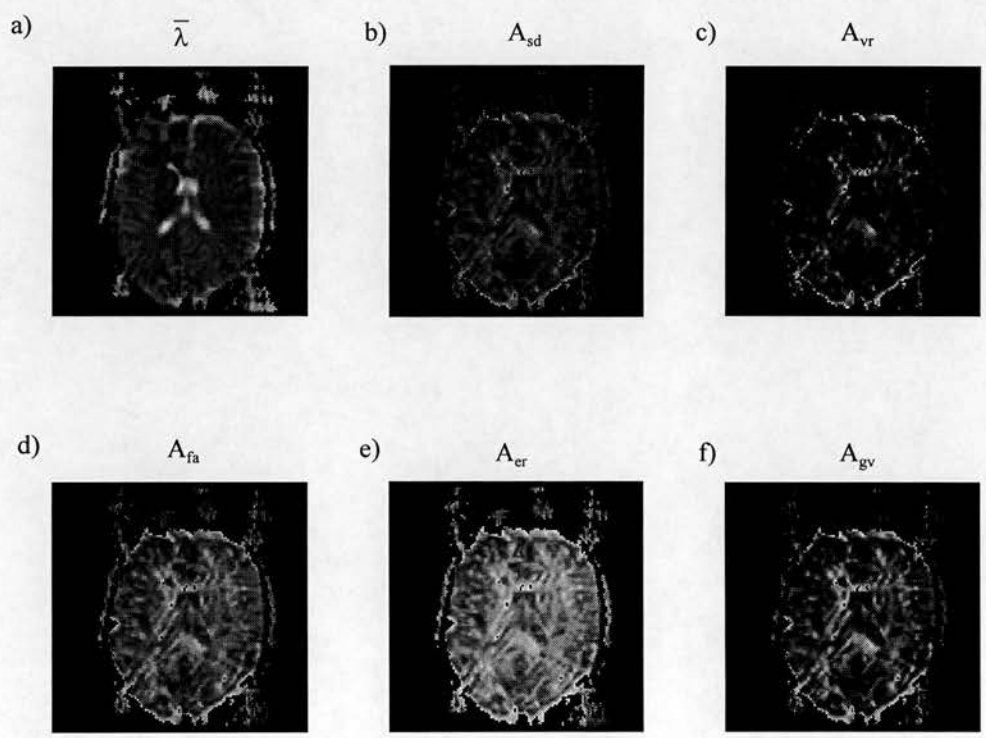


Figure 8.7 Maps of the a) $tr(\boldsymbol{D})$ / 3, b) A_{sd}, c) A_{vr}, d) A_{fa}, e) A_{er} and f) A_{gv}, obtained from a typical transverse DTI data set of a normal volunteer.

to lead to just such a compressed range of anisotropy values and is discussed in more detail in Chapter 9. While this may effect the distribution of the histogram, giving a somewhat lesser spread, the exact form of A_{gv} can be easily altered to compensate for this, as shown in §8.3.2. However, the grey-white matter transition appears to be consistent between different acquisition schemes (128,185,169), so this adjustment may not be necessary.

The form of the gamma-variate function used as the basis for the sensitivity of A_{gv}, can be easily modified for other applications. This type of function will always give a relatively low sensitivity to noise, but the point where the maximum value occurs can be shifted to make the index sensitive to a different range of anisotropy values e.g. for different pathologies. Furthermore, the steepness of the rise and fall of the curve can be altered, to make the range of values to which the index is sensitive considerably greater or smaller, as discussed in §8.3.2.

There may be special cases where an index that is sensitive to isotropic diffusion is desirable. If, for example, it was required to test a protocol for its susceptibility to noise, or for any small gradient mismatches, then the use of an isotropic phantom and an index that is highly sensitive to isotropic diffusion would be appropriate, and so A_{er} or A_{fa} would be suitable. If it was required to delineate highly anisotropic diffusion from isotropic diffusion, as in mapping some highly ordered white matter tracts (128), then an index would be required that showed little sensitivity to isotropic diffusion and a high sensitivity to anisotropic diffusion. Of the DAIs studied, none is ideal, although A_{sd} or A_{vr} may provide enough contrast as they retain some sensitivity to anisotropic diffusion. However, A_{gv} could be modified such that its maximum sensitivity occurred at a high value of anisotropy with an appropriate rise and fall defined, such that little sensitivity was given to isotropic diffusion.

It is proposed that A_{gv}, along with the concept of the anisotropy sensitivity, would be used as a method for displaying anisotropy maps with improved contrast for specific applications. It is not proposed as a new quantitative measure for comparative studies, as analysis of anisotropy data obtained between different groups, using different indices, is difficult enough. It is suggested that the standard deviation index A_{sd}, be used as the definitive quantitative measure of diffusion

anisotropy as this agrees with a previously defined, physically meaningful, chemical definition of anisotropy (182). However, the other DAIs may be useful for delineating specific cases, and for improving the visual appearance of anisotropy maps.

8.4 A full characterisation of diffusion using 2D maps

Traditional radiological images have usually been of a two-dimensional grey scale format. Therefore, most of the images derived from the tensor have attempted to provide images that conform with this tradition. These images should be physically meaningful and provide image contrast useful for highlighting pathologies. The maps of D_{av} are well established and provide images that indicate the magnitude of diffusion with black - white representing low - high diffusivities. Maps of anisotropy are less well established, with many different indices being proposed, making comparisons between research groups difficult. These indices exploit the variation of the eigenvalues in different ways, to provide subtly different image contrast and different sensitivity to noise, as discussed in §8.3. However, as described in §8.2.3, these indices alone cannot characterise unambiguously the physical shape of the diffusion ellipsoid.

The prolateness index (P) was introduced in §6.2.5, to provide a measure of the symmetry of the diffusion environment, which provides complementary

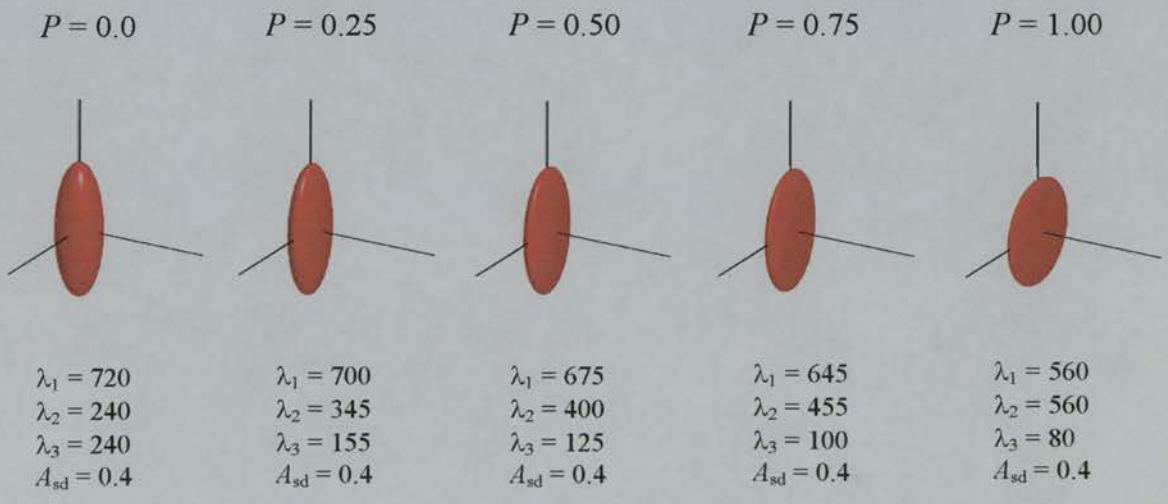


Figure 8.8 Illustration of the effects of ellipsoid shape and symmetry on the Prolateness index (P). All ellipsoids have constant $A_{sd} = 0.4$ and $tr(\boldsymbol{D}) / 3 = 400$ a.u.

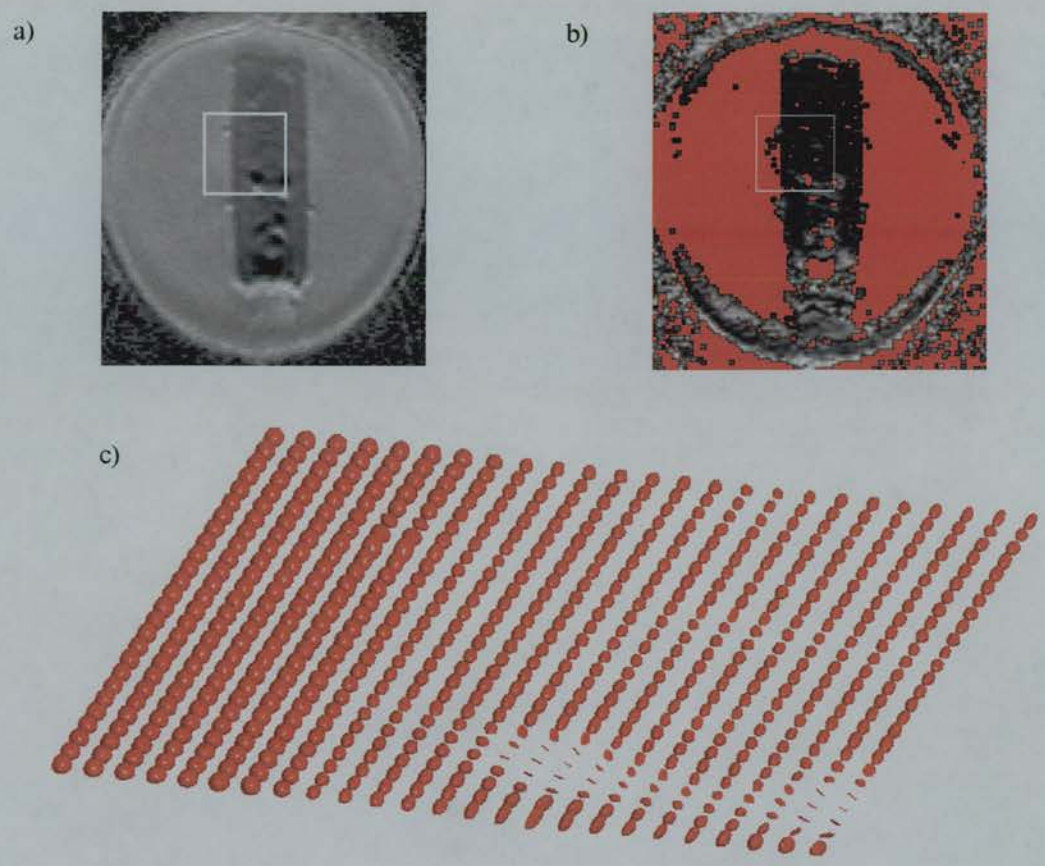


Figure 8.9 Results obtained from a phantom consisting of an anisotropic vegetable (leek) suspended in a reservoir of water, showing a) ADC_{av}, b) Prolateness (P) and c) diffusion ellipsoid image corresponding to the ROI illustrated in a) and b).

information to A_{sd} and enables a complete characterisation of the diffusion ellipsoid. The idea behind the index was to provide a physically interpretable representation of the diffusion ellipsoid symmetry. Therefore, P was designed such that a value of zero represents axisymmetric prolate environments and a value of one represents axisymmetric oblate ellipsoids as illustrated in Fig 8.8. Values in between represent a loss of symmetry, with a value of 0.5 representing the transition between prolate and oblate diffusion. For an ellipsoid with $P = 0.5$, the maximum degree of asymmetry is displayed, which is indicated schematically in Fig. 8.8.

The effectiveness of P at providing interpretable images of structural symmetry was investigated using in vitro and in vivo data. For nearly isotropic

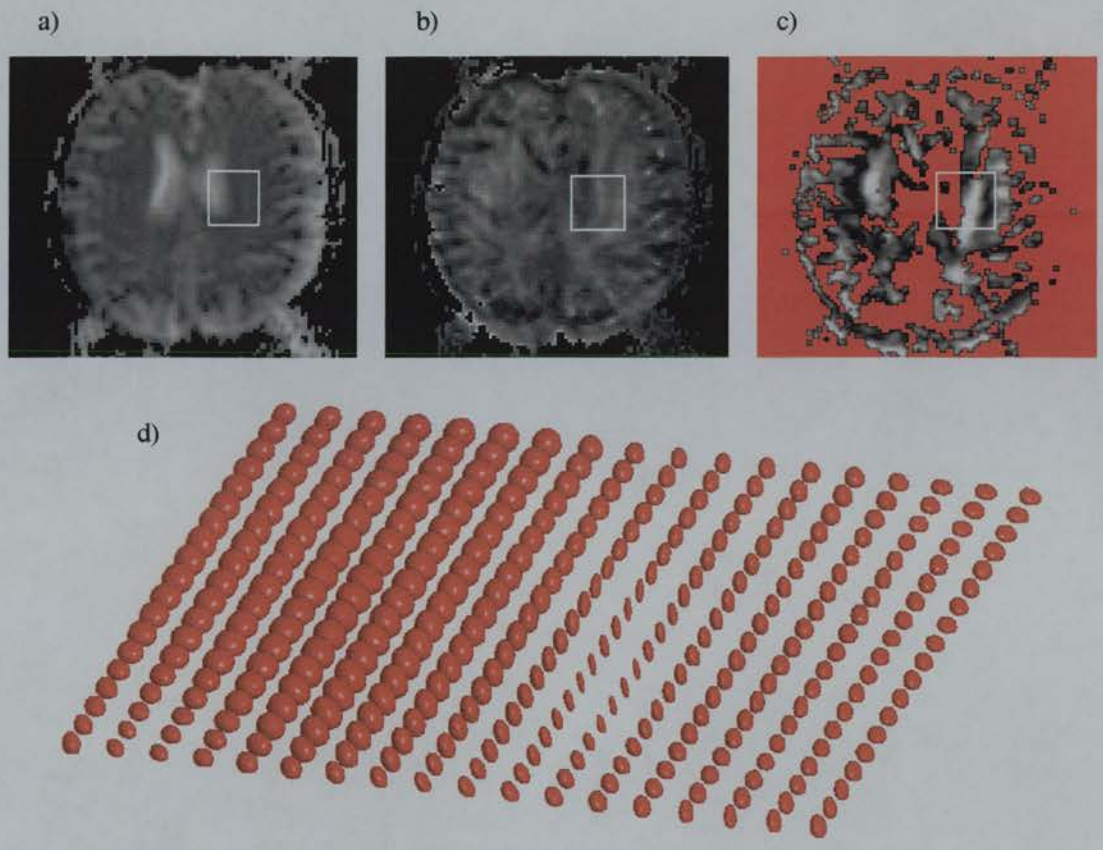


Figure 8.10 In vivo results showing a) ADC_{av}, b) A_{sd}, c) P and d) the diffusion ellipsoid image corresponding to the ROI illustrated in a) - c).

diffusion, P provides a measurement of the symmetry that is susceptible to errors. This arises due to the definition of P in Eq. 6.22 and the close proximity of the prolate and oblate curves shown in Fig. 6.2. However, P has little effect on the appearance of the ellipsoid, unless the diffusion is significantly different from the isotropic or anisotropic extremities. For nearly isotropic diffusion, all of the eigenvalues must be similar and so deviations between the two equatorial diffusivities (asymmetry) will have little effect on the overall appearance of the ellipsoid. Similarly, for highly anisotropic diffusion, one of the eigenvalues must be considerably different from the other two and so again the deviation between the two equatorial elements will be insignificant. For visualisation purposes, it was found that filtering out these highly isotropic or anisotropic pixels provided a clearer indication as to the symmetry of the ellipsoids.

To illustrate these effects a phantom consisting of an anisotropic vegetable suspended in a reservoir of water was used to demonstrate the effectiveness of P at

characterising ellipsoid symmetry, as illustrated in Fig. 8.9. The image is masked such that P is calculated only when $0.25 < A_{sd} < 0.75$, otherwise pixels are displayed red. It can be clearly seen from Fig. 8.9 how the water is red, indicating isotropic diffusion, while the vegetable demonstrates prolate diffusion, indicated by the low pixel intensity. In the root of the vegetable (bottom of image), some of the pixels appear bright, indicating a tendency towards oblate diffusion. The in vivo case is illustrated in Fig. 8.10. The masking procedure applied to Fig. 8.10c has segmented the image such that only white matter regions are shown as grey scale. The map of P appears to highlight the different forms of ellipsoid shown in Fig. 8.10d reasonably effectively. Isotropic diffusion in the ventricles appears red on the map of P due to the spherical symmetry of the diffusion ellipsoids in this region. The band of anisotropic oblate ellipsoids are highlighted as white on the P map, while the prolate ellipsoids near to the edge of the ROI and below the ventricles show up black, as expected. It is interesting to observe that a reasonable proportion of voxels contain oblate diffusion ellipsoids, possibly suggesting a diffusion environment consisting of stacked planes (128). The cellular environment is characterised by the macroscopic diffusion tensor on a voxel-averaged basis (24) and so is not likely to consist of fibrous axisymmetric structures alone (85). Rather, the diffusion tensor and its derived parameters are likely to describe a complex average of the different structures present in the voxel.

8.5 Providing an indication of fibre orientation

The images shown in Fig. 8.9c and 8.10d illustrate the size, shape and orientation of the diffusion ellipsoids as measured from a DTI experiment. Therefore, these images provide a complete illustration of the diffusion environment. The ellipsoid orientation indicates the orientation of the restrictions as determined by the principal diffusion coordinate system. However, the usefulness of ellipsoid maps for providing an indication of ellipsoid orientation is seriously limited by the size of the ROI. On typical paper printouts, a large ROI results in ellipsoids that are very small and it can be extremely difficult to observe the orientation of the ellipsoid. The

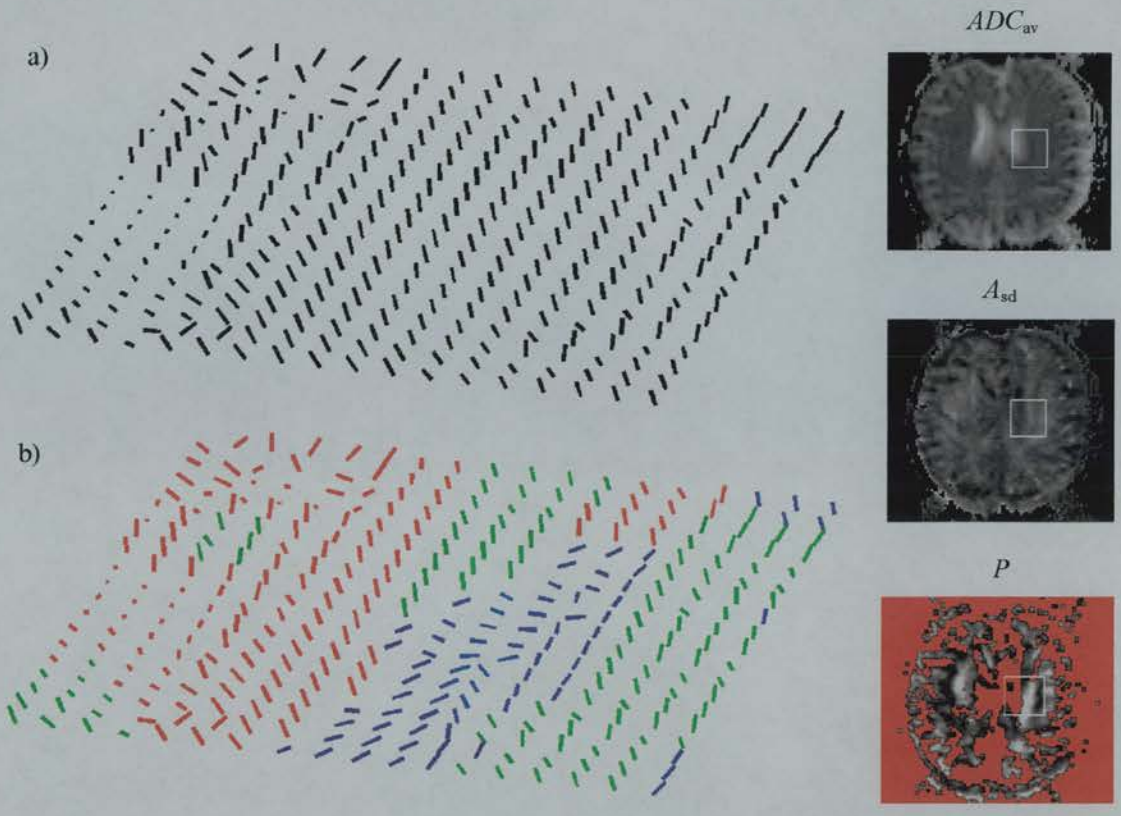


Figure 8.11 Illustration of fibre-tract orientation maps obtained in vivo for the ROI corresponding to Fig. 8.10, demonstrating the random nature of the fibre orientations within the ventricles. This arises because there is no preferred principal diffusion direction in an isotropic medium such as CSF. The map in a) is based solely on the orientation of the principal eigenvalue ε_1, while b) is based on ε_1 for prolate diffusion and ε_3 for oblate diffusion. The lower map is colour coded with red representing isotropic diffusion ($A_{sd} < 0.25$), green prolate diffusion and blue oblate diffusion.

development of maps that clearly indicate the orientation of the diffusion ellipsoid for large ROIs is therefore highly desirable. Several colour maps have been suggested in the literature for mapping fibre orientation, but the colour contrast is often extremely difficult to interpret (177-181).

A graphical representation of fibre-tract orientation can be constructed by plotting a line along the principal diffusion direction (corresponding to λ_1 and ε_1) for each voxel in the image. These maps only show the principal orientation of a fibre and not the magnitude and shape, a feature which simplifies the interpretation of the image, when compared to diffusion ellipsoid maps. A disadvantage of fibre orientation maps based solely on the principal eigenvalue λ_1 is that they are only valid for prolate ellipsoids. For oblate ellipsoids the symmetry axis is determined by

the smallest eigenvalue λ_3 and so it is appropriate to plot the corresponding direction given by ε_3. An example of a fibre-tract orientation map is given in Fig. 8.11, corresponding to the slice in Fig. 8.10, with a) a map based solely on ε_1 and b) a map based on ε_1 and ε_3 with appropriate colour coding.

Generally, the orientation of the principal eigenvalue is only of interest for anisotropic diffusion as there is no preferred principal diffusion direction for isotropic diffusion, resulting in a largely random distribution of fibre directions. It is therefore usual to apply a mask to these images and only plot the fibre orientation for voxels demonstrating significant anisotropy (e.g. $A_{sd} > 0.25$). This procedure has been taken further, to observe the connectivity of the major white matter fibres. In these images, the likelihood of fibres being connected is predicted from the principal eigenvectors in adjacent voxels and maps are produced only for fibres that show connectivity between adjacent voxels (180,181). These maps may be extremely useful for characterising white matter diseases such as schizophrenia that are believed to result from a loss of order in white matter fibres.

8.6 Conclusions

This chapter has been concerned with techniques for improving the visualisation of quantitative diffusion data. Many different scalar indices have been proposed for illustrating diffusion data, but only ADC_{av} is generally accepted as a standard measure, as used by the majority of research groups. A comparison of the sensitivity of the different anisotropy indices proposed in the literature was undertaken. This lead to a methodology being developed for producing anisotropy indices with improved contrast and a low sensitivity to noise. This was demonstrated in vivo with the production of the gamma-variate anisotropy (A_{gv}). This index can be modified to exploit the image contrast and provide a clearer visualisation of structural differences. Conventional anisotropy indices were shown to provide ambiguous measures of the ellipsoid shape and so a further 'prolateness' index was developed to describe the degree of ellipsoid symmetry and overcome these ambiguities. This index was illustrated for in vivo and in vitro data and was shown to provide a physically interpretable measure of symmetry that enables a complete

characterisation of the diffusion ellipsoid shape. Several graphical representations of the diffusion environment have been illustrated and their shortcomings discussed, notably diffusion ellipsoid and fibre-tract orientation maps. It was shown that while the ellipsoid maps provide a complete description of the diffusion environment in a single image, they contain too much information to be visually helpful for many situations.

9

In Vivo Diffusion Imaging Studies

9.1 Introduction

Chapters 6-8 have been concerned with developing a methodology for acquiring and displaying reliable diffusion data, based on the shortcomings of the preliminary experiments made in Chapter 5. However, as the aim of the project was to implement quantitative diffusion imaging into a clinical environment, a large proportion of the project has been involved with acquiring and processing images obtained from the various clinical studies that utilise DWI or DTI techniques within our establishment. In total 175 clinical diffusion imaging studies were performed as part of this project; 63 using the navigated spin-echo sequence on the Siemens scanner, 89 using the orthogonal EPI protocol, and 23 using the DTI protocol, on the Elscint scanner. The majority of these were acute stroke patients (88%), derived from the Lothian Stroke Register (LSR). The LSR is co-ordinated at the Western General Hospital (WGH) in parallel with the MRC funded clinical research initiative (CRI) into stroke and head injury, through which this work was funded. In addition, a series of 16 acute head injured patients were recruited from the intensive care unit of the WGH. The complications arising due to the necessary intensive care and physiological monitoring of these patients, means that studies of this group of patient have been limited, in the literature.

9.2 Acute stroke DWI study

9.2.1 Introduction

In Chapter 5, results were presented from a series of acute stroke patients that underwent DWI using the orthogonal imaging protocol on the Siemens scanner. This section presents results from a similar study obtained using the Elscint EPI orthogonal diffusion protocol discussed in §4.3.2. The reliability and reproducibility of the Elscint and Siemens data sets is discussed, along with an evaluation of the quantitative data obtained. The study consisted of 59 patients, 42 of which were scanned once, 9 were scanned twice and 8 were scanned three times. The first scan was acquired during the period 1 - 4 days post-ictus, the second was obtained 4 - 7 days post-ictus and the third after 10 - 14 days, enabling the evolution of the diffusion abnormality to be observed over a period of time. Also, the abnormal ROIs were separated into those occurring in grey and white matter and so the evolution of the diffusion parameters could be compared between lesions evolving in grey and white matter. The limitations of the results obtained in this study are discussed in terms of the shortcomings of the orthogonal sampling technique described in previous chapters.

All imaging was performed on the Elscint 1.9T Prestige scanner, using the EPI diffusion sequence with diffusion gradients applied along orthogonal x, y and z directions, along with a baseline (b = 0) image with no diffusion gradients applied. Imaging parameters were: TE = 120 ms, TR = 10s image^{-1}, Δ = 43.9 ms, δ = 40 ms, 128 x 72 matrix (zero filled to 256 x 128), FOV = 460 x 230 mm and slice thickness = 6 mm. The diffusion-weighted images were acquired by applying diffusion gradients of magnitude 14 mTm^{-1}, leading to a scalar b-value of approximately 686 smm^{-2}. Ten transverse slices were acquired giving 40 images in a total acquisition time of 40s. All images were transferred to a Sun Ultrasparc workstation and aligned using SPM95, to correct for patient motion occurring between acquisitions, as discussed in §4.5. The data was processed using the C programs, `scale_ort` and `map_ort`, and saved in the ANALYZE readable .img and .hdr format as described in §4.5.2. Areas of diffusion abnormality were then identified by a neuroradiologist and ROIs defined on the quantitative diffusion maps using ANALYZE.

9.2.2 Diffusion measurements in acute stroke

Of the 59 patients presenting with symptoms of acute ischaemic stroke, 32 were demonstrated to have a visible, measurable diffusion abnormality. As described in Chapter 5, ROI measurements of ADC_{av} and RV A_{sd} were obtained from lesion and contralateral normal appearing regions of the brain. In total, 148 ROIs were obtained from the 32 patients, ranging in size from 4 to 865 pixels (mean 138 ± 134). The difference in ROI size between this data set and the one studied in Chapter 5, where the mean ROI size ranged from 3 to 62 pixels (mean 18 ± 17) can be clearly seen. This reflects the different selection criteria for patients recruited into the two studies. For the study in Chapter 5, patients were recruited who had small cortical or lacunar strokes (patients with larger strokes underwent spectroscopic imaging), while in this study patients were recruited regardless of lesion size.

The data are summarised using statistical box-plots in Fig. 9.1 and clearly demonstrates the reduction of ADC_{av} and the increase of A_{sd} within the lesions. For comparison, the data obtained in Chapter 5 using the Siemens scanner is also displayed. The mean ADC_{av} was $0.82 \pm 0.19 \times 10^{-3}$ mm^2s^{-1} within the lesion and $1.33 \pm 0.26 \times 10^{-3}$ mm^2s^{-1} within contralateral normal appearing regions and is in good agreement with previous acute stroke studies (e.g. 21,111,135,167,168,169). The qualitative RV measure of A_{sd} gave values of 0.21 ± 0.08 within the lesion and 0.16 ± 0.06 for normal appearing regions. The data were demonstrated to be normally distributed, using the Anderson-Darling normality test and subsequent one-sample t-tests of the mean stroke-to-normal ratios (s/n) revealed that both the reduction in ADC_{av} ($p < 0.0001$) and the increase in A_{sd} ($p < 0.0001$) were significant.

However, the data obtained in this study were acquired using non-optimised experimental parameters, whereby a single acquisition was performed for each image at b-values of 0 and 686 smm^{-2}. Chapter 7 demonstrated that the optimal b-value for imaging in the human brain was around $\Delta b \approx 900$ smm^{-2}, and that signal averaging is likely to be needed to obtain noise-independent measurements. The optimisation scheme was presented for a DTI protocol, but is equally applicable to orthogonal imaging (where $DNR_{eff} = DNR_{min} / \sqrt{3}$). Following the scheme proposed in Chapter

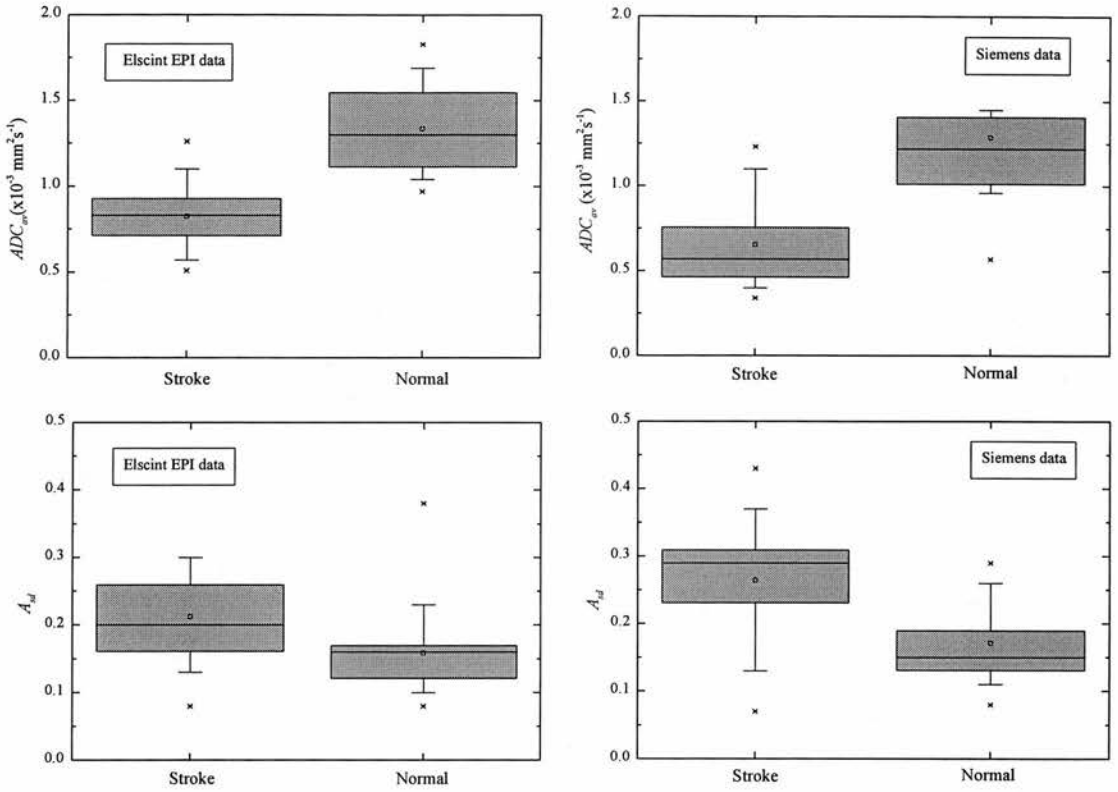


Figure 9.1 Statistical box-plots representing the ADC_{av} reduction and A_{sd} increase observed in acute stroke. The data is presented for the study performed using the Elscint EPI sequences and also, for comparison, the Siemens data set discussed in Chapter 5.

7, with a typical SNR_1 of 40 (as was typical in this study), orthogonal imaging would require $n_1 = 24$ and $n_2 = 38$ to achieve $DNR_{av} > 50$ over all voxels in the image. Even relaxing this condition to $DNR_{min} = 30$ requires $n_1 = 8$ and $n_2 = 13$. Therefore, the measurements of anisotropy obtained in this study are likely to be highly noise dependent. This is a possible explanation for the elevated A_{sd} observed in the stroke regions, as the reduced ADC_{av} in stroke results in a considerably lower DNR_{av} in these regions and hence a potentially artificially elevated A_{sd}. In contrast, Chapter 7 demonstrated that measurements of ADC_{av} were much less sensitive to the effects of noise contamination and so these measurements should be reasonably reliable and indicate a real reduction of ADC_{av} in stroke regions.

9.2.3 Comparison of Siemens and Elscint data

The data sets obtained using the Siemens spin-echo and the Elscint EPI sequences differ in their acquisition times by three orders of magnitude and it may be expected that some degradation in image quality could result from acquiring images extremely rapidly (20,188). It is essential that any image artefact increase is not sufficiently great such that the quantitative data is compromised. Therefore, this section concentrates on the differences between the in vivo quantitative data obtained using the Elscint and Siemens scanners.

The Siemens data set consisted of 12 patients, mean age 66 ± 14 years, with a mean time from stroke onset (TSO) of 4.0 ± 3.8 days, while the Elscint data set consisted of 32 patients, of mean age 67 ± 21, with a mean TSO of 1.7 ± 1.2 days. As can be seen, the patients are extremely well age matched (p = 0.85), but the TSO

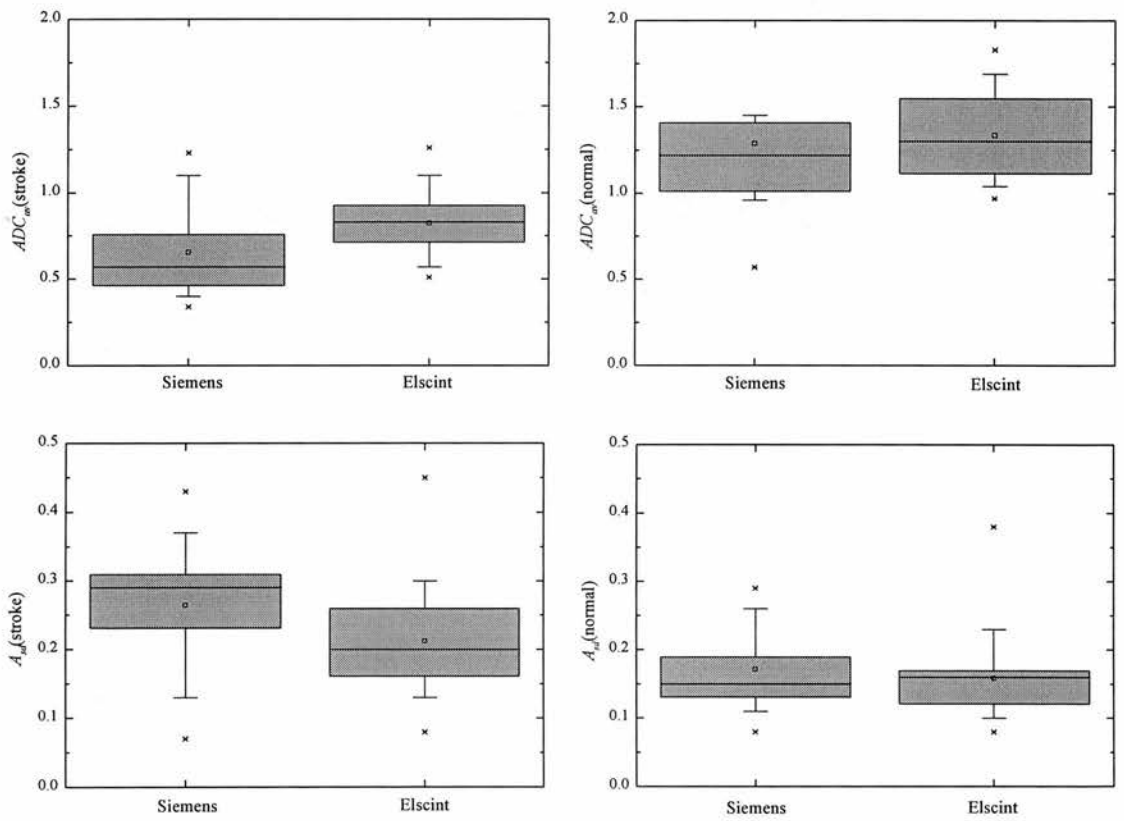


Figure 9.2 The differences between ADC_{av} and A_{sd} obtained from stroke and contralateral normal ROIs, using the Siemens and Elscint scanners. The general trend was for an increased ADC_{av} and reduced A_{sd} for the Elscint EPI sequence, however these differences were not statistically significant.

is less well matched ($p = 0.059$). The increased TSO in the Siemens data set was due to the heavy clinical demands placed on this scanner, which meant that only three research imaging slots were available during the week. In contrast, the Elscint scanner was research dedicated and stroke patients were prioritised, thereby reducing the delay before imaging.

The differences between the data sets are summarised in Fig. 9.2 for ADC_{av} and A_{sd} in both stroke and normal regions. There was no statistically significant difference between the two data sets for any of the measurements, although there was a general trend of reduced ADC_{av} and increased A_{sd} in the Siemens data. The increased ADC_{av} in EPI images seems to be a general trend followed in the literature (167,169). This may be due to the increased susceptibility to artefacts of EPI which generally lead to a loss of signal, particularly in the high *b*-value images. Also, the navigated spin-echo sequence has a typical SNR_1 of 35, whereas the EPI sequence has SNR_1 of around 41. The lower SNR_1 in the Siemens images could result in an increased A_{sd} because the necessary signal averaging was not performed to make A_{sd} noise independent (as described in §9.2.2). Other potential reasons for the discrepancies include the differences in TSO between the studies and the differences in imaging parameters, such as Δ and δ (69). Nevertheless, these small changes are not statistically significant and would suggest that quantitative in vivo data obtained from EPI images can be compared to data obtained using conventional imaging.

9.2.4 Grey and white matter changes

It is well known that diffusion in white matter is highly anisotropic, while diffusion in grey matter is relatively isotropic, but ADC_{av} is believed to be consistent between both tissue types. As discussed in §3.7.1, the process of infarction causes a reduction in ADC_{av} during the early acute stages which subsequently rises until finally becoming elevated above normal after the onset of vasogenic oedema. The anisotropy is thought to reduce during the onset of vasogenic oedema, due to disintegrating cell membranes (126,169). Changes in ADC_{av} occurring during the ischaemic process may be expected to be similar for both grey and white matter because the initial (normal) and final (vasogenic oedema) values are likely to be the

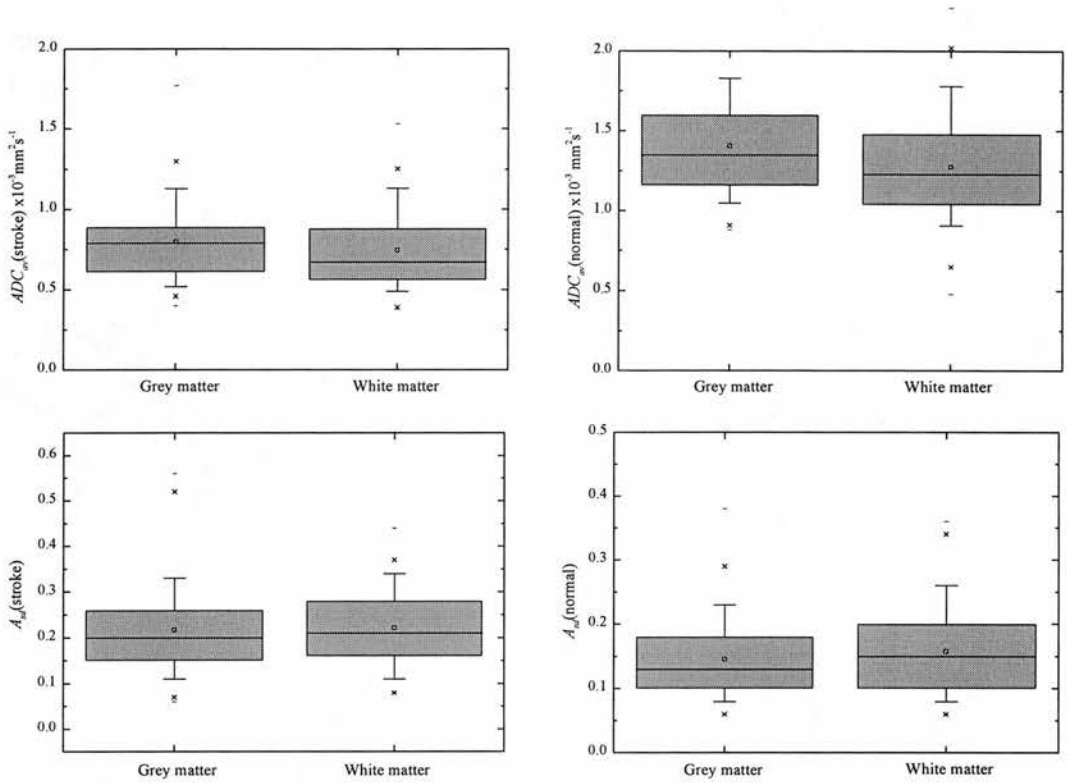


Figure 9.3 Illustration of the changes in ADC_{av} and A_{sd} for lesions evolving in grey and white matter along with the corresponding contralateral normal regions. As expected ADC_{av} demonstrates no significant difference between grey and white matter. The expected increase of A_{sd} in normal white matter is not observed, highlighting the limitations of the orthogonal technique and the degree of noise contamination, discussed in Chapters 6 and 7.

same for both tissue types. However, for changes in anisotropy, the initial normal values are considerably higher in white matter, due to the ordered structure, while in the final state of vasogenic oedema, the infarct would be expected to demonstrate isotropic diffusion irrespective of whether the lesion evolved in grey or white matter because all ordered structure is destroyed. Therefore, the evolution of anisotropy in white matter and grey matter must be different and it would be potentially valuable to identify the point at which white matter structures begin to break-down significantly. Therefore, this section is concerned with observing the differences in the measured ADC_{av} and A_{sd} for infarcts evolving in grey and white matter, while §9.2.5 observes these changes serially in time.

The results for the grey-white matter comparison are summarised in Fig. 9.3, for both ADC_{av} and A_{sd} in lesion and contralateral normal ROIs. As expected, ADC_{av}

was not significantly different between grey and white matter regions in either lesion or normal regions. However, there was a tendency towards slightly higher ADC_{av} in grey matter regions. This is probably due to the close proximity to CSF of many grey matter regions, where the high diffusivity of CSF results in an increase of the measured ADC_{av} due to partial volume effects at tissue-CSF borders. Measurements of A_{sd} demonstrated no significant difference, between grey and white matter regions either within the lesion or within contralateral normal regions. This is in disagreement with the expected result of increased anisotropy in white matter regions and may be indicative of the poor quality of the anisotropy measurements obtained using the orthogonal imaging technique. As discussed in Chapter 6, the RV A_{sd} provides a measurement of anisotropy that is highly dependent on the fibre orientation and is likely to underestimate the true anisotropy greatly (on average by 35%). The RV nature of this measure is therefore likely to reduce the contrast between grey and white matter. Also, as discussed previously, the measurements of anisotropy obtained in this study are not noise independent, which again is likely to mask out any true differences between A_{sd} in grey and white matter. These findings support the observations in Chapters 6 and 7 that it is necessary to obtain rotationally invariant and noise independent measurements before a reliable assessment of anisotropy can be made.

9.2.5 Serial diffusion changes

The evolution of the diffusion parameters during the ischaemic process may provide information on the likely progression of the tissue towards infarction. For example, a rapid fall in ADC_{av} during the first few days, may be indicative of a large increase in the intracellular-extracellular water fraction (87). This in turn could lead to irreversible tissue damage and consequent infarction. It may be that a less rapid decrease in ADC_{av} during the first few days, indicates a lower degree of cell swelling and possibly reversible structural changes, if reperfusion occurs. While these suggestions are speculative, it seems reasonable that the alteration in ADC_{av} and A_{sd} in the early stages, may indicate the likely probability of infarction. The serial changes in ADC_{av} have been well documented in the literature (90,137,189), with the

initial reduction in ADC_{av} being followed by a more gradual decrease up to a few days, followed by a steady rise, until becoming elevated above normal after a period of several weeks. Serial changes in anisotropy have not been studied extensively in the literature (169,126), probably due to the time consuming necessity to perform DTI. The few studies undertaken suggest that there is a possible increase in anisotropy in the early stages followed by a gradual decrease during later acute and chronic stages. The aims of this section are to study the serial changes in ADC_{av} over time, and verify the time course observed by other groups. The time evolution of the anisotropy is not considered due to the unreliability of the measurements obtained in this study, as was demonstrated in §9.2.4.

Fifteen patients with measurable lesions were studied serially using the orthogonal diffusion imaging protocol. The patients were observed during three time bands, 0 - 4 days, 4 - 7 days and 10 - 14 days, following the onset of stroke symptoms. Seven of the patients were scanned in the first two time bands and eight in all three time bands. The ROIs were defined by a neuroradiologist, marking the hyperintense regions on the anisotropy independent DWI_{av} image. The lesion ROIs were separated into those evolving in grey matter and white matter, and contralateral normal regions were also marked. The results are summarised in Fig. 9.4, demonstrating the patient specific changes (a, c, e) and the overall trends (b, d, f) in all tissue types (a, b) and for grey matter (c, d) and white matter (e, f) separately.

Figures 9.4a, 9.4c & 9.4e clearly demonstrate the wide variation in ADC_{av}(s/n) between the different patients, with no obvious trends being visible. However, by grouping the patients according to their time bands and plotting the mean ADC_{av} as a function of the mean TSO the overall trend of ADC_{av} recovery is revealed, as shown in Figs. 9.4b, 9.4d & 9.4f. The statistical significance of the ADC_{av} changes between time groups were obtained using one-sample paired t-tests. The data set obtained from 'all regions' demonstrated no significant change between time groups 1 and 2 ($p = 0.22$), but a significant change between time groups 2 and 3 ($p = 0.001$). This finding agrees with previous studies, whereby ADC_{av} demonstrates a gradual decrease up to around three days after stroke onset and then increases steadily afterwards. Therefore, it is expected that there may be no statistically significant difference between time groups 1 and 2 obtained after a TSO of 1.6 ± 1.1

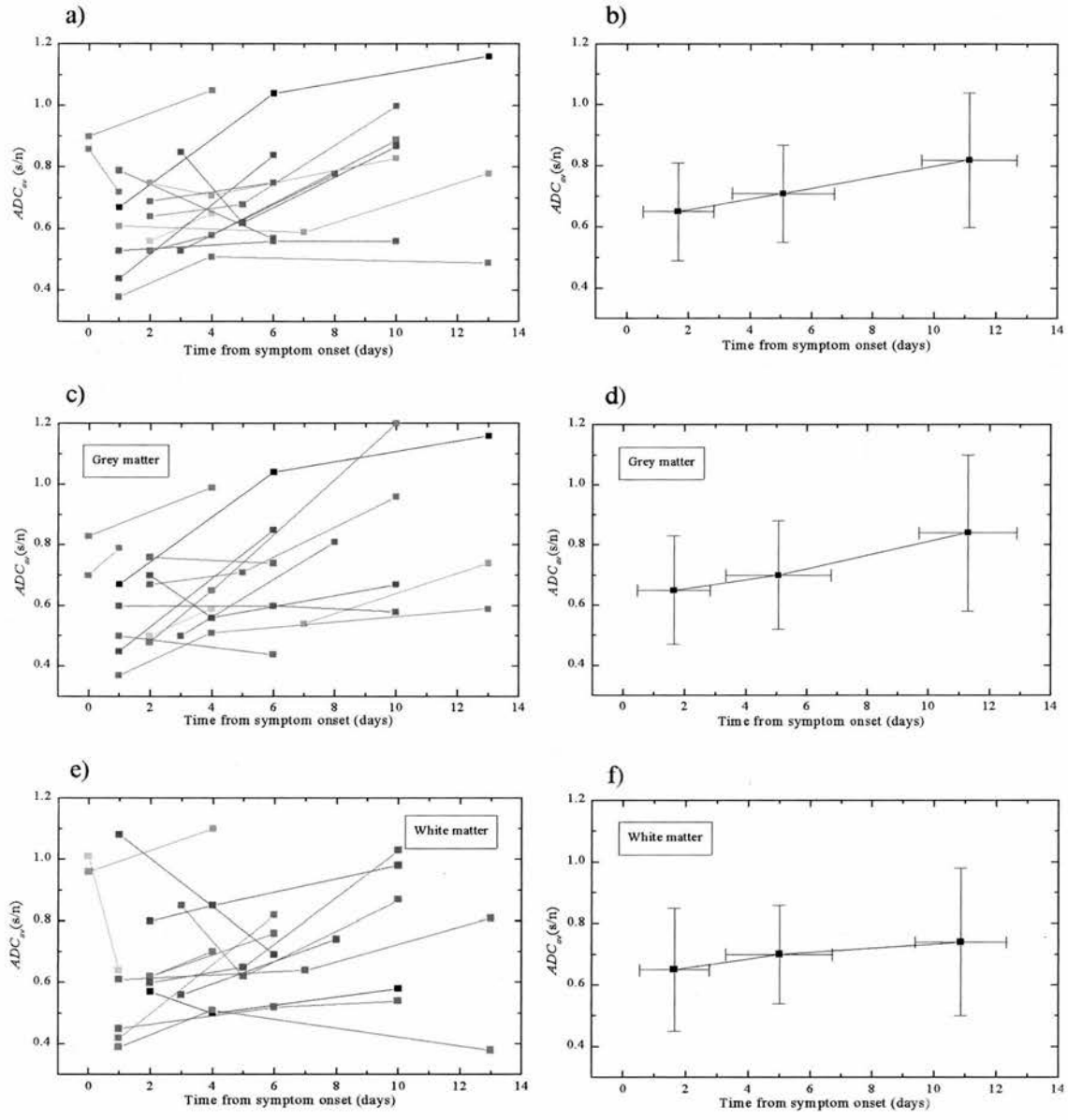


Figure 9.4 The temporal evolution of ADC_{av}(s/n) in the 15 patients studied serially for a) all ROIs, c) grey matter ROIs and e) white matter ROIs. The general trends for each scan are observed for b) all ROIs, d) grey matter ROIs and f) white matter ROIs.

and 5.0 ± 1.7 days, respectively. However, at a later time (10.9 days from TSO), the cell membranes begin to disintegrate and the ADC_{av} increase dominates and becomes statistically significant.

It is interesting to observe the differences between ADC_{av} evolution in grey and white matter regions. Grey matter regions demonstrate a steady increase in ADC_{av}, with a statistically significant increase between both time group 1 and 2 ($p = 0.02$) and time group 2 and 3 ($p = 0.04$). In contrast, the evolution of ADC_{av} in white

matter demonstrates more variability, with a greater proportion of patients having a reduction in ADC_{av} between the first two time points. Despite the average trend being for an ADC_{av} increase over time, these changes are not statistically significant, particularly between time group 1 and 2 ($p = 0.25$), while the difference between time group 2 and 3 approaches significance ($p = 0.08$). This may indicate that the previously observed decrease in ADC_{av} (169) during the first few days is a feature of lesions evolving in white matter, but not grey matter. This may indicate a difference in structural breakdown mechanisms between lesions evolving in white matter and grey matter. Clearly, with the large variability between patients, these observations would need to be verified with a larger group of patients, but the preliminary results presented here, indicate that there may be differences between the diffusion evolution in grey and white matter. The ability to obtain reliable anisotropy measurements, would further support these observations, providing a more robust picture of the cell membrane breakdown. Observing these changes may provide information on the point at which irreversible cell damage has occurred, which may be useful for determining the type of drug therapy administered.

9.3 Acute stroke DTI study

9.3.1 Introduction

Results were presented in §9.2 for measurements obtained from a series of stroke patients using the orthogonal imaging technique. This imaging strategy produces a fast acquisition of ADC_{av}, but as has been discussed in previous chapters, does not provide a quantitative analysis of anisotropy. Furthermore, the acquisition strategy was not optimised to produce noise independent measurements in this study. Therefore, in this section a DTI acquisition protocol is discussed which was designed to overcome the limitations of the orthogonal imaging technique. Due to time constraints, only 15 patients underwent DTI, with 9 of these having visible lesions, compared to 59 for the orthogonal DWI study.

Each patient was scanned using a DTI protocol based on the optimisation scheme developed and verified in Chapter 7. In this study, the typical baseline SNR_1 value was around 35 and Table 7.2 suggests that an appropriate scheme would have

b_1 = 100 smm^{-2}, b_2 = 1000 smm^{-2}, n_1 = 2 and n_2 = 4, as was verified in §7.6.2. The standard Elscint EPI diffusion sequences were used (§4.3.2), with gradients of magnitude 3.78 and 11.95 mTm^{-1} applied along vectors $(1/\sqrt{2},0,1/\sqrt{2})$, $(-1/\sqrt{2},0,1/\sqrt{2})$, $(0,1/\sqrt{2},1/\sqrt{2})$, $(0,1/\sqrt{2},-1/\sqrt{2})$, $(1/\sqrt{2},1/\sqrt{2},0)$ and $(-1/\sqrt{2},1/\sqrt{2},0)$. This sampling strategy leads to the acquisition of 36 images per slice, 12 at b_1 = 100 smm^{-2} and 24 at b_2 = 1000 smm^{-2}. For all patients, 10 slices were obtained giving 360 images in a total imaging time of 6 minutes. The data was then transferred off-line to a Sun workstation and processed using `scale_ten7` and `map_ten7fsr` into ANALYZE readable .img and .hdr form, as described in §4.5.3. As before, areas of diffusion abnormality were then identified by a neuroradiologist and ROIs defined on the quantitative diffusion maps using ANALYZE.

9.3.2 Quantitative anisotropy measurements in normal subjects

In §7.6.2, the optimised DTI protocol was shown to produce noise independent anisotropy measurements. This section compares the anisotropy measurements obtained using this protocol with those published previously for normal volunteers. ROIs were defined by a neuroradiologist to cover different anatomical features and the anisotropy was measured in each region. The regions were chosen to cover a range of different grey and white matter structures, to verify

	P $1-A_{vr}$	O $1-A_{vr}$	P A_{li}	Z A_{li}	Z A_{fa}	O A_{fa}	O A_{sd}
Splenium of the corpus callosum	0.86	0.38	0.77	0.82	0.85	0.57	0.38
Optic radiations	0.62	0.31	0.54	0.49	0.58	0.50	0.33
Posterior limb of the internal capsule	0.70	0.25	0.61	0.64	0.70	0.45	0.28
U-fibres	0.55	0.28	0.46	0.50	0.52	0.47	0.30
Centrum semiovale	0.27	0.23	0.31	0.42	0.48	0.43	0.27
Caudate nucleus	0.08	0.19	0.12	0.13	0.24	0.40	0.25
Frontal cortex	0.08	0.10	0.09	0.13	0.23	0.29	0.18
CSF	0.02	0.10	0.05	0.10	0.15	0.26	0.17

Table 9.1 Anisotropy measurements obtained from different anatomical regions, comparing our data (O), with that of Pierpaoli et al. (P) (128) and Zelaya et al. (Z) (169). The compressed range of anisotropy measured in our study is clearly demonstrated.

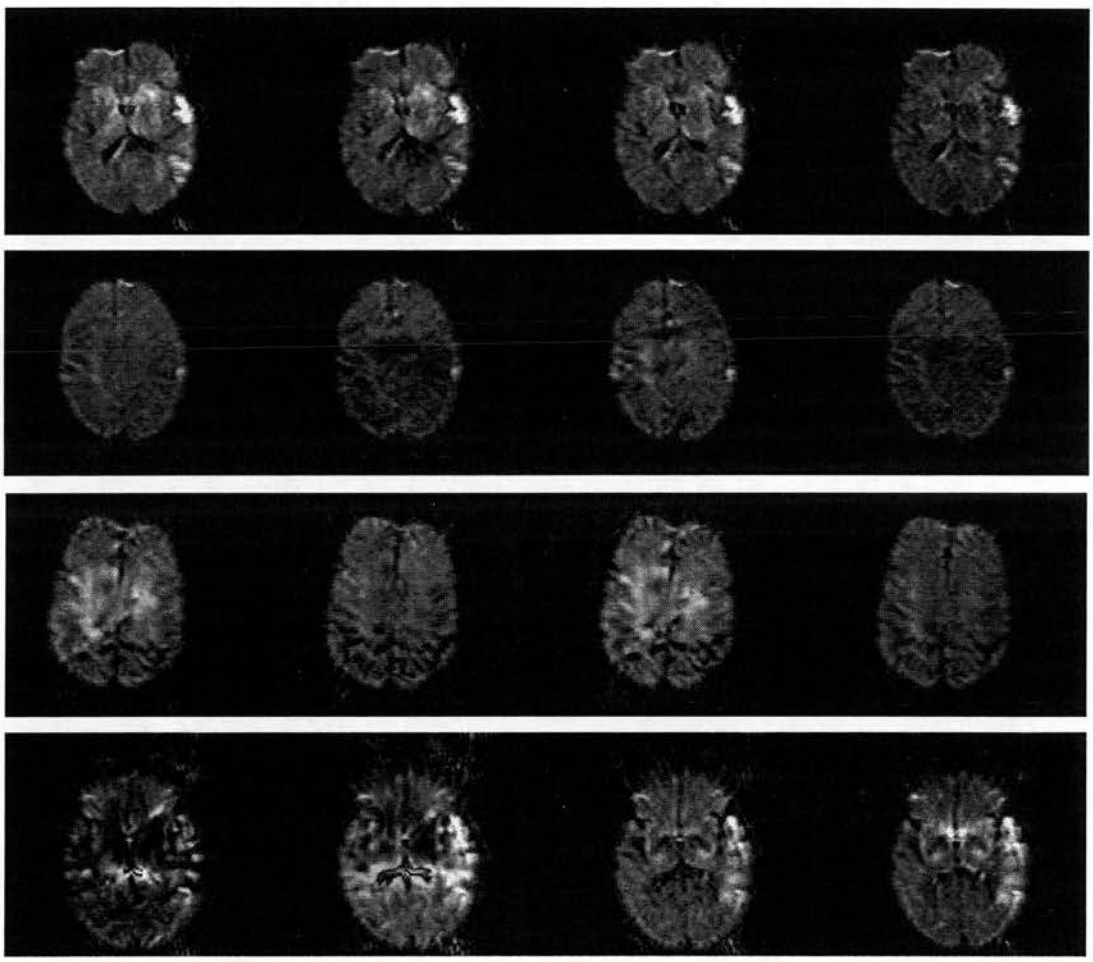


Figure 9.5 Illustration of the poor reliability of the Elscint EPI diffusion sequence (b = 1000 smm^{-2}) for in vivo imaging of acute stroke patients. Each row represents a set of images obtained with exactly identical pulse sequence parameters. A large variation of signal intensities can be observed for these supposedly identical images.

that the anisotropy was giving sensible information on the known structural ordering in white matter fibres. The measurements are summarised in Table 9.1, along with those previously presented by Pierpaoli et al. (14) and Zelaya et al. (6).

Table 9.1 demonstrates a significant difference between the data obtained in this study and that obtained in the previously published studies. There is also a small difference between the data of Pierpaoli et al. (128) and the data of Zelaya et al. (169). The difference between these two data sets can probably be ascribed to increased noise contamination or gradient mis-calibrations, in the latter study, resulting in a systematic increase in anisotropy over the full range measured. The data obtained in this study appears to be measured on a compressed scale, with

isotropic structures being seen with a relatively high anisotropy, while highly anisotropic structures appear more isotropic. As can be seen from Fig. 7.7, this is exactly the effect that is observed for images with extremely low DNR_{av} (< 10). However, as was shown in Fig. 7.13, the DNR_{av} obtained using the optimised acquisition scheme is considerably higher than this value. Therefore, the compressed range of anisotropy observed in this study must result from some other mechanism.

As was discussed in §7.6.2 and §8.3.4, the EPI images show extremely poor reproducibility within a data set. This is illustrated by several examples of acute stroke patients, in Fig. 9.5, where consecutive acquisitions are shown obtained with exactly the same imaging parameters and pulse sequence. These images demonstrate a high degree of signal variability, both in the form of 'patchy' regional variations and as global variations between acquisitions. These artefactual signal variations are likely to have a similar effect on the anisotropy measurements as a reduction in SNR (or DNR). While the worst of these images can be rejected from the tensor calculation by visually inspecting the data set, many of the images appear to have smaller signal variations that cannot be easily detected by eye, but may be significantly detrimental to the measurement of anisotropy. The exact reason for these artefacts is unclear, however, it would appear to result either from patient motion or from the limitations of the automatic shimming routine, because the artefacts are not observed on stationary phantom images. The EPI sequence has a long readout train (≈ 80ms) and patient motion may result in a distortion of the phase evolution and resulting image quality. Similarly, poor quality shimming is likely to result in image distortions due to the high B_0 inhomogeneity, as was discussed in §2.6.3. The linear (x, y and z) shim coils and optimisation algorithm on the Elscint scanner may be sufficient to shim around a uniform phantom, but may not provide the high order corrections necessary to effectively shim a complex structure such as the human head.

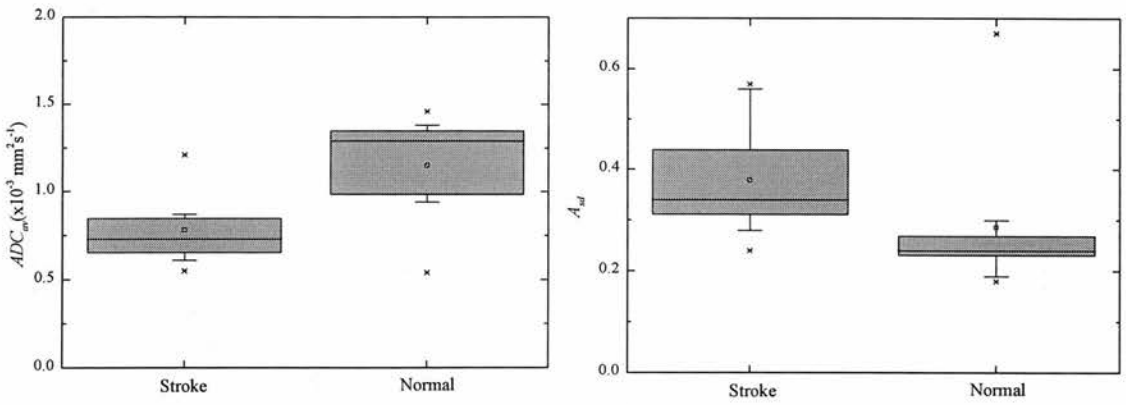


Figure 9.6 Statistical box-plots representing the ADC_{av} reduction and A_{sd} increase observed in acute stroke. The data is presented for the study performed using the DTI protocol on the Elscint scanner.

9.3.3 Diffusion measurements in acute stroke

Of the 15 acute stroke patients scanned 1 - 4 days post-ictus using the DTI protocol, 9 were demonstrated to have a visible, measurable diffusion abnormality. As described in Chapter 5, ROI measurements of ADC_{av} and A_{sd} were obtained from lesion and contralateral normal appearing regions of the brain. In total 53 ROIs were obtained from the 9 patients, ranging in size from 4 to 286 pixels (mean 47 ± 53). The data is summarised using statistical box-plots in Fig. 9.6 and clearly supports the reduction of ADC_{av} and the increase of A_{sd} within the lesions, observed in the previous two studies. The mean ADC_{av} was 0.78 ± 0.20 × 10^{-3} mm^2s^{-1} within the lesion and 1.15 ± 0.29 × 10^{-3} mm^2s^{-1} within contralateral normal appearing regions and is in good agreement with previous acute stroke studies (e.g. 21,111,135,167-169). The quantitative RI measure of A_{sd} gave values of 0.38 ± 0.12 within the lesion and 0.29 ± 0.15 for normal appearing regions. The data was demonstrated to be normally distributed, using the Anderson-Darling normality test and subsequent one-sample t-tests of the mean stroke-to-normal ratios revealed that both the reduction in ADC_{av} ($p = 0.001$), and the increase in A_{sd} ($p = 0.011$) were significant.

9.3.4 Comparison of orthogonal and tensor data sets

The theoretical differences between the orthogonal and tensor sampling strategies was discussed in Chapter 6, demonstrating that both schemes should

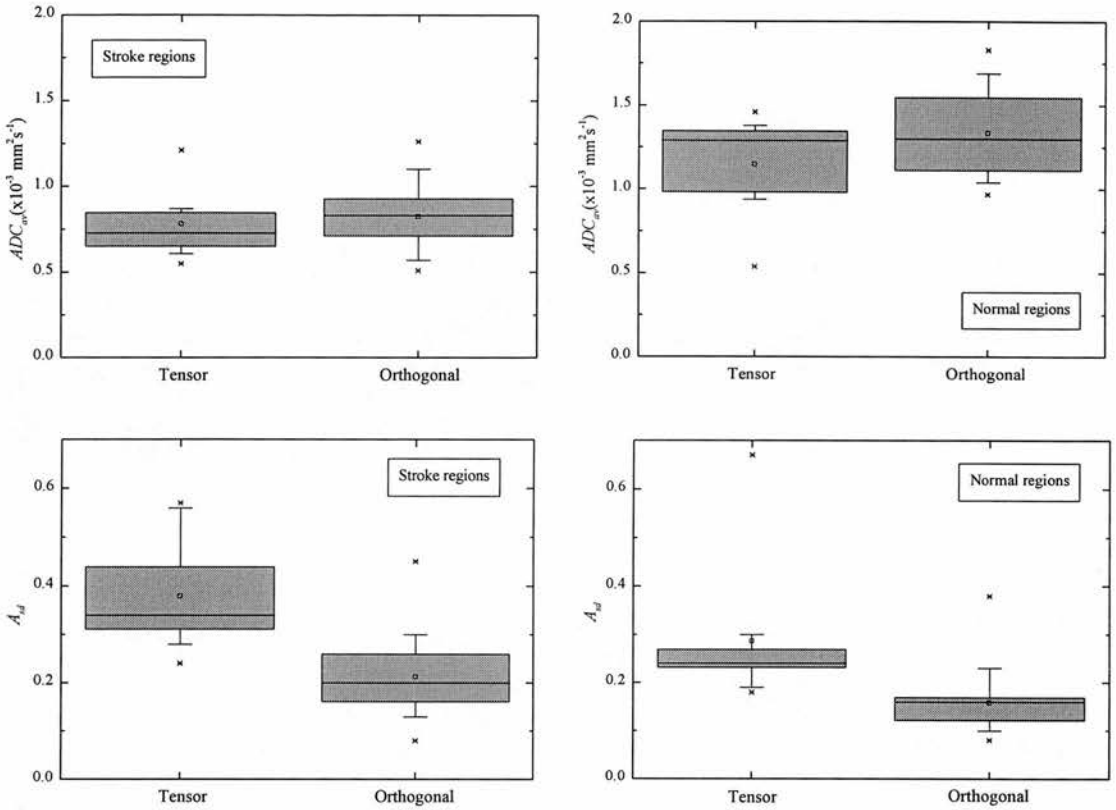


Figure 9.7 Statistical box-plots representing the differences in the measured ADC_{av} and A_{sd} in stroke and normal regions using the orthogonal and tensor acquisition schemes. The plots demonstrate that there is no significant difference in ADC_{av} measurements, but A_{sd} is considerably reduced when the orthogonal scheme is used. These findings agree with the theoretical predictions made in Chapter 6.

reliably estimate ADC_{av}, while the orthogonal technique considerably underestimates A_{sd} in many situations. Therefore, the aims of this section are to verify whether these theoretical expectations hold true in the practical imaging of acute stroke patients. The basis for the comparisons utilise the DTI data discussed in §9.3.3 and the orthogonal DWI data presented in §9.2.2. The two data sets were reasonably well matched for age ($p = 0.24$), with the orthogonal data set having mean age 67 ± 21 and the tensor 74 ± 13; and extremely well matched for TSO ($p = 0.75$), with the orthogonal being 1.7 ± 1.2 and the tensor being 1.9 ± 1.8. The differences in ADC_{av} and A_{sd} obtained using the orthogonal and tensor acquisition schemes are illustrated for both stroke and normal regions in Fig. 9.7.

As expected, Fig. 9.7 illustrates that there is no statistically significant difference between ADC_{av} measured using the orthogonal or tensor techniques, in

either stroke ($p = 0.57$) or normal regions ($p = 0.11$). However, in both regions, ADC_{av} is insignificantly higher when using the orthogonal technique. This small increase may result from using the dominating diagonal element of the *b*-matrix to calculate the diffusion attenuation from Eq. 3.14a in the orthogonal case, rather than using the full expression given in Eq. 3.23. The measured A_{sd} also follows the theoretical predictions made in Chapter 6, being considerably higher when measured using the tensor technique. This increase is statistically significant in both the stroke ($p = 0.003$) and normal ($p = 0.035$) regions, with both regions demonstrating a 45% A_{sd} reduction when obtained using the orthogonal measure. This error is reasonably close to the 35% mean underestimation predicted in Chapter 6 and the greater error may result from a non-random distribution of fibre orientations or from further experimental errors that were not considered in Chapter 6.

On a more positive note, the orthogonal and tensor data sets further provide evidence to support the success of the optimised acquisition scheme. The practice adopted throughout this work has been to reject data obtained from any voxels in which the calculated ADC is negative in any of the diffusion measurements. Clearly, negative ADCs are meaningless and the ADC_{av} and anisotropy images derived from these erroneous values are likely to be misleading and should therefore be ignored. The number of rejected pixels has been noted for each of the ROIs measured in this study and a comparison made between the number rejected in the orthogonal and tensor data sets. It would be expected that the optimised tensor acquisition scheme should ensure that all pixels are reliably measured as discussed in Chapter 8, while the orthogonal scheme may show significant noise contamination and hence some erroneous pixels. This was found to be the case where, for any given ROI, 8.7 ± 10.8 % of voxels were meaningless in the orthogonal data set, while for the tensor case only 1.0 ± 3.9 % of voxels were erroneous. This demonstrates the importance of using optimised acquisition parameters, even for the measurement of ADC_{av}, if reliable measurements are to be obtained. The erroneous data residing after optimisation can be assigned to misregistration of diffusion images, possibly resulting from eddy-current distortions or pulsatile brain motion.

9.3.5 Diffusion in grey and white matter

The differences between diffusion in grey and white matter were discussed in §9.2.4, for the data obtained using the orthogonal acquisition technique. This section aims to demonstrate diffusion differences between grey and white matter, using the data obtained with the tensor acquisition scheme. The results are summarised in Fig. 9.8 for ADC_{av} and A_{sd} obtained from stroke and normal regions. In §9.2.4, a slight increase in ADC_{av} was observed for grey matter, in both stroke and normal regions, a trend which is also observed with this patient group. The increased ADC_{av} in grey matter remains insignificant in stroke regions ($p = 0.06$), but is significantly elevated in normal regions ($p = 0.02$). Again, this difference can be ascribed to the close proximity of many grey matter structures to CSF, resulting in partial volume contamination of voxels thereby increasing ADC_{av}. The increased significance for the tensor data compared to the orthogonal can be assigned to an increased proportion of patients with cortical grey matter lesions.

The principal reason for performing a full DTI experiment, rather than just obtaining the orthogonal data set, is to enable a fully quantitative evaluation of anisotropy. As was demonstrated in §9.2.4 the orthogonal imaging technique was unable to discriminate effectively the differences in anisotropy between grey and white matter. Unfortunately, Fig. 9.8 demonstrates that the DTI acquisition scheme used in this study was also unable to detect any significant differences between grey and white matter, either in stroke ($p = 0.81$) or normal ($p = 0.30$) regions. This is a disappointing observation because it is well known that anisotropy is increased in white matter regions, a feature which has been demonstrated using DTI on many previous occasions (128,169). The probable reason for this is due to the poor quality of the EPI images used to calculate the tensor, as was discussed in §9.3.2 and illustrated in Fig. 9.5. Table 9.1 demonstrates the compressed range of anisotropy observed in a normal volunteer for a series of ROI measurements obtained from well defined grey and white matter structures. In contrast, this study observes regions that are broadly classified into grey or white matter, but many of these regions encompass a relatively large area of different anatomical features. The combined effect of the

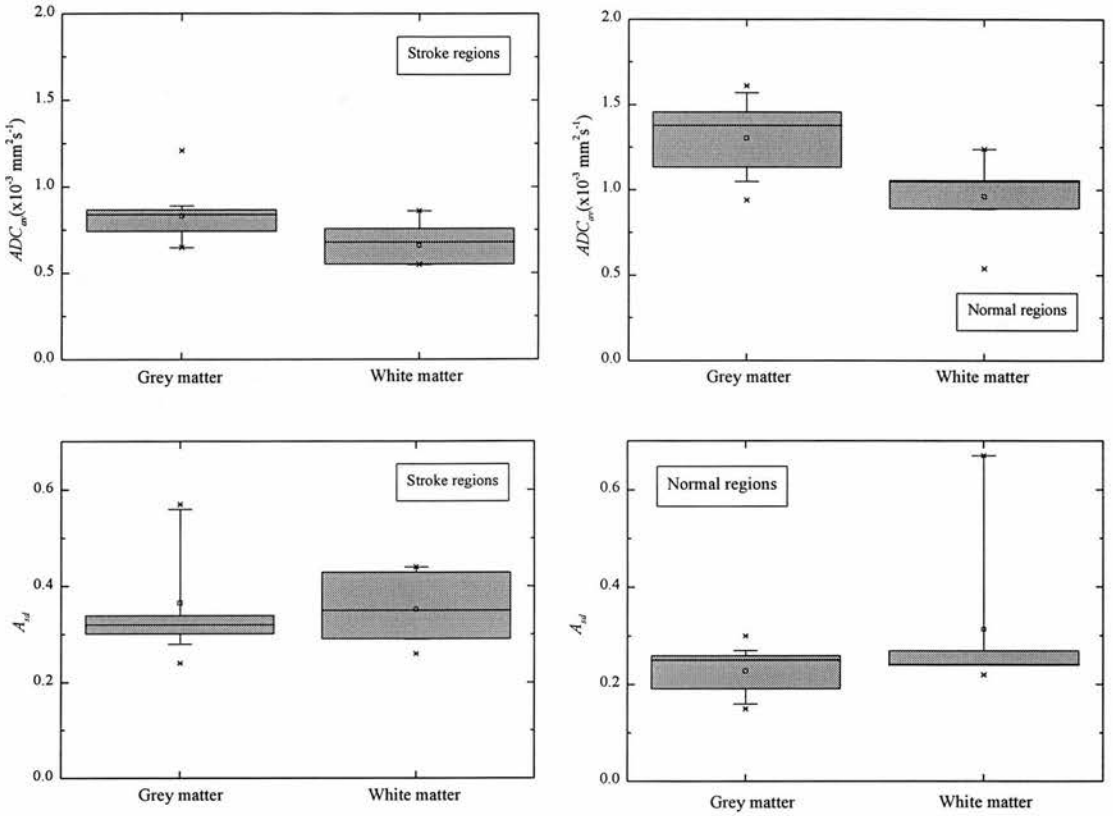


Figure 9.8 Illustration of the changes in ADC_{av} and A_{sd} for lesions evolving in grey and white matter along with the corresponding contralateral normal regions. Grey matter regions demonstrate an elevated ADC_{av} due to CSF contamination. The expected increase of A_{sd} in white matter is not observed, probably due to the poor quality of the EPI images.

relatively small number of patients studied using DTI, the compressed range of anisotropy observed and the lack of well defined grey and white matter regions is believed to lead to the lack of a significant difference between the observed anisotropy in grey and white matter regions.

9.4 DWI and head injury

9.4.1 Introduction

The usefulness of DWI in acute stroke is well documented and universally acknowledged. However, many other clinical uses for DWI and DTI have been emergent in the literature, some of which were discussed in §3.7.2. While this work has been primarily concerned with the imaging of acute stroke patients, other patient

groups have been studied on a less routine basis, notably 16 patients suffering from acute head injury.

Often associated with closed head injury are large mechanical distortions of tissue, and haematoma (blood in the brain parenchyma). These large scale structural changes may be identified readily using conventional MR and CT imaging. However, these patients often have diffuse axonal injuries, such as axonal shear and hypoxic brain damage, changes which are less well identified using conventional imaging (191-193). More recently, MR spectroscopy and spectroscopic imaging techniques have been utilised to identify the chemical changes associated with diffuse axonal injury and a reduction of NAA has been correlated with a poorer clinical outcome (194). SPECT perfusion studies have also been performed and indicate a reduced level of perfusion, even in areas where no abnormality is seen on conventional MR imaging (195). As reduced perfusion and NAA concentration are often evident along with a diffusion abnormality, it was hypothesised that diffuse axonal damage may result in a diffusion abnormality that could be identified using DWI.

The importance of identifying diffuse axonal injury using imaging is highlighted by current accident and emergency procedures. At present, all unconscious (Glasgow Coma Scale ≤ 8) or deteriorating head injured patients are immobilised, intubated, ventilated and sedated, either at the scene of the accident or upon arrival at the hospital. This limits the clinical examination of the CNS to pupilliary light response tests. After admission, the patient undergoes CT examination and if no lesion is visible, the patient undergoes a trial awakening period, without further intervention. However, at this stage some patients may not awake and are potentially at risk from secondary insults which may add to the brain injury, such as elevations in intracranial pressure (ICP). If evidence of diffuse axonal injury could be identified using MR at the initial scan stage, whether a focal abnormality was visible or not, the patient could be kept in ITU with the knowledge that a serious injury had been sustained.

The scarcity of acute head injury studies can be explained because of the anaesthetic complexity of performing such a scan, whereby the patients require continual ventilation and ICP monitoring. For this study, a dedicated research

anaesthetist (Dr Carol MacMillan) was responsible for recruiting the patients and providing their continuing intensive care during the scan. Additional image artefacts are observed due to the ICP monitoring equipment (196), which consists of an MR compatible ICP transducer and monitoring box (Codman, Johnson & Johnson Professional Inc., Raynham, MA). The presence of the monitoring equipment did not adversely affect the image quality, until the monitoring box was switched on, a procedure which resulted in a 60% reduction in SNR. This considerable reduction is likely to have serious implications for achieving reliable diffusion anisotropy measurements in acute brain injury studies, if ICP monitoring is a necessary requirement.

9.4.2 Methods and results

This head injury study was principally concerned with observing metabolite changes using spectroscopic imaging, with diffusion imaging being a lower priority. Therefore, due to the considerable time involved with imaging a patient under intensive care conditions, DWI was often omitted from the protocol or a shortened version used. As such, from the 16 patients studied only four underwent the full orthogonal imaging protocol, while the remaining 12 had a single DW image obtained with the diffusion gradient applied along the phase encoding direction. Of the four patients studied quantitatively, 3 were diagnosed with traumatic brain injury (TBI) and 1 with subarachnoid haemorrhage (SAH). All imaging was performed using the Siemens scanner and the navigated spin-echo DWI pulse sequence. The acquisition and processing was performed as described in Chapter 4.

A qualitative evaluation of the clinical usefulness of the DW images was performed for the 16 patients. All visible abnormal diffusion regions were found to be equally visible on conventional structural T_2 images. Of the four patients that underwent the full orthogonal DWI protocol, one also underwent spectroscopic imaging, two had single voxel spectroscopy and one had no spectroscopy. For the patients undergoing spectroscopy, ADC_{av} was measured from an ROI corresponding to the spectroscopy voxel of interest (VOI). For the patient that had no spectroscopy

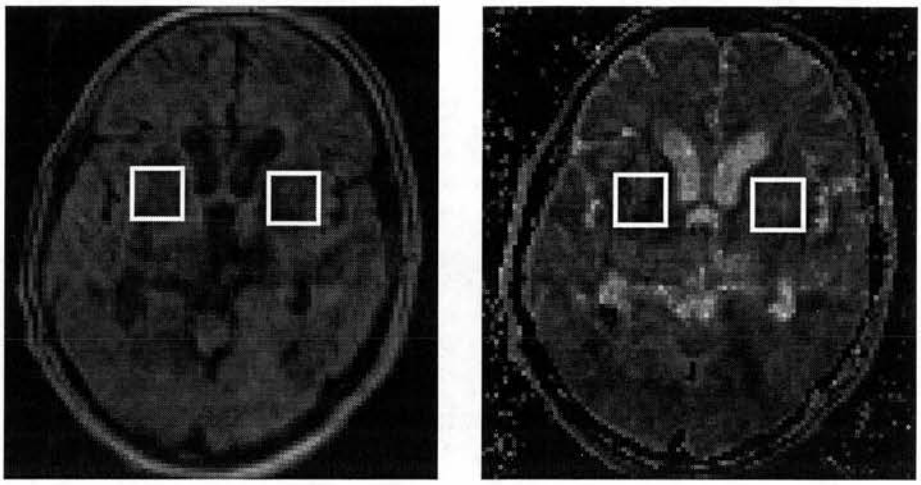


Figure 9.9 Illustration of the typical ROI positions for single voxel spectroscopy and the corresponding diffusion measurements obtained in the head injury study. The images displayed are DWI_{av} (left) and ADC_{av} (right) obtained from a 60 year old patient with SAH.

(due to technical difficulties) the ROI was positioned where the spectroscopy VOI would have been placed. The VOIs were positioned around the basal ganglia and medial temporal regions, as these are common sites of injury. A typical example of the spectroscopy voxels and corresponding diffusion ROI positions are shown in Fig. 9.9. A statistically significant reduction of ADC_{av} in the TBI patient group (0.73 ± 0.23) relative to a group of normal control regions (1.23 ± 0.46) was found ($p = 0.0001$). Of the nine measured ROIs in the TBI patient group, only two demonstrated visible T_2 abnormalities (as classified by a neuroradiologist). Also, no significant difference in ADC_{av} reduction was observed between T_2 normal or abnormal appearing regions ($p = 0.39$). The spectroscopy study also demonstrated that the NAA level was consistently reduced, relative to normal subjects, both in T_2 abnormal and T_2 normal regions of both TBI and SAH patients (194). In contrast, the mean ADC_{av} measured from the two ROIs in the SAH patient was (1.49 ± 0.32) which is elevated above normal, but is not statistically significant ($p = 0.48$), probably due to the small number of measurements. It may be that the rise of ADC_{av} in the SAH patient is due to the presence of blood in the brain, which is likely to increase the measured diffusivity. This suggestion of haematoma was also indicated by a T_2 abnormality in both of these regions. These results are suggestive that quantitative diffusion imaging and spectroscopy may provide complementary

evidence of diffuse axonal injury that is not evident from conventional images. Clearly, a larger study would be required to confirm the preliminary observations of ADC_{av} in this small group of head injured patients.

9.5 Summary of in vivo findings

This chapter has primarily been concerned with demonstrating diffusion changes occurring in stroke. Measurements of ADC_{av} were shown to be considerably reduced in acute stroke compared to contralateral normal regions. It was also demonstrated that these changes can be measured with equal reliability, using either orthogonal or diffusion tensor protocols and the values obtained were largely scanner and sequence independent. Therefore, if ADC_{av} changes and not anisotropy are of interest then it is sensible to use the orthogonal imaging technique to reduce considerably the imaging time. The temporal evolution of ADC_{av} was studied over the first 14 days following symptom onset and a general recovery was observed with time. However, some evidence was found to suggest that the form of the recovery may be different in grey and white matter, possibly indicating the different structural evolution of these tissue types in stroke. Diffusion anisotropy measurements were shown to be relatively unreliable, due to the poor quality of the EPI-DW images, whether the anisotropy was calculated using the full diffusion tensor or not. Quantitative data was also presented for a series of patients suffering from acute head injury. This is believed to be the first diffusion study of this patient group and demonstrated an ADC_{av} reduction in both T_2 normal and abnormal appearing regions, which may provide evidence of diffuse axonal injury.

10

Conclusions

10.1 Introduction

This section aims to summarise the findings of the three years work performed for this PhD project. The novel aspects of the work will be stressed and the success and limitations of the various techniques developed will be highlighted. Also, the limitations regarding technical aspects of the imaging, particularly the hardware constraints and limited access to the scanner and its software will be discussed. The aim of the project, as stated in the Introduction (Chapter 1), was to work towards a reliable quantitative assessment of diffusion in vivo. The contribution of this project towards achieving this goal is discussed and recommendations are made for further improving the techniques developed.

The general form of the thesis has been written to take the reader through the logical development of a quantitative diffusion protocol. Firstly, the necessary background information and theory was presented to provide the fundamental concepts discussed in later chapters. The methods section followed in Chapter 4, highlighting the image acquisition and post-processing procedures. Results from a preliminary study were presented in Chapter 5, demonstrating the clinical usefulness of the navigator DWI technique. A number of problems regarding the reliable acquisition of quantitative diffusion data were highlighted, particularly the rotational

variance and the degree of noise contamination. These problems were discussed in more detail in Chapters 6 and 7. Chapter 6 demonstrated that while either of the orthogonal, tetrahedral or tensor sampling schemes would provide an adequate characterisation of ADC_{av}, the whole diffusion tensor must be sampled if a reliable quantification of anisotropy is required. The effect of noise contamination on diffusion measurements was discussed in Chapter 7 and an optimised acquisition scheme proposed to allow a quantitative assessment of diffusion in the human brain. In theory, by following the procedures outlined in Chapters 6 and 7 a reliable quantitative analysis should be obtained and a discussion of the methods for displaying this quantitative data were presented in Chapter 8. A new method was developed for comparing DAIs and predicting the likely contrast produced in the resulting images. Finally, in Chapter 9, in vivo results were presented from stroke and head injured patients and the data sets were used to observe both the clinical usefulness of the technique and the effectiveness of the optimisation scheme, in a clinical environment.

10.2 Technical limitations

The work for this project was all undertaken in a clinical environment and therefore limited access was provided to the scanner, typically a maximum of 1-2 hours per week. Also, both of the scanners used were of conventional specification, with relatively low maximum gradient strengths (10 & 15 mTm^{-1}) and slow rise times (10 & 30 $mTm^{-1}ms^{-1}$). High gradient performance is essential to achieve reliable diffusion images because the diffusion attenuation is proportional to the square of the gradient strength and fast rise times allow for reliable EPI which freezes out motion artefacts. While the Elscint scanner is equipped with EPI sequences, the slow rise times are considerably below the recommended for EPI of around 100 $mTm^{-1}ms^{-1}$. Similarly, access to the pulse sequence code was restricted; with complete access being available to a limited number of sequences on the Siemens scanner; and limited access being available to all sequences on the Elscint scanner. These constraints have meant that only simple modifications to the pulse sequences

were possible and so the project has been largely reliant on the sequences provided by the manufacturers.

10.3 Theoretical findings

i) Errors arising from using the orthogonal encoding scheme

Many studies have been performed using the orthogonal imaging technique, both to measure ADC_{av} and diffusion anisotropy. It has since been suggested that these measurements will be rotationally variant and depend on the orientation of the sample within the magnet. However, a quantitative assessment of the magnitude of these rotation induced errors has not been previously reported. Chapter 6 addressed this issue, demonstrating that by theoretical considerations alone, the anisotropy is likely to be underestimated greatly using the orthogonal imaging technique. An average underestimation of around 35% was demonstrated, however, the magnitude of the error varied greatly depending on the orientation of the sample. In theory, orthogonal measurements of ADC_{av} were found to be rotationally invariant, and this was demonstrated to be the case to within experimental error in subsequent in vivo studies.

ii) Errors arising from using the tetrahedral encoding scheme

The recently proposed tetrahedral encoding scheme potentially allowed for a full evaluation of diffusion in an axisymmetric environment. As many regions of brain tissue are believed to consist of axisymmetric fibres, this technique offered great potential for in vivo imaging, allowing a complete description of diffusion with a reduced number of acquisitions. However, the detailed simulations performed in Chapter 6 demonstrate that anisotropy measurements obtained using this technique are exquisitely sensitive to slight deviations from the axisymmetric condition (which could be induced by noise). It was shown that the maximum error can occur when the diffusion is minutely asymmetric and that this error does not consistently increase with further deviation from the axisymmetric case. Also, the errors arising are highly dependent on the assignment and orientation of the coordinate system for non-axisymmetric diffusion. However, as with the orthogonal case, measurements of

ADC_{av} were found to be free from the effects of ellipsoid symmetry and rotational variance.

iii) Quantifying the degree of noise contamination in diffusion measurements

Previous studies have characterised noise contamination by the SNR in the diffusion-weighted image. This measure was shown in Chapter 7 to be highly dependent on experimental parameters, such as *b*-value and diffusivity. Therefore, it is not possible to quote a value for a minimum acceptable SNR that is generally applicable. This study demonstrated that DNR_{av} (the SNR in the tr(***D***) image), provides a measurement of noise contamination that is largely independent of experimental parameters. Simulations demonstrated that a $DNR_{av} > 50$ was necessary to provide acceptable noise independent anisotropy measurements. This facilitates the simple verification of an experiments immunity to noise contamination by the measurement of this single parameter, rather than undergoing lengthy tests for each experimental situation.

iv) Optimisation of diffusion acquisition parameters

The reliable measurement of diffusivities involves a careful balance between the amount of signal attenuation and the noise contamination in each measurement. An error analysis was recently proposed for two-point diffusion measurements which allowed for the optimum *b*-value to be found for a given measurement and demonstrated that the accuracy can be improved by performing an optimised signal averaging scheme. However, this signal averaging scheme was only optimised for an individual diffusivity and not for a range. In Chapter 7, this scheme was extended to produce optimised experimental parameters for imaging the range of diffusivities typically encountered in the human brain. The acquisition parameters were presented for a range of *b*-values and noise levels, although the optimum *b*-value was shown to be around 900 smm^{-2} for human brain imaging. This improved optimisation scheme implicitly assumes an isotropic model for diffusion and so care is required in its application. Future work could involve the extension of this optimisation scheme to include a more realistic anisotropic model of diffusion and also optimisation schemes could be developed for multiple *b*-value diffusion acquisition schemes.

v) Comparison of DAIs

Many scalar DAIs have been proposed in the literature to characterise the shape of the diffusion ellipsoid. A method was devised in Chapter 8 for predicting the sensitivity of the different DAIs to anisotropy and hence to predict the likely contrast produced in DAI maps. This characterisation of DAI sensitivity lead to the development of a new anisotropy index (A_{gv}) that improved grey-white matter delineation in the human brain, while retaining a low sensitivity to noise. The suitability of a given anisotropy index was shown to be highly dependent on the imaging situation and anisotropy indices such as A_{fa} and A_{er} were shown to be highly sensitive to noise contamination. In general, two conventional scalar anisotropy indices were required in order that an unambiguous characterisation of the diffusion ellipsoid shape could be obtained. The prolateness index was suggested as a suitable physically meaningful measure of ellipsoid asymmetry, which when used in conjunction with conventional anisotropy indices provides a complete description of diffusion ellipsoid shape. Due to the increasing number of DAIs defined in the literature, it is suggested that a standard reference is adopted to allow for comparison between research groups. An appropriate index would be A_{sd}, because this is a physically meaningful, normalised index which conforms to a previously defined chemical definition of anisotropy.

10.4 In vivo findings

i) Diffusion and acute stroke

A significant reduction in ADC_{av} was shown in Chapter 9 for a study of over 100 acute stroke patients, a result which has been demonstrated on many previous occasions. The values obtained from the navigated spin-echo orthogonal protocol, the EPI orthogonal protocol and the diffusion tensor EPI protocol were all equivalent to within experimental error. Some evidence was found to suggest that the temporal evolution of ADC_{av} in grey and white matter shows a different evolutionary pattern for the different tissue types, which may be suggestive of different structural breakdown mechanisms. This is believed to be the first study classifying the diffusion evolution in terms of evolving grey and white matter lesions, although a

much larger study would be required to verify these observations. In vivo anisotropy measurements were of poor quality, both for orthogonal and tensor acquisition methods, with the measured anisotropy being on a compressed scale compared to other published studies. This is believed to be due to the poor quality of the EPI images obtained, probably resulting from the low specification hardware and poor shimming routines. It would be extremely interesting to observe the anisotropy evolution within evolving grey and white matter lesions, as this may indicate the point at which cell breakdown and irreparable damage occurs. However, such measurements would require a reliable set of DW images from which the diffusion tensor could be calculated.

ii) Diffusion and acute brain injury

A small study of one SAH and three TBI patients demonstrated a significant reduction of ADC_{av} in TBI patients within regions that appeared normal on both T_2 and DW images, while an increase of ADC_{av} was observed in the SAH patient. The ADC_{av} decrease in TBI correlated well with a reduction of NAA concentration, demonstrated by MR spectroscopy techniques. This ADC_{av} reduction may provide evidence of diffuse axonal injury, a serious condition that is not visible using conventional imaging techniques. The logistics of scanning patients under intensive care conditions, with constant physiological monitoring has meant that studies of head injured patients have been limited and this is believed to be the first study of this patient group utilising quantitative diffusion imaging. Again, anisotropy information could provide a further understanding of the biophysical processes involved in these diffusion changes and a future study utilising DTI techniques may provide valuable information.

10.5 Final summary

The recurring theme of this thesis has been the difficulty in obtaining reliable anisotropy measurements. In contrast, measurements of ADC_{av} appear to be reasonably tolerant of image artefacts, noise contamination and acquisition method. The findings of this thesis suggest that DTI is likely to remain a pure research tool

for the foreseeable future, due to the need for state of the art imaging hardware and complex post-processing procedures. Nevertheless, this work demonstrates that it is possible to provide a quantitative analysis of diffusion, provided that sufficient image quality is obtained in an acceptable scan time. The potential to perform high speed imaging with conventional gradients using sequences such as GRASE may enable the use of DTI with current clinical systems, however this study suggests that the limited SNR of these techniques may preclude a reliable assessment of anisotropy in an acceptable imaging time. Therefore, it is unlikely that DTI will become a widespread clinical utility until high performance gradients become available on clinical scanners.

A1

Computer Source Code

A1.1 dse.c

```
/* This program is written to calculate the ADC and anisotropy
images reading */
/* in the input from the respective T2, X, Y and Z images.
The program */
/* calculates for any usual combination of b values and FOV's
*/

/* Originally written: 5th December 1996 */

/* Last updated: 20th January 1998 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define n_rows 128
#define n_cols 128

double **imagex, **imagey, **imagez, **imager, **ad_image;
double **imageax, **imageay, **imageaz, row, col;
double **anivg_image, **adc_image, **ave_image, **anivr_image;
FILE   *data_filex, *data_filey, *data_filez, *data_filer;
FILE   *out_file1, *out_file2, *out_file3, *out_file4;
char   file_name[15], file_num[6], pat_name[29], st_name[10];
char   study_date[10], NACQ_string[5], msg[30];
char   input_type[3];
char   image_r[6], image_x[6], image_y[6], image_z[6];
char   image_out1[6], image_out2[6], image_out3[6],
image_out4[6];
char   seven[]="7", eight[]="8", nine[]="9", ten[]="10",
slash[]="/";
char   in1[4], in2[4];
int    im_x, im_y, im_z, im_r, slices, n_slices, in_type,
b_val, fov, b_type;
int    no_slices;

char   in_path[60] = "/export/home/paul/diffusion/";


int    read_image_file(double **rdata, char *fname, FILE
*data_file, char *st_name, int in_type);
int    read_int(FILE *data_file);
void   save_int(int value, FILE *outfile);
long   read_long(FILE *data_file);
double ani_images(int b_val, int b_type, int fov);
int    save_image_file(char *fname, FILE *out_file, double
**calc, char *file_no, char *st_name, int in_type, char
*image_x);

/* functions taken from Numerical Recipes     */
void   nrerror(char error_text[]);
int    **imatrix(int nrl,int nrh,int ncl,int nch);
void   free_imatrix(int **m,int nrl,int nrh,int ncl,int nch);
double **dmatrix(int nrl, int nrh, int ncl, int nch);
void   free_dmatrix(double **m, int nrl, int nrh, int ncl, int
nch);


/*************************************************************
**********/

main()
{

           /* Main Menu */

           printf("\n ***ORTHOGONAL DIFFUSION PROTOCOL
MENU***\n\n");

           printf("\n Please enter the following
information:\n");

           printf("\n The study name: ");
           gets(st_name);

           printf("\n The type of input data used:\n");
           printf(" 1 - Non-corrected\n");
           printf(" 2 - Independent zero plus first
order\n");
           printf(" 5 - Zero plus first order\n");
           printf(" 6 - Zero order\n");
           scanf("%i", &in_type);

           printf("\n The first reference (T2/D0) image
number: ");
           scanf("%i", &im_r);

           printf("\n The first image number with x-diffusion
weighting: ");
           scanf("%i", &im_x);

           printf("\n The number of slices (max. = 8): ");
           scanf("%i", &n_slices);

           if(n_slices == 8)
           printf("\n ***Repeat program twice adding 4 to D0
and DX image numbers***\n");

           printf("\n The field of view used (120 or 230):
");
           scanf("%i", &fov);

           printf("\n The nominal scalar b-value used: ");
           scanf("%i", &b_val);

           printf("\n The type of b_value analysis
required:\n");
           printf(" 1 - Nominal scalar value in all
directions\n");
           printf(" 2 - Numerical read, phase and slice
values\n");
           printf(" 3 - Numerical phase, swap phase and slice
values\n");
           scanf("%i", &b_type);

           sprintf(in1, "0");
           sprintf(in2, "%d",in_type);
           strcat(input_type, in1);
           strcat(input_type, in2);

           if(n_slices == 8)
           no_slices = 4;
           else
           no_slices = n_slices;

           /* Loop to repeat for n slices incrementing the
file nos each time */

           for (slices=1; slices<=no_slices; slices++)
           {

           /* Reset image file strings */
```

```
            char  image_r[6]="", im_r2[4]="", im_r3[3]="";
            char  image_x[6]="", im_x2[4]="", im_x3[3]="";
            char  image_y[6]="", im_y2[4]="", im_y3[3]="";
            char  image_z[6]="", im_z2[4]="", im_z3[3]="";
            char  image_out1[6]="", image_out2[6]="";
            char  image_out3[6]="", image_out4[6]="";

            /* Define image type required */

            sprintf(image_r, input_type);
            sprintf(image_x, input_type);
            sprintf(image_y, input_type);
            sprintf(image_z, input_type);
            sprintf(image_out1, "07");
            sprintf(image_out2, "08");
            sprintf(image_out3, "09");
            sprintf(image_out4, "10");

            /* Program expects Dx, Dy and Dz images to be
separated by 2*slices */

            im_y = im_x + 2 * n_slices;
            im_z = im_x + 4 * n_slices;

            /* Manipulate image file number string for
reference T2 image */

            if (im_r <= 9)  {
            sprintf(im_r2, "00");
            sprintf(im_r3, "%d", im_r);
            strcat(image_r, im_r2);
            strcat(image_r, im_r3);  }

            if (im_r <= 99 && im_r >= 10)  {
            sprintf(im_r2, "0");
            sprintf(im_r3, "%d", im_r);
            strcat(image_r, im_r2);
            strcat(image_r, im_r3);  }

            if (im_r >= 100) {
            sprintf(im_r2, "%d", im_r);
            strcat(image_r, im_r2); }

            /* Manipulate image file number string for Dx
image */
            /* Program also saves output images with last 3
digits of Dx image */
            /* prefixed by relevant input type file number */

            if (im_x <= 9)  {
            sprintf(im_x2, "00");
            sprintf(im_x3, "%d", im_x);
            strcat(image_x, im_x2);
            strcat(image_x, im_x3);
            strcat(image_out1, im_x2);
            strcat(image_out1, im_x3);
            strcat(image_out2, im_x2);
            strcat(image_out2, im_x3);
            strcat(image_out3, im_x2);
            strcat(image_out3, im_x3);
            strcat(image_out4, im_x2);
            strcat(image_out4, im_x3); }

            if (im_x <= 99 && im_x >= 10)  {
            sprintf(im_x2, "0");
            sprintf(im_x3, "%d", im_x);
            strcat(image_x, im_x2);
            strcat(image_x, im_x3);
            strcat(image_out1, im_x2);
            strcat(image_out1, im_x3);
            strcat(image_out2, im_x2);
            strcat(image_out2, im_x3);
            strcat(image_out3, im_x2);
            strcat(image_out3, im_x3);
            strcat(image_out4, im_x2);
            strcat(image_out4, im_x3);  }

            if (im_x >= 100) {
            sprintf(im_x2, "%d", im_x);
            strcat(image_x, im_x2);
            strcat(image_out1, im_x2);
            strcat(image_out2, im_x2);
            strcat(image_out3, im_x2);
            strcat(image_out4, im_x2);}

            /* Manipulate image file number string for Dy
image */

            if (im_y <= 9)  {
            sprintf(im_y2, "00");
            sprintf(im_y3, "%d", im_y);
            strcat(image_y, im_y2);
            strcat(image_y, im_y3);  }

            if (im_y <= 99 && im_y >= 10)  {
            sprintf(im_y2, "0");
            sprintf(im_y3, "%d", im_y);
            strcat(image_y, im_y2);
            strcat(image_y, im_y3);  }

            if (im_y >= 100) {
            sprintf(im_y2, "%d", im_y);
            strcat(image_y, im_y2); }

            /* Manipulate image file number string for Dz
image */

            if (im_z <= 9)  {
            sprintf(im_z2, "00");
            sprintf(im_z3, "%d", im_z);
            strcat(image_z, im_z2);
            strcat(image_z, im_z3);  }

            if (im_z <= 99 && im_z >= 10)  {
            sprintf(im_z2, "0");
            sprintf(im_z3, "%d", im_z);
            strcat(image_z, im_z2);
            strcat(image_z, im_z3);  }

            if (im_z >= 100) {
            sprintf(im_z2, "%d", im_z);
            strcat(image_z, im_z2); }

            /* declare image arrays */

            imagex=dmatrix(1,n_rows,1,n_cols);
            imagey=dmatrix(1,n_rows,1,n_cols);
            imagez=dmatrix(1,n_rows,1,n_cols);
            imager=dmatrix(1,n_rows,1,n_cols);
            imageax=dmatrix(1,n_rows,1,n_cols);
            imageay=dmatrix(1,n_rows,1,n_cols);
            imageaz=dmatrix(1,n_rows,1,n_cols);
            ave_image=dmatrix(1,n_rows,1,n_cols);
            anivg_image=dmatrix(1,n_rows,1,n_cols);
            anivr_image=dmatrix(1,n_rows,1,n_cols);
            adc_image=dmatrix(1,n_rows,1,n_cols);
            ad_image=dmatrix(1,n_rows,1,n_cols);

            /* Load image files */

            if (read_image_file(imager,image_r, data_filer,
st_name, in_type)!=0)
                  {
                        printf("Error reading data file\n");
                        return(1);
            }

            if (read_image_file(imagex,image_x, data_filex,
st_name, in_type)!=0)
                  {
                    printf("Error reading data file\n");
                    return(1);
                  }


            if (read_image_file(imagey,image_y, data_filey,
st_name, in_type)!=0)
                  {
                    printf("Error reading data file\n");
                    return(1);
                  }


            if (read_image_file(imagez,image_z, data_filez,
st_name, in_type)!=0)
                  {
                    printf("Error reading data file\n");
                    return(1);
                  }

            /* Calculate adc, ave and anisotropy images */

            ani_images(b_val, b_type, fov);

            /* Save calculated image files */

            if (save_image_file(image_out1, out_file1,
ave_image, seven, st_name, in_type, image_x)!=0)
                  {
                    printf("Error saving data\n");
                    return(1);
                  }

            if (save_image_file(image_out2, out_file2,
ad_image, eight, st_name, in_type, image_x)!=0)
                  {
                    printf("Error saving data\n");
                    return(1);
                  }

            if (save_image_file(image_out3, out_file3,
anivg_image, nine, st_name, in_type, image_x)!=0)
                  {
                    printf("Error saving data\n");
                    return(1);
                  }

            if (save_image_file(image_out4, out_file4,
anivr_image, ten, st_name, in_type, image_x)!=0)
                  {
                    printf("Error saving data\n");
                    return(1);
                  }

            /* Close data files */


            fclose(data_filer);
            fclose(data_filex);
            fclose(data_filey);
            fclose(data_filez);
            fclose(out_file1);
            fclose(out_file2);
            fclose(out_file3);
            fclose(out_file4);

            /* free memory */

            free_dmatrix(ave_image,1,n_rows,1,n_cols);
            free_dmatrix(adc_image,1,n_rows,1,n_cols);
            free_dmatrix(ad_image,1,n_rows,1,n_cols);
            free_dmatrix(anivg_image,1,n_rows,1,n_cols);
            free_dmatrix(anivr_image,1,n_rows,1,n_cols);
            free_dmatrix(imager,1,n_rows,1,n_cols);
            free_dmatrix(imagez,1,n_rows,1,n_cols);
            free_dmatrix(imagey,1,n_rows,1,n_cols);
            free_dmatrix(imagex,1,n_rows,1,n_cols);
            free_dmatrix(imageax,1,n_rows,1,n_cols);
            free_dmatrix(imageay,1,n_rows,1,n_cols);
            free_dmatrix(imageaz,1,n_rows,1,n_cols);

            /* Increment image file number */

            im_x++;
            im_r++;
            }

            return(0);

}     /* main */

/*************************************************************
***********/


int   read_image_file(double **rdata, char *fname, FILE
*data_file, char *st_name, int in_type)

{
            char    full_file_name[60], tempy[60];
            int     row, col, i;
```

```
            sprintf(tempy, "%d",in_type);
            strcpy(full_file_name,in_path);
            strcat(full_file_name,st_name);
            strcat(full_file_name,slash);
            strcat(full_file_name,tempy);
            strcat(full_file_name,slash);
            strcat(full_file_name,fname);
            strcat(full_file_name,".ima");
            printf("Reading from %s\n",full_file_name);
            if ((data_file=fopen(full_file_name,"rb"))==NULL)
return(1);

            /* read header */
            fseek(data_file,0x1C1A,SEEK_SET);
            fgets(file_num,6,data_file);
            fseek(data_file,0x1C23,SEEK_SET);
            fgets(pat_name,29,data_file);
            fseek(data_file,0x1C75,SEEK_SET);
            fgets(study_date,10,data_file);
            fseek(data_file,0x1D1C,SEEK_SET);
            fgets(NACQ_string,5,data_file);


            printf("\n");
            printf("Name        : %s\n",pat_name);
            printf("Study date  : %s\n",study_date);
            printf("File number : %s\n",file_num);
            printf("%s acquisitions\n\n",NACQ_string);

            /* extract data */
            fseek(data_file,0x2000,SEEK_SET);
            for (row=1; row<=n_rows; row++)
            {
                        for (col=1; col <= n_cols; col++)
                        {
                                    rdata[row][col]  =
read_int(data_file);
                        }
            }

            return(0);

}  /*   read_data_file   */

/*************************************************************
************/

int  read_int(FILE *data_file)

{
            long  value, temp;
            int   j;
            char  b[2];

            for (j=0; j<2; j++) b[j]=fgetc(data_file);

            temp=b[1] & 0xFF;
            temp=temp << 8;
            value=temp | (b[0] & 0xFF);
            return(value);

}   /* read_int  */

/*************************************************************
***********/

void save_int(int value, FILE *out_file)

/* save 'value' in byte-reversed order in out_file */

{
            char  low, high;


            high=(value & 0xFF00) >> 8;

            low =value & 0xFF;

            fputc(low,out_file);

            fputc(high,out_file);


}


/*************************************************************
***********/


long read_long(FILE *data_file)


{
            long  value, temp;

            int   j;

            char  b[4];


            for (j=0; j<4; j++) b[j]=fgetc(data_file);


            value=b[3] & 0xFF;

            value=value << 24;

            temp=b[2] & 0xFF;

            temp=temp << 16;

            value=value | temp;

            temp=b[1] & 0xFF;

            temp=temp << 8;

            value=value | temp;

            value=value | (b[0] & 0xFF);

            return(value);


}   /* read_long  */

/*************************************************************
**************/

double ani_images(int b_val, int b_type, int fov)

{
            int row,col;
            double pl = 0.051;
            double gd;
            double dt = 0.05744;
            double gy = 267500000.0;
            double bval;
            double sq = 2.0;
            double cu = 3.0;
            double frac = 0.1666666666666666;
            double bx, by, bz;

            if(b_type == 1)
            {
                        if(b_val == 700)
                        gd = 0.009643108;
                        else
                        if(b_val == 600)
                        gd = 0.00892845;
                        else
                        if(b_val == 500)
                        gd = 0.008150522;
                        else
                        if(b_val == 400)
                        gd = 0.007290049;
                        else
                        if(b_val == 300)
                        gd = 0.006313367;
                        else
                        if(b_val == 200)
                        gd = 0.005154843;
                        else
                        if(b_val == 100)
                        gd = 0.003645024;

                        bx = (pow((gy * gd * pl), sq)) * (dt -
(pl / 3.0));
                        by = (pow((gy * gd * pl), sq)) * (dt -
(pl / 3.0));
                        bz = (pow((gy * gd * pl), sq)) * (dt -
(pl / 3.0));

            }
            else
            if(b_type == 2)
            {
                        if(fov == 230)
                        {

                                    if(b_val == 700) {
                                    bx = 733580000.0;
                                    by = 699730000.0;
                                    bz = 696250000.0;       }
                                    else
                                    if(b_val == 500)  {
                                    bx = 528620000.0;
                                    by = 499920000.0;
                                    bz = 496990000.0;       }
                                    else
                                    if(b_val == 300)  {
                                    bx = 322350000.0;
                                    by = 300000000.0;
                                    bz = 297730000.0;       }
                                    else
                                    if(b_val == 100)  {
                                    bx = 113150000.0;
                                    by = 100000000.0;
                                    bz = 98710000.0;        }
                        }
                        else
                        if(fov == 120)
                        {
                                    if(b_val == 700) {
                                    bx = 763550000.0;
                                    by = 699730000.0;
                                    bz = 701360000.0;       }
                                    else
                                    if(b_val == 500)  {
                                    bx = 528620000.0;
                                    by = 499920000.0;
                                    bz = 496990000.0;       }
                                    else
                                    if(b_val == 300)  {
                                    bx = 322350000.0;
                                    by = 300000000.0;
                                    bz = 297730000.0;       }
                                    else
                                    if(b_val == 100)  {
                                    bx = 113150000.0;
                                    by = 100000000.0;
                                    bz = 98710000.0;        }


                                    }
            }

            else
            if(b_type == 3)
            {

                        if(b_val == 700) {
                        bx = 699730000.0;
                        by = 699730000.0;
                        bz = 696250000.0;       }
                        else
                        if(b_val == 500)  {
                        bx = 499920000.0;
                        by = 499920000.0;
                        bz = 496990000.0;       }
                        else
```

```
                        if(b_val == 300)  {
                        bx = 300000000.0;
                        by = 300000000.0;
                        bz = 297730000.0;       }
                        else
                        if(b_val == 100)  {
                        bx = 100000000.0;
                        by = 100000000.0;
                        bz = 98710000.0;        }

            }

            printf("bx is %10.2f \n",bx);
            printf("by is %10.2f \n",by);
            printf("bz is %10.2f \n\n",bz);

                        for (row=1; row<=n_rows; row++)
            {
               for (col=1; col<=n_cols; col++)
               {
                        imageax[row][col] = -
(log((float)imagex[row][col] / (float)imager[row][col])) / bx;
                        imageay[row][col] = -
(log((float)imagey[row][col] / (float)imager[row][col])) / by;
                        imageaz[row][col] = -
(log((float)imagez[row][col] / (float)imager[row][col])) / bz;
                        if (imageax[row][col] < 0.0) {
                                    anivg_image[row][col] =
1400;
                                    anivr_image[row][col] = -
1.0;
                                    ad_image[row][col] = -1.0;
}
                        else
                                    if (imageay[row][col] <
0.0) {

            anivg_image[row][col] = 1400;

            anivr_image[row][col] = -1.0;

            ad_image[row][col] = -1.0; }
                        else
                                    if (imageaz[row][col] <
0.0) {

            anivg_image[row][col] = 1400;

            anivr_image[row][col] = -1.0;

            ad_image[row][col] = -1.0; }
                        else {

            adc_image[row][col]=((float)imageax[row][col] +
(float)imageay[row][col] + (float)imageaz[row][col])/3.0;
                                    anivg_image[row][col] =
1200.0*  (sqrt(frac))*(sqrt((pow((((float)imageax[row][col]) -
((float)adc_image[row][col])), sq)) +
(pow((((float)imageay[row][col]) -
((float)adc_image[row][col])), sq)) +
(pow((((float)imageaz[row][col]) -
((float)adc_image[row][col])), sq)) /
((float)adc_image[row][col])));
                                    anivr_image[row][col] =
1200.0*(((float)imageax[row][col]*(float)imageay[row][col]*(fl
oat)imageaz[row][col])/(pow((((float)imageax[row][col]+(float)
imageay[row][col]+(float)imageaz[row][col])/3),cu)));

            ad_image[row][col]=((float)imageax[row][col] +
(float)imageay[row][col] +
(float)imageaz[row][col])/(3.0*0.0000000000005);
                                    }
                                    ave_image[row][col]=
((float)imagex[row][col] + (float)imagey[row][col] +
(float)imagez[row][col])/3.0;

                        if (anivr_image[row][col] < 0.0)
                        anivr_image[row][col] = -1.0;
                        if (anivr_image[row][col] > 1200.0)
                        anivr_image[row][col] = -1.0;
                        if (anivg_image[row][col] < 0.0)
                        anivg_image[row][col] = 1400.0;
                        if (anivg_image[row][col] > 1200.0)
                        anivg_image[row][col] = 1400.0;
               }
            }


}           /* ani_images */


/*************************************************************
**************/

int   save_image_file(char *fname, FILE *out_file, double
**calc, char *file_no, char *st_name, int in_type, char
*image_x)

/* save the calculated image file with the given file number
*/


{

            int     row, col;
            char    full_file_name[60], label[30], tempr[60];
            char    ch;
            long    tail_pos;
            FILE    *data_file;
            char    header_path[80] = "";

            sprintf(tempr, "%d",in_type);
            strcpy(header_path,in_path);
            strcat(header_path,st_name);
            strcat(header_path,slash);
            strcat(header_path,tempr);
            strcat(header_path,slash);
            strcat(header_path,image_x);
            strcat(header_path,".ima");
            strcat(header_path,slash);

            strcpy(full_file_name,in_path);
            strcat(full_file_name,st_name);
            strcat(full_file_name,slash);
            strcat(full_file_name,file_no);
            strcat(full_file_name,slash);
            strcat(full_file_name,fname);
            strcat(full_file_name,".ima");
            printf("Saving image as %s\n",full_file_name);
            if ((out_file=fopen(full_file_name,"wb"))==NULL)
return(1);

            /* copy header */
            data_file=fopen(header_path,"rb");
            fseek(data_file,0x0000,SEEK_SET);
            while (ftell(data_file) < 0x2000)
            {
                        ch=fgetc(data_file);
                        fputc(ch,out_file);
            }

            /* now save data */
            for (row=1; row<=n_rows; row++)
            {
                        for (col=1; col <= n_cols; col++)
                        {

            save_int((int)calc[row][col], out_file);
                        }
            }

            /* save tailer */
            tail_pos=(long)8*1024 + (long)n_rows*n_cols*2;
            fseek(data_file,tail_pos,SEEK_SET);
            while ((ch=fgetc(data_file))!=EOF)
fputc(ch,out_file);

            fclose(out_file);

            return(0);

}    /* save_image_file */

/*************************************************************
**********/


void nrerror(char error_text[])


/* Print an error message and exit the program
*/

{

            fprintf(stderr,"Numerical Recipes run-time
error...\n");

            fprintf(stderr,"%s\n",error_text);

            fprintf(stderr,"...now exiting to system...\n");

            exit(1);

}       /* nrerror */


/*************************************************************
**********/


int **imatrix(int nrl,int nrh,int ncl,int nch)


{

            int i,**m;


            m=(int **)malloc((unsigned) (nrh-
nrl+1)*sizeof(int*));

            if (!m) nrerror("allocation failure 1 in
imatrix()");

            m -= nrl;


            for(i=nrl;i<=nrh;i++) {

                        m[i]=(int *)malloc((unsigned) (nch-
ncl+1)*sizeof(int));

                        if (!m[i]) nrerror("allocation failure
2 in imatrix()");

                        m[i] -= ncl;

            }

            return m;

}


/*************************************************************
**********/


void free_imatrix(int **m,int nrl,int nrh,int ncl,int nch)

{

            int i;


            for(i=nrh;i>=nrl;i--) free((char*) (m[i]+ncl));

            free((char*) (m+nrl));

}
```

```
/*************************************************************
******************/
double **dmatrix(int nrl,int nrh,int ncl,int nch)

/* Declare a dynamic, two dimensional double precision matrix
with

   elements [nrl..nrh][ncl..nch]      */


{

          int i;

          double **m;


          m=(double **) malloc((unsigned) (nrh-
nrl+1)*sizeof(double*));

          if (!m) nrerror("allocation failure 1 in
dmatrix()");

          m -= nrl;


          for(i=nrl;i<=nrh;i++) {

                    m[i]=(double *) malloc((unsigned)
(nch-ncl+1)*sizeof(double));

                    if (!m[i]) nrerror("allocation failure
2 in dmatrix()");

                    m[i] -= ncl;

          }

          return m;

}     /* dmatrix */


/*************************************************************
************/


void free_dmatrix(double **m,int nrl,int nrh,int ncl,int nch)


/* Free memory previously allocated by dmatrix()
*/


{

          int i;


          for(i=nrh;i>=nrl;i--) free((char*)
(m[i]+ncl));

          free((char*) (m+nrl));

}     /* free_dmatrix   */

/*************************************************************
**************************/
```

A1.2 map_ten7fsr.c

```
/* This program is written to calculate the ADC and anisotropy
images  */
/* using multivariate linear regression to calculate the
diffusion tensor */
/* 1. calculates for any usual combination of b values and
FOV's. */
/* 2. calculates for a choice of sampling strategies eg
uniform & conventional*/
/* 3. Allows a non-zero baseline b-value to be used */
/* 4. Allows different nex's on baseline and dw acquisitions
*/
/* 5. Allows multiple b_values to be used */
/* 6. Smooths images to remove noise from calculated images */
/* 7. Reads in noise data from scale_ten7, allowing cropping
before allignment*/
/* 8. Allows poor images to be rejected from calcualted images
in 'map' */

/* Originally written: 11th May 1998 */
/* Major revision: 30th March 1999 */
/* Last updated: 18th May 1999 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define NRANSI
#define NR_END 1
#define FREE_ARG char*
#define TINY 1.0e-20;
#define ROTATE(a,i,j,k,l) g=a[i][j];h=a[k][l];a[i][j]=g-
s*(h+g*tau);\
          a[k][l]=h+s*(g-h*tau);

float **imaxx, **imayy, **imazz, **imr, **sumr;
float **imaxy, **imaxz, **imayz, **imagebase;
float **imaget1, **imaget2, **imaget3, **imaget4, **imaget5,
**imaget6;
float **sumt1, **sumt2, **sumt3, **sumt4, **sumt5, **sumt6;
float **ave_image, **imager;
float **ave_imager, **ave_imaget1, **ave_imaget2,
**ave_imaget3;
float **ave_imaget4, **ave_imaget5, **ave_imaget6, ***ave_sum;
float **aver, ***avet1, ***avet2, ***avet3, ***avet4,
***avet5, ***avet6;
float **adc_tensor, **dtensor_image, **eig_vect, **asd_tensor,
**avr_tensor;
float **pro_tensor, **el_tensor, **e2_tensor, **e3_tensor;
float **sumt4, **sumt5, **sumt6, **fa_tensor, **aer1_tensor,
**aer2_tensor;
float **eigval1, **eigval2, **eigval3, **eigvec11, **eigvec12,
**eigvec13;
float **eigvec21, **eigvec22, **eigvec23, **eigvec31,
**eigvec32, **eigvec33;
float  *eig_val;
float ***t1_bad, ***t2_bad, ***t3_bad, ***t4_bad, ***t5_bad,
***t6_bad;
float t1_bad_tmp, t2_bad_tmp, t3_bad_tmp, t4_bad_tmp,
t5_bad_tmp, t6_bad_tmp;
float **count_bad_t1, **count_bad_t2, **count_bad_t3,
**count_bad_t4;
float **count_bad_t5, **count_bad_t6;
float scale_adc=5000000.0, scale_asd=10000.0;
FILE   *data_filet1, *data_filet2, *data_filet3, *data_filer,
*data_filet4;
FILE   *data_filet5, *data_filet6;
FILE   *out_filer, *out_filet1, *out_filet2, *out_file_adc,
*out_file_pro;
FILE   *out_filet3, *out_filet4, *out_filet5, *out_filet6,
*out_file_adc;
FILE   *out_file_adc_ten, *out_file_asd_ten,
*out_file_avr_ten;
FILE   *out_file_pro_ten, *out_file_el_ten, *out_file_e2_ten;
FILE   *out_file_e3_ten, *out_file_e4_ten, *out_fileb6,
*out_file_base_ten;
FILE   *out_file_dxx_ten, *out_file_dyy_ten,
*out_file_dzz_ten;
FILE   *out_file_dxy_ten, *out_file_dxz_ten,
*out_file_dyz_ten;
FILE   *out_file_fa_ten, *out_file_aer1_ten,
*out_file_aer2_ten;
FILE   *out_file_ave_sum;
FILE   *mat_file_eval1, *mat_file_eval2, *mat_file_eval3;
FILE   *mat_file_evec11, *mat_file_evec12, *mat_file_evec13;
FILE   *mat_file_evec21, *mat_file_evec22, *mat_file_evec23;
FILE   *mat_file_evec31, *mat_file_evec32, *mat_file_evec33;
FILE   *mat_file_t2, *out_file_base;
char   input_type[3], st_name[255], dir_make[255];
char   image_no[20], sl[10], acq[10], bas[10];
char   slash[]="/", map[]="map", sc[]="scale",
r[]="r",base[]="base_ten";
char   hdr[]=".hdr", dot[]="do", dxx[]="dxx_ten",
dyy[]="dyy_ten";
char   dxy[]="dxy_ten", dxz[]="dxz_ten", dyz[]="dyz_ten",
dzz[]="dzz_ten";
char   sot[]="so", spxx[]="spxx_", spyy[]="spyy_",
spzz[]="spzz_";
char   spxy[]="spxy_", spxz[]="spxz_", spyz[]="spyz_";
char   snxy[]="snxy_", snxz[]="snxz_", snyz[]="snyz_";
char   outpxx[255], outpyy[255], outpzz[255];
char   outpxy[255], outpyz[255], outpyz[255];
char   outnxy[255], outnxz[255], outnyz[255], outave[255];
char   adc_ten[]="adc_ten", asd_ten[]="asd_ten",
avr_ten[]="avr_ten";
char   pro_ten[]="pro_ten", el_ten[]="el_ten",
e2_ten[]="e2_ten";
char   e3_ten[]="e3_ten", fdr[]="sl", mp[]="map",
mt[]="matlab";
char   t2[]="t2", d2[]="adc", fa_ten[]="fa_ten",
aer1_ten[]="aer1_ten";
char   im_fdr[255], aer2_ten[]="aer2_ten";
char   eval1[]="eval1", eval2[]="eval2", eval3[]="eval3",
evec11[]="evec11";
char   evec12[]="evec12", evec13[]="evec13",
evec21[]="evec21";
char   evec22[]="evec22", evec23[]="evec23",
evec31[]="evec31";
char   evec32[]="evec32", evec33[]="evec33";
int    slices, n_slices, in_type, b_val_temp, fov, b_type,
reg, save_choice;
int    no_slices, fov_choice, n_rows, n_cols, mat_size, aq,
n_aq_temp, row, col;
int    samp_strat, b_bas, slew, bs, n_bs, prot_type, no_b, nb,
bn, b_max;
int   *b_val, *n_aq, dep, rej_choice, n_aq_max;
int    i_b1, i_r, i_fdr,k, bad;
int        bad_count_t1, bad_count_t2, bad_count_t3,
bad_count_t4, bad_count_t5;
int        bad_count_t6;
char   s[]="S", score[]="_", slice[10], ave[]="ave",
img[]=".img", re[]="";
char   in_path[255] = "/image1/mri/dwi/";

/* header parameters */
char  hdr_name[255], type[]="SHORT";
short int  nrw, ncw, nsl, nv;
float   max_val_adc_ten, min_val_adc_ten, max_val_asd_ten,
min_val_asd_ten;
float   max_val_avr_ten, min_val_avr_ten, max_val_pro_ten,
min_val_pro_ten;
float   max_val_dxx_ten, min_val_dxx_ten, max_val_dyy_ten,
min_val_dyy_ten;
float   max_val_dzz_ten, min_val_dzz_ten, max_val_dxy_ten,
min_val_dxy_ten;
float   max_val_dxz_ten, min_val_dxz_ten, max_val_dyz_ten,
min_val_dyz_ten;
```

```
float   max_val_e1_ten, min_val_e1_ten, max_val_e2_ten,
min_val_e2_ten;
float   max_val_e3_ten, min_val_e3_ten, max_val_base_ten,
min_val_base_ten;
float   max_val_fa_ten, min_val_fa_ten, min_val_aer2_ten;
float       max_val_aer1_ten, min_val_aer1_ten,
max_val_aer2_ten;
int   max_val_r, max_val_t1, max_val_t2, max_val_t3, min_val;
int   max_val_t4, max_val_t5, max_val_t6;
int   max_val_ave_sum;
float x_dim, y_dim, z_dim;

int    read_image_file(float **rdata, char *fname, FILE
*data_file, int slices, int aq);
int    read_cropped_image_file(float **rdata, char *fname,
FILE *data_file, int slices, int aq);
int    read_scale_file(float *scale, char *fname, FILE
*scale_file);
int   save_f3_file(char *fname, FILE *out_file, float
***calc);
int    read_int(FILE *data_file);
void   save_int(int value, FILE *outfile);
long   read_long(FILE *data_file);
float sum_dw_images(void);
float ave_images(void);
float tensor_images(void);
float multivariate(int samp_strat);
float covariance(float **cov);
int    save_image_file(char *fname, FILE *out_file, float
**calc);
int   save_cropped_image_file(char *fname, FILE *out_file,
float **calc);
int   save_cropped_f3_file(char *fname, FILE *out_file, float
***calc);
int   save_matlab_file(char *fname, FILE *mat_file, float
**data);
int   save_cropped_matlab_file(char *fname, FILE *mat_file,
float **data);
void   make_hdr(char *hdr_name, short int nrw, short int ncw,
short int nsl, short int nv, char *type, int max_val, int
min_val, float x_dim, float y_dim, float z_dim);
float   inverse(float **matrix, float **inverse, int N);
float   transpose(float **matrix, float **transpose, int r,
int c);
float   multmat(float **a, float **b, float **c, int numrows1,
int numcols1, int numcols2);
float   multmatvec(float **a, float *b, float *c, int
nummrows, int nummcols);
void  make_dir(char *dir_name);

/* functions taken from Numerical Recipes     */

void   nrerror(char error_text[]);
int    **imatrix(int nrl,int nrh,int ncl,int nch);
void   free_imatrix(int **m,int nrl,int nrh,int ncl,int nch);
double **dmatrix(int nrl, int nrh, int ncl, int nch);
void   free_dmatrix(double **m, int nrl, int nrh, int ncl, int
nch);
void  jacobi(float **a, int n, float d[], float **v);
void  eigsrt(float d[], float **v, int n);
float *vector(int nl, int nh);
void  free_vector(float *v, int nl, int nh);
void  lubksb(float **a, int n, int *indx, float b[]);
void  ludcmp(float **a, int n, int *indx, float *d);
int   *ivector(int nl, int nh);
void  free_ivector(int *v, int nl, int nh);
float ***f3tensor(int nrl, int nrh, int ncl, int nch, int ndl,
int ndh);
void  free_f3tensor(float ***t, int nrl, int nrh, int ncl, int
nch, int ndl, int ndh);
float **matrix(int nrl, int nrh, int ncl, int nch);
void  free_matrix(float **m, int nrl, int nrh, int ncl, int
nch);
void moment(float data[], int n, float *ave, float *adev,
float *sdev, float *var, float *skew, float *curt);

/*************************************************************
**********/

main()
{

            /* Main Menu */

            printf("\n ***ELSCINT V.3 DIFFUSION TENSOR
PROTOCOL MENU***\n\n");

            printf("\n Calculates adc and anisotropy maps
using the Multivariate");
            printf("\n diffusion tensor analysis\n\n");

            printf("\n Please enter the following
information:\n");

            printf("\n The study name: ");
            gets(st_name);

            printf("\n The protocol used: \n");
            printf(" 1 - Conventional (b=0   & 700 smm-2)\n");
            printf(" 2 - Uniform      (b=0   & 700 smm-2)\n");
            printf(" 3 - Uniform      (b=0   & 900 smm-2)\n");
            printf(" 4 - Uniform      (b=100 & 1000 smm-
2)\n");
            printf(" 5 - Other non-standard protocol\n");
            scanf("%i", &prot_type);

            /* Define b_val and acquisition vectors */


            if(prot_type == 1)  {
            samp_strat = 1;
            no_b = 2;
            n_aq = ivector(1,no_b);
            b_val = ivector(1,no_b);
            n_aq[1] = 13;
            n_aq[2] = 3;
            b_val[1] = 0;
            b_val[2] = 700;
            slew = 1;
            reg = 2;
            n_slices = 10;
            mat_size = 4;
            fov_choice = 4;
            save_choice = 1;
            }
            if(prot_type == 2)  {
            samp_strat = 2;
            no_b = 2;
            n_aq = ivector(1,no_b);
            b_val = ivector(1,no_b);
            n_aq[1] = 13;
            n_aq[2] = 3;
            b_val[1] = 0;
            b_val[2] = 700;
            slew = 1;
            reg = 2;
            n_slices = 10;
            mat_size = 4;
            fov_choice = 4;
            save_choice = 1;
            }
            if(prot_type == 3)  {
            samp_strat = 2;
            no_b = 2;
            n_aq = ivector(1,no_b);
            b_val = ivector(1,no_b);
            n_aq[1] = 7;
            n_aq[2] = 2;
            b_val[1] = 0;
            b_val[2] = 900;
            slew = 1;
            reg = 2;
            n_slices = 10;
            mat_size = 4;
            fov_choice = 4;
            save_choice = 1;
            }
            if(prot_type == 4)  {
            samp_strat = 2;
            no_b = 2;
            n_aq = ivector(1,no_b);
            b_val = ivector(1,no_b);
            n_aq[1] = 2;
            n_aq[2] = 4;
            b_val[1] = 100;
            b_val[2] = 1000;
            slew = 1;
            reg = 2;
            n_slices = 10;
            mat_size = 1;
            fov_choice = 4;
            save_choice = 1;
            }

            if(prot_type == 5)  {
            printf("\n The Sampling strategy: \n");
            printf(" 1 - Conventional: {xx,  yy, zz,  xy, xz,
yz}\n");
            printf(" 2 - Uniform:      {xz, -xz, yz, -yz, xy,
-xy}\n");
            scanf("%i",&samp_strat);
            }

            if(prot_type == 5) {

            printf("\n The number of b-values sampled: ");
            scanf("%i",&no_b);

            n_aq = ivector(1,no_b);
            b_val = ivector(1,no_b);

            for(nb=1; nb<=no_b; nb++){

            b_val_temp = 0;

            printf("\n The nominal scalar b%i-value used
(lowest-highest): ", nb);

            scanf("%i",&b_val_temp);

            b_val[nb] = b_val_temp;

            printf("\n The number of acquisitions at b = %i
smm-2:  ",b_val[nb]);
            scanf("%i", &n_aq_temp);

            if(n_aq_temp > n_aq_max) {
            n_aq_max = n_aq_temp;
            }

            n_aq[nb] = n_aq_temp;

            }

            fflush(stdin);

            if(b_val[1] == 0){
            printf("\n The reference (T2/D0) image number: ");
            fflush(stdin);
            scanf("%i",&i_r);
            }
            else
            if(samp_strat == 1){
            printf("\n The first image number sampled in xx:
");
            scanf("%i",&i_b1); }
            else
            if(samp_strat == 2){
            printf("\n The first image number sampled in xz:
");
            scanf("%i",&i_b1); }

/*          if(samp_strat == 2){
            if(b_bas == 0){
            printf("\n The diffusion gradient slew rate: \n");
            printf(" 1 - Fast\n");
            printf(" 2 - Slow\n");
            scanf("%i", &slew);  } }
            else
            if(samp_strat == 1) {
            slew = 1;  }                          */
            slew = 1;   /* Assume fast slew rate */

            printf("\n The registration method: \n");
            printf(" 1 - No registration\n");
            printf(" 2 - SPM95 allignment\n");
            scanf("%i", &reg);

            printf("\n The number of slices: ");
            scanf("%i", &n_slices);

            printf("\n The matrix size : \n");
            printf(" 1 - 128x128\n");
```

```
            printf(" 2 - 256x128\n");
            printf(" 3 - 256x256\n");
            printf(" 4 - 256x128 cropped to 128x128\n");

            scanf("%i", &mat_size);

/*          if(slew == 1){
            printf("\n The field of view:\n");
            printf(" 1 - 380 x 190 mm\n");
            printf(" 2 - 420 x 210 mm\n");
            printf(" 3 - 440 x 220 mm\n");
            printf(" 4 - 460 x 230 mm\n");

            scanf("%i", &fov_choice);  }        */

            /* For slow slew rate assume FOV of 460x220 */

/*          else
            if(slew == 2){
            fov_choice = 4;  }      */

            fov_choice = 4;                     /* Assume FOV
= 460 x 230 */

            printf("\n Do you want to save matlab files (1-
yes, 0-no) ?\n");
            scanf("%i", &save_choice);

            }

            if(b_val[1] == 0) {
            i_fdr = i_r+1;
            b_max = no_b-1;
            }
            else  {
            i_fdr = i_b1;
            b_max = no_b;
            }

            fflush(stdin);

            if(prot_type != 5){

            if(b_val[1] == 0){
            printf("\n The reference (T2/D0) image number: ");
            fflush(stdin);
            scanf("%i",&i_r);
            }
            else
            if(samp_strat == 1){
            printf("\n The first image number sampled in xx:
");
            scanf("%i",&i_b1); }
            else
            if(samp_strat == 2){
            printf("\n The first image number sampled in xz:
");
            scanf("%i",&i_b1); }
            }

            /* Define tensors top store bad data locations */

            t1_bad=f3tensor(1,n_slices,1,b_max,1,n_aq_max);
            t2_bad=f3tensor(1,n_slices,1,b_max,1,n_aq_max);
            t3_bad=f3tensor(1,n_slices,1,b_max,1,n_aq_max);
            t4_bad=f3tensor(1,n_slices,1,b_max,1,n_aq_max);
            t5_bad=f3tensor(1,n_slices,1,b_max,1,n_aq_max);
            t6_bad=f3tensor(1,n_slices,1,b_max,1,n_aq_max);
            count_bad_t1=matrix(1,n_slices,1,b_max);
            count_bad_t2=matrix(1,n_slices,1,b_max);
            count_bad_t3=matrix(1,n_slices,1,b_max);
            count_bad_t4=matrix(1,n_slices,1,b_max);
            count_bad_t5=matrix(1,n_slices,1,b_max);
            count_bad_t6=matrix(1,n_slices,1,b_max);

            /* Set bad data tensors top default */

            for(slices = 1; slices <= n_slices; slices++) {
              for(nb = 1; nb <= b_max; nb++) {
                for(dep = 1; dep <= n_aq_max; dep++) {
            t1_bad[slices][nb][bad] = -1;
            t2_bad[slices][nb][bad] = -1;
            t3_bad[slices][nb][bad] = -1;
            t4_bad[slices][nb][bad] = -1;
            t5_bad[slices][nb][bad] = -1;
            t6_bad[slices][nb][bad] = -1;
                }
              }
            }

            printf("\n Do you want to reject any poor quality
images (1-yes, 0-no) ? \n");
            scanf("%i", &rej_choice);

            if(rej_choice == 1) {

            for(slices = 1; slices <= n_slices; slices++) {
            printf("\n Enter the numbers of  the poor images
for slice %d",slices);
            printf("\n (-1) to escape");

            for(nb = 1; nb<=b_max; nb++) {

            printf("\n in b = %d smm-2, PXZ image:
",b_val[nb]);

            bad = 1;
            count_bad_t1[slices][nb]=0;
            scanf("%f", &t1_bad_tmp);
            t1_bad[slices][nb][bad] = t1_bad_tmp;

            while(t1_bad[slices][nb][bad] != -1) {
            count_bad_t1[slices][nb]++;
            scanf("%f", &t1_bad_tmp);
            t1_bad[slices][nb][bad+1] = t1_bad_tmp;
            bad++;
            }

            printf("\n in b = %d smm-2, NXZ image:
",b_val[nb]);

            bad = 1;
            count_bad_t2[slices][nb]=0;
            scanf("%f", &t2_bad_tmp);
            t2_bad[slices][nb][bad] = t2_bad_tmp;

            while(t2_bad[slices][nb][bad] != -1) {
            count_bad_t2[slices][nb]++;
            scanf("%f", &t2_bad_tmp);
            t2_bad[slices][nb][bad+1] = t2_bad_tmp;
            bad++;
            }

            printf("\n in b = %d smm-2, PYZ image:
",b_val[nb]);

            bad = 1;
            count_bad_t3[slices][nb]=0;
            scanf("%f", &t3_bad_tmp);
            t3_bad[slices][nb][bad] = t3_bad_tmp;

            while(t3_bad[slices][nb][bad] != -1) {
            count_bad_t3[slices][nb]++;
            scanf("%f", &t3_bad_tmp);
            t3_bad[slices][nb][bad+1] = t3_bad_tmp;
            bad++;
            }

            printf("\n in b = %d smm-2, NYZ image:
",b_val[nb]);

            bad = 1;
            count_bad_t4[slices][nb]=0;
            scanf("%f", &t4_bad_tmp);
            t4_bad[slices][nb][bad] = t4_bad_tmp;

            while(t4_bad[slices][nb][bad] != -1) {
            count_bad_t4[slices][nb]++;
            scanf("%f", &t4_bad_tmp);
            t4_bad[slices][nb][bad+1] = t4_bad_tmp;
            bad++;
            }

            printf("\n in b = %d smm-2, PXY image:
",b_val[nb]);

            bad = 1;
            count_bad_t5[slices][nb]=0;
            scanf("%f", &t5_bad_tmp);
            t5_bad[slices][nb][bad] = t5_bad_tmp;

            while(t5_bad[slices][nb][bad] != -1) {
            count_bad_t5[slices][nb]++;
            scanf("%f", &t5_bad_tmp);
            t5_bad[slices][nb][bad+1] = t5_bad_tmp;
            bad++;
            }

            printf("\n in b = %d smm-2, NXY image:
",b_val[nb]);

            bad = 1;
            count_bad_t6[slices][nb]=0;
            scanf("%f", &t6_bad_tmp);
            t6_bad[slices][nb][bad] = t6_bad_tmp;

            while(t6_bad[slices][nb][bad] != -1) {
            count_bad_t6[slices][nb]++;
            scanf("%f", &t6_bad_tmp);
            t6_bad[slices][nb][bad+1] = t6_bad_tmp;
            bad++;
            }
            }          /* end b_values nb loop */
            }          /* end slices loop */
            }          /* end rej choice == 1 */

            /* Set reg prefix to r if images are registered
using spm */

            if(reg == 2)
            sprintf(re,r);

            nsl=(short int)n_slices;
            nv=1;

            if(mat_size == 1){
            nrw=128;
            ncw=128;
            n_rows=128;
            n_cols=128; }
            else
            if(mat_size == 2){
            nrw=256;
            ncw=128;
            n_rows=128;
            n_cols=256; }
            else
            if(mat_size == 3){
            nrw=256;
            ncw=256;
            n_rows=256;
            n_cols=256; }
            else
            if(mat_size == 4){
            nrw=128;
            ncw=128;
            n_rows=128;
            n_cols=256; }

            z_dim = 6.0;            /* Default slice thickness
= 6mm */
            if(fov_choice == 1){
            x_dim = 1.484;
            y_dim = 1.484; }
            if(fov_choice == 2){
            x_dim = 1.641;
            y_dim = 1.641; }
            if(fov_choice == 3){
            x_dim = 1.719;
            y_dim = 1.719; }
            if(fov_choice == 4){
            x_dim = 1.797;
            y_dim = 1.797; }

            /* Print protocol summary to screen */

            printf("\n Summary of chosen protocol:\n\n");

            if(samp_strat == 1){
            printf(" Conventional Sampling\n");}
            else
            if(samp_strat == 2){
            printf(" Uniform Sampling \n");}
            else
```

```
            printf(" ***ERROR IN CHOSEN SAMPLING
STRATEGY***\n");

            printf(" b_values obtained at: \n");
            for(nb=1; nb<=no_b; nb++){
            printf(" %i smm-2 with %i
acquisitions\n",b_val[nb],n_aq[nb]);
            }

            if(slew == 1){
            printf(" Fast gradient slew rate\n");}
            else
            if(slew == 2){
            printf(" Slow gradient slew rate\n");}
            else
            printf(" ***ERROR IN CHOSEN GRADIENT SLEW
RATE***\n");

            if(reg == 1){
            printf(" No image alignment used\n");}
            else
            if(reg == 2){
            printf(" Images aligned with SPM95\n");}
            else
            printf(" ***ERROR IN IMAGE ALIGNMENT
CHOICE***\n");

            printf(" %i slices obtained\n",n_slices);

            if(mat_size == 1){
            printf(" 128 x 128 matrix with");}
            else
            if(mat_size == 2){
            printf(" 256 x 128 matrix with");}
            else
            if(mat_size == 3){
            printf(" 256 x 256 matrix with");}
            else
            if(mat_size == 4){
            printf(" 256 x 128 matrix cropped to 128 x 128
with");}
            else
            printf(" ***ERROR IN CHOSEN MATRIX SIZE***");

            if(fov_choice == 1){
            printf(" a FOV of 380 x 190mm\n");}
            else
            if(fov_choice == 2){
            printf(" a FOV of 420 x 210mm\n");}
            else
            if(fov_choice == 3){
            printf(" a FOV of 440 x 220mm\n");}
            else
            if(fov_choice == 4){
            printf(" a FOV of 460 x 230mm\n");}
            else
            printf(" ***ERROR IN CHOSEN FIELD OF VIEW
SIZE***\n");

            if(save_choice == 1){
            printf(" Matlab data files for ellipsoid images
saved");}
            else
            if(save_choice == 0){
            printf(" Matlab data files for ellipsoid images
not saved");}
            else
            printf(" ***ERROR IN MATLAB FILE SAVE
CHOICE***\n");

            printf("\n\n ***TYPE CONTROL-C TO EXIT DATA
PROCESSING***\n\n");

            if(b_val[1] == 0) {
            i_fdr = i_r+1;
            b_max = no_b-1;
            }
            else  {
            i_fdr = i_b1;
            b_max = no_b;
            }

            sprintf(im_fdr,"%i",i_fdr);

            /* Create folders 'matlab' and 'map' for storing
data */

            make_dir(mp);
            if(save_choice == 1) {
            make_dir(mt); }

            /* Set max and min initial values */


            max_val_adc_ten = 0, max_val_asd_ten = 0,
max_val_avr_ten = 0;
            max_val_pro_ten = 0, max_val_base_ten = 0;
            max_val_dxx_ten = 0, max_val_dyy_ten = 0,
max_val_dzz_ten = 0;
            max_val_dxy_ten = 0, max_val_dxz_ten = 0,
max_val_dyz_ten = 0;
            max_val_e1_ten = 0, max_val_e2_ten = 0,
max_val_e3_ten = 0;
            max_val_fa_ten = 0, min_val_fa_ten = 32767;
            max_val_aer1_ten = 0, max_val_aer2_ten = 0;
            min_val_adc_ten = 32767, min_val_asd_ten = 32767;
            min_val_avr_ten = 32767, min_val_pro_ten = 32767;
            min_val_dxx_ten = 32767, min_val_dyy_ten = 32767;
            min_val_dzz_ten = 32767, min_val_dxy_ten = 32767;
            min_val_dxz_ten = 32767, min_val_dyz_ten = 32767;
            min_val_e1_ten = 32767, min_val_e2_ten = 32767;
            min_val_e3_ten = 32767, min_val_base_ten = 32767;
            min_val_aer1_ten = 32767, min_val_aer2_ten =
32767;

            /* Loop to repeat for n slices incrementing the
file nos each time */

            for (slices=1; slices<=n_slices; slices++)
            {

            /* Reset strings */

            char slice[10]="", sl[10]="", acq[10]="";

            printf("\nPerforming Slice %d
calculations\n",slices);

            /* Define average images for multiple b-values */

            aver=matrix(1,n_rows,1,n_cols);
            avet1=f3tensor(1,b_max,1,n_rows,1,n_cols);

            avet2=f3tensor(1,b_max,1,n_rows,1,n_cols);
            avet3=f3tensor(1,b_max,1,n_rows,1,n_cols);
            avet4=f3tensor(1,b_max,1,n_rows,1,n_cols);
            avet5=f3tensor(1,b_max,1,n_rows,1,n_cols);
            avet6=f3tensor(1,b_max,1,n_rows,1,n_cols);
            ave_sum=f3tensor(1,b_max,1,n_rows,1,n_cols);

            for(dep=1;dep<=b_max;dep++) {
              for(row=1;row<=n_rows;row++){
                 for(col=1;col<=n_cols;col++){
            avet1[dep][row][col]=0.0;
            avet2[dep][row][col]=0.0;
            avet3[dep][row][col]=0.0;
            avet4[dep][row][col]=0.0;
            avet5[dep][row][col]=0.0;
            avet6[dep][row][col]=0.0;
            ave_sum[dep][row][col]=0.0;
                 }
              }
            }

            /* declare adc and anisotropy image arrays */

            imaxx=matrix(1,n_rows,1,n_cols);
            imayy=matrix(1,n_rows,1,n_cols);
            imazz=matrix(1,n_rows,1,n_cols);
            imaxy=matrix(1,n_rows,1,n_cols);
            imaxz=matrix(1,n_rows,1,n_cols);
            imayz=matrix(1,n_rows,1,n_cols);
            imagebase=matrix(1,n_rows,1,n_cols);
            ave_image=matrix(1,n_rows,1,n_cols);
            adc_tensor=matrix(1,n_rows,1,n_cols);
            asd_tensor=matrix(1,n_rows,1,n_cols);
            avr_tensor=matrix(1,n_rows,1,n_cols);
            aer1_tensor=matrix(1,n_rows,1,n_cols);
            aer2_tensor=matrix(1,n_rows,1,n_cols);
            pro_tensor=matrix(1,n_rows,1,n_cols);
            fa_tensor=matrix(1,n_rows,1,n_cols);
            e1_tensor=matrix(1,n_rows,1,n_cols);
            e2_tensor=matrix(1,n_rows,1,n_cols);
            e3_tensor=matrix(1,n_rows,1,n_cols);
            eigval1=matrix(1,n_rows,1,n_cols);
            eigval2=matrix(1,n_rows,1,n_cols);
            eigval3=matrix(1,n_rows,1,n_cols);
            eigvec11=matrix(1,n_rows,1,n_cols);
            eigvec12=matrix(1,n_rows,1,n_cols);
            eigvec13=matrix(1,n_rows,1,n_cols);
            eigvec21=matrix(1,n_rows,1,n_cols);
            eigvec22=matrix(1,n_rows,1,n_cols);
            eigvec23=matrix(1,n_rows,1,n_cols);
            eigvec31=matrix(1,n_rows,1,n_cols);
            eigvec32=matrix(1,n_rows,1,n_cols);
            eigvec33=matrix(1,n_rows,1,n_cols);

            /* Loop to repeat for multiple b values */

            for(nb=1; nb<=no_b; nb++)  {

            char bv[10]="", outpxx[255]="", outpyy[255]="",
outpzz[255]="";
            char outpxy[255]="", outpxz[255]="",
outpyz[255]="";
            char outnxy[255]="", outnxz[255]="",
outnyz[255]="";
            max_val_r = 0;
            max_val_t1 = 0, max_val_t2 = 0, max_val_t3 = 0;
            max_val_t4 = 0, max_val_t5 = 0, max_val_t6 = 0;
            max_val_ave_sum = 0;
            bad_count_t1 = 0, bad_count_t2 = 0, bad_count_t3 =
0;
            bad_count_t4 = 0, bad_count_t5 = 0, bad_count_t6 =
0;

            sprintf(bv,"%i",b_val[nb]);

            strcpy(outpxx,spxx);
            strcat(outpxx,bv);
            strcpy(outpyy,spyy);
            strcat(outpyy,bv);
            strcpy(outpzz,spzz);
            strcat(outpzz,bv);
            strcpy(outpxy,spxy);
            strcat(outpxy,bv);
            strcpy(outpxz,spxz);
            strcat(outpxz,bv);
            strcpy(outpyz,spyz);
            strcat(outpyz,bv);
            strcpy(outnxy,snxy);
            strcat(outnxy,bv);
            strcpy(outnxz,snxz);
            strcat(outnxz,bv);
            strcpy(outnyz,snyz);
            strcat(outnyz,bv);
            strcpy(outave,s);
            strcat(outave,score);
            strcat(outave,ave);
            strcat(outave,bv);

            /* Define image matrices for image summing and
averaging */

            sumr=matrix(1,n_rows,1,n_cols);
            sumt1=matrix(1,n_rows,1,n_cols);
            sumt2=matrix(1,n_rows,1,n_cols);
            sumt3=matrix(1,n_rows,1,n_cols);
            sumt4=matrix(1,n_rows,1,n_cols);
            sumt5=matrix(1,n_rows,1,n_cols);
            sumt6=matrix(1,n_rows,1,n_cols);

            /* Initialise image arrays for each b_value*/

            for(row=1; row<=n_rows; row++){
              for(col=1; col<=n_cols; col++){
            sumr[row][col]=0.0;
            sumt1[row][col]=0.0;
            sumt2[row][col]=0.0;
            sumt3[row][col]=0.0;
            sumt4[row][col]=0.0;
            sumt5[row][col]=0.0;
            sumt6[row][col]=0.0;
              }
```

```
            }

            printf("\nReading and scaling files with b = %i
smm-2\n",b_val[nb]);


            /* Start loop for multiple acquisitions */

            for(bs=1; bs<=n_aq[nb]; bs++){

            /* Reset strings */

            char  sl[10]="", bas[10]="";

            /* Define image matrices to be read in */

            imager=matrix(1,n_rows,1,n_cols);
            imaget1=matrix(1,n_rows,1,n_cols);
            imaget2=matrix(1,n_rows,1,n_cols);
            imaget3=matrix(1,n_rows,1,n_cols);
            imaget4=matrix(1,n_rows,1,n_cols);
            imaget5=matrix(1,n_rows,1,n_cols);
            imaget6=matrix(1,n_rows,1,n_cols);


            /* Initialise image matrices for each acquisition
*/

            for(row=1; row<=n_rows; row++){
              for(col=1; col<=n_cols; col++){
            imager[row][col]=0.0;
            imaget1[row][col]=0.0;
            imaget2[row][col]=0.0;
            imaget3[row][col]=0.0;
            imaget4[row][col]=0.0;
            imaget5[row][col]=0.0;
            imaget6[row][col]=0.0;
              }
            }

            /* Read in image files for averaging */

            if(samp_strat == 1){

            if(b_val[nb] == 0){

            if (read_image_file(imager, sot, data_filer,
slices, bs)!=0)
            {
              printf("Error reading data file\n");
              return(1);
            }
            }
            else {
            if (read_image_file(imaget1, outpxx, data_filet1,
slices, bs)!=0)
            {
              printf("Error reading data file\n");
              return(1);
            }
            if (read_image_file(imaget2, outpyy, data_filet2,
slices, bs)!=0)
            {
              printf("Error reading data file\n");
              return(1);
            }
            if (read_image_file(imaget3, outpzz, data_filet3,
slices, bs)!=0)
            {
              printf("Error reading data file\n");
              return(1);
            }
            if (read_image_file(imaget4, outpxy, data_filet4,
slices, bs)!=0)
            {
              printf("Error reading data file\n");
              return(1);
            }
            if (read_image_file(imaget5, outpxz, data_filet5,
slices, bs)!=0)
            {
              printf("Error reading data file\n");
              return(1);
            }
            if (read_image_file(imaget6, outpyz, data_filet6,
slices, bs)!=0)
            {
              printf("Error reading data file\n");
              return(1);
            }
            }
            }
            else
            if(samp_strat == 2){

            if(b_val[nb] == 0){
            if (read_image_file(imager, sot, data_filer,
slices, bs)!=0)
            {
              printf("Error reading data file\n");
              return(1);
            }
            }
            else {
            if (read_image_file(imaget1, outpxz, data_filet1,
slices, bs)!=0)
            {
              printf("Error reading data file\n");
              return(1);
            }
            if (read_image_file(imaget2, outnxz, data_filet2,
slices, bs)!=0)
            {
              printf("Error reading data file\n");
              return(1);
            }
            if (read_image_file(imaget3, outpyz, data_filet3,
slices, bs)!=0)
            {
              printf("Error reading data file\n");
              return(1);
            }
            if (read_image_file(imaget4, outnyz, data_filet4,
slices, bs)!=0)
            {
              printf("Error reading data file\n");
              return(1);
            }
            if (read_image_file(imaget5, outpxy, data_filet5,
slices, bs)!=0)
            {
              printf("Error reading data file\n");
              return(1);
            }
            if (read_image_file(imaget6, outnxy, data_filet6,
slices, bs)!=0)
            {
              printf("Error reading data file\n");
              return(1);
            }
            }
            }

            /* Sum multiple images from multiple acquisitions
*/
            /* But do not sum error images */

            sum_dw_images();

            /* Free image matrix memory before next
acquisition */

            free_matrix(imager,1,n_rows,1,n_cols);
            free_matrix(imaget1,1,n_rows,1,n_cols);
            free_matrix(imaget2,1,n_rows,1,n_cols);
            free_matrix(imaget3,1,n_rows,1,n_cols);
            free_matrix(imaget4,1,n_rows,1,n_cols);
            free_matrix(imaget5,1,n_rows,1,n_cols);
            free_matrix(imaget6,1,n_rows,1,n_cols);
            }           /* End loop for multiple acquisitions
*/

            /* Average images for each slice */

            ave_images();

            /* Make header files for averaged images */

            min_val = 0;

            if(mat_size == 4) {

            if(samp_strat == 1){

            if(b_val[nb] ==0){
            make_hdr(sot, nrw, ncw, nsl, nv, type, max_val_r,
min_val, x_dim, y_dim, z_dim);
            }
            else {
            make_hdr(outpxx, nrw, ncw, nsl, nv, type,
max_val_t1, min_val, x_dim, y_dim, z_dim);
            make_hdr(outpyy, nrw, ncw, nsl, nv, type,
max_val_t2, min_val, x_dim, y_dim, z_dim);
            make_hdr(outpzz, nrw, ncw, nsl, nv, type,
max_val_t3, min_val, x_dim, y_dim, z_dim);
            make_hdr(outpxy, nrw, ncw, nsl, nv, type,
max_val_t4, min_val, x_dim, y_dim, z_dim);
            make_hdr(outpxz, nrw, ncw, nsl, nv, type,
max_val_t5, min_val, x_dim, y_dim, z_dim);
            make_hdr(outpyz, nrw, ncw, nsl, nv, type,
max_val_t6, min_val, x_dim, y_dim, z_dim);
            make_hdr(outave, nrw, ncw, nsl, nv, type,
max_val_ave_sum, min_val, x_dim, y_dim, z_dim);
            }

            /* Save averaged images */

            printf("\nSaving averaged image files.....\n");

            if(b_val[nb] == 0){

            if (save_cropped_image_file(sot, out_filer,
aver)!=0)
            {
              printf("Error saving data\n");
              return(1);
            }
            }
            else {
            if (save_cropped_f3_file(outpxx, out_filet1,
avet1)!=0)
            {
              printf("Error saving data\n");
              return(1);
            }

            if (save_cropped_f3_file(outpyy, out_filet2,
avet2)!=0)
            {
              printf("Error saving data\n");
              return(1);
            }

            if (save_cropped_f3_file(outpzz, out_filet3,
avet3)!=0)
            {
              printf("Error saving data\n");
              return(1);
            }
            if (save_cropped_f3_file(outpxy, out_filet4,
avet4)!=0)
            {
              printf("Error saving data\n");
              return(1);
            }

            if (save_cropped_f3_file(outpxz, out_filet5,
avet5)!=0)
            {
              printf("Error saving data\n");
              return(1);
            }

            if (save_cropped_f3_file(outpyz, out_filet6,
avet6)!=0)
            {
              printf("Error saving data\n");
              return(1);
            }

            if (save_cropped_f3_file(outave, out_file_ave_sum,
ave_sum)!=0)
            {
              printf("Error saving data\n");
```

```
            return(1);
          }
          }
          }
          else
          if(samp_strat == 2){

          if(b_val[nb] == 0){
          make_hdr(sot, nrw, ncw, nsl, nv, type, max_val_r,
min_val, x_dim, y_dim, z_dim);
          }
          else {
          make_hdr(outpxz, nrw, ncw, nsl, nv, type,
max_val_t1, min_val, x_dim, y_dim, z_dim);
          make_hdr(outnxz, nrw, ncw, nsl, nv, type,
max_val_t2, min_val, x_dim, y_dim, z_dim);
          make_hdr(outpyz, nrw, ncw, nsl, nv, type,
max_val_t3, min_val, x_dim, y_dim, z_dim);
          make_hdr(outnyz, nrw, ncw, nsl, nv, type,
max_val_t4, min_val, x_dim, y_dim, z_dim);
          make_hdr(outpxy, nrw, ncw, nsl, nv, type,
max_val_t5, min_val, x_dim, y_dim, z_dim);
          make_hdr(outnxy, nrw, ncw, nsl, nv, type,
max_val_t6, min_val, x_dim, y_dim, z_dim);
          make_hdr(outave, nrw, ncw, nsl, nv, type,
max_val_ave_sum, min_val, x_dim, y_dim, z_dim);
          }

          /* Save averaged orthogonal and tetrahedral images
*/

          printf("\nSaving averaged image files.....\n");

          if(b_val[nb] == 0){

          if (save_cropped_image_file(sot, out_filer,
aver)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }
          }
          else{
          if (save_cropped_f3_file(outpxz, out_filet1,
avet1)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_cropped_f3_file(outnxz, out_filet2,
avet2)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_cropped_f3_file(outpyz, out_filet3,
avet3)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }
          if (save_cropped_f3_file(outnyz, out_filet4,
avet4)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_cropped_f3_file(outpxy, out_filet5,
avet5)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_cropped_f3_file(outnxy, out_filet6,
avet6)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_cropped_f3_file(outave, out_file_ave_sum,
ave_sum)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }
          }
          }
          }
          else {
          if(samp_strat == 1){

          if(b_val[nb] ==0){
          make_hdr(sot, nrw, ncw, nsl, nv, type, max_val_r,
min_val, x_dim, y_dim, z_dim);
          }
          else {
          make_hdr(outpxx, nrw, ncw, nsl, nv, type,
max_val_t1, min_val, x_dim, y_dim, z_dim);
          make_hdr(outpyy, nrw, ncw, nsl, nv, type,
max_val_t2, min_val, x_dim, y_dim, z_dim);
          make_hdr(outpzz, nrw, ncw, nsl, nv, type,
max_val_t3, min_val, x_dim, y_dim, z_dim);
          make_hdr(outpxy, nrw, ncw, nsl, nv, type,
max_val_t4, min_val, x_dim, y_dim, z_dim);
          make_hdr(outpxz, nrw, ncw, nsl, nv, type,
max_val_t5, min_val, x_dim, y_dim, z_dim);
          make_hdr(outpyz, nrw, ncw, nsl, nv, type,
max_val_t6, min_val, x_dim, y_dim, z_dim);
          make_hdr(outave, nrw, ncw, nsl, nv, type,
max_val_ave_sum, min_val, x_dim, y_dim, z_dim);
          }

          /* Save averaged images */

          printf("\nSaving averaged image files.....\n");

          if(b_val[nb] == 0){

          if (save_image_file(sot, out_filer, aver)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }
          }
          else {
          if (save_f3_file(outpxx, out_filet1, avet1)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_f3_file(outpyy, out_filet2, avet2)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_f3_file(outpzz, out_filet3, avet3)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }
          if (save_f3_file(outpxy, out_filet4, avet4)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_f3_file(outpxz, out_filet5, avet5)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_f3_file(outpyz, out_filet6, avet6)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_f3_file(outpyz, out_file_ave_sum,
ave_sum)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }
          }
          }
          }
          else
          if(samp_strat == 2){

          if(b_val[nb] == 0){
          make_hdr(sot, nrw, ncw, nsl, nv, type, max_val_r,
min_val, x_dim, y_dim, z_dim);
          }
          else {
          make_hdr(outpxz, nrw, ncw, nsl, nv, type,
max_val_t1, min_val, x_dim, y_dim, z_dim);
          make_hdr(outnxz, nrw, ncw, nsl, nv, type,
max_val_t2, min_val, x_dim, y_dim, z_dim);
          make_hdr(outpyz, nrw, ncw, nsl, nv, type,
max_val_t3, min_val, x_dim, y_dim, z_dim);
          make_hdr(outnyz, nrw, ncw, nsl, nv, type,
max_val_t4, min_val, x_dim, y_dim, z_dim);
          make_hdr(outpxy, nrw, ncw, nsl, nv, type,
max_val_t5, min_val, x_dim, y_dim, z_dim);
          make_hdr(outnxy, nrw, ncw, nsl, nv, type,
max_val_t6, min_val, x_dim, y_dim, z_dim);
          make_hdr(outave, nrw, ncw, nsl, nv, type,
max_val_ave_sum, min_val, x_dim, y_dim, z_dim);
          }

          /* Save averaged orthogonal and tetrahedral images
*/

          printf("\nSaving averaged image files.....\n");

          if(b_val[nb] == 0){

          if (save_image_file(sot, out_filer, aver)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }
          }
          else{
          if (save_f3_file(outpxz, out_filet1, avet1)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_f3_file(outnxz, out_filet2, avet2)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_f3_file(outpyz, out_filet3, avet3)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }
          if (save_f3_file(outnyz, out_filet4, avet4)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_f3_file(outpxy, out_filet5, avet5)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_f3_file(outnxy, out_filet6, avet6)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_f3_file(outave, out_file_ave_sum,
ave_sum)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }
```

```
            }
            }
            }

            /* free memory from sum images */

            free_matrix(sumr,1,n_rows,1,n_cols);
            free_matrix(sumt1,1,n_rows,1,n_cols);
            free_matrix(sumt2,1,n_rows,1,n_cols);
            free_matrix(sumt3,1,n_rows,1,n_cols);
            free_matrix(sumt4,1,n_rows,1,n_cols);
            free_matrix(sumt5,1,n_rows,1,n_cols);
            free_matrix(sumt6,1,n_rows,1,n_cols);

            }           /* End loop for multiple b values */

            /* Calculate adc, ave and anisotropy images */

            printf("\nPerforming Multivariate
Analysis.....\n");

            multivariate(samp_strat);

            tensor_images();

            /* Make header files for adc and anisotropy images
*/

            printf("\nSaving anisotropy maps.....\n");

            if(slices == n_slices){

            make_hdr(adc_ten, nrw, ncw, nsl, nv, type,
(int)max_val_adc_ten, (int)min_val_adc_ten, x_dim, y_dim,
z_dim);
            make_hdr(asd_ten, nrw, ncw, nsl, nv, type,
(int)max_val_asd_ten, (int)min_val_asd_ten, x_dim, y_dim,
z_dim);
            make_hdr(avr_ten, nrw, ncw, nsl, nv, type,
(int)max_val_avr_ten, (int)min_val_avr_ten, x_dim, y_dim,
z_dim);
            make_hdr(pro_ten, nrw, ncw, nsl, nv, type,
(int)max_val_pro_ten, (int)min_val_pro_ten, x_dim, y_dim,
z_dim);
            make_hdr(e1_ten, nrw, ncw, nsl, nv, type,
(int)max_val_e1_ten, (int)min_val_e1_ten, x_dim, y_dim,
z_dim);
            make_hdr(e2_ten, nrw, ncw, nsl, nv, type,
(int)max_val_e2_ten, (int)min_val_e2_ten, x_dim, y_dim,
z_dim);
            make_hdr(e3_ten, nrw, ncw, nsl, nv, type,
(int)max_val_e3_ten, (int)min_val_e3_ten, x_dim, y_dim,
z_dim);
            make_hdr(dxx, nrw, ncw, nsl, nv, type,
(int)max_val_dxx_ten, (int)min_val_dxx_ten, x_dim, y_dim,
z_dim);
            make_hdr(dyy, nrw, ncw, nsl, nv, type,
(int)max_val_dyy_ten, (int)min_val_dyy_ten, x_dim, y_dim,
z_dim);
            make_hdr(dzz, nrw, ncw, nsl, nv, type,
(int)max_val_dzz_ten, (int)min_val_dzz_ten, x_dim, y_dim,
z_dim);
            make_hdr(dxy, nrw, ncw, nsl, nv, type,
(int)max_val_dxy_ten, (int)min_val_dxy_ten, x_dim, y_dim,
z_dim);
            make_hdr(dxz, nrw, ncw, nsl, nv, type,
(int)max_val_dxz_ten, (int)min_val_dxz_ten, x_dim, y_dim,
z_dim);
            make_hdr(dyz, nrw, ncw, nsl, nv, type,
(int)max_val_dyz_ten, (int)min_val_dyz_ten, x_dim, y_dim,
z_dim);
            make_hdr(base, nrw, ncw, nsl, nv, type,
(int)max_val_base_ten, (int)min_val_base_ten, x_dim, y_dim,
z_dim);
            make_hdr(fa_ten, nrw, ncw, nsl, nv, type,
(int)max_val_fa_ten, (int)min_val_fa_ten, x_dim, y_dim,
z_dim);
            make_hdr(aer1_ten, nrw, ncw, nsl, nv, type,
(int)max_val_aer1_ten, (int)min_val_aer1_ten, x_dim, y_dim,
z_dim);
            make_hdr(aer2_ten, nrw, ncw, nsl, nv, type,
(int)max_val_aer2_ten, (int)min_val_aer2_ten, x_dim, y_dim,
z_dim);
            }

            /* Save calculated adc and anisotropy image files
*/

            if(mat_size == 4) {

            if (save_cropped_image_file(adc_ten,
out_file_adc_ten, adc_tensor)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }


            if (save_cropped_image_file(asd_ten,
out_file_asd_ten, asd_tensor)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }


            if (save_cropped_image_file(avr_ten,
out_file_avr_ten, avr_tensor)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }


            if (save_cropped_image_file(pro_ten,
out_file_pro_ten, pro_tensor)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }

            if (save_cropped_image_file(e1_ten,
out_file_e1_ten, e1_tensor)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }

            if (save_cropped_image_file(e2_ten,
out_file_e2_ten, e2_tensor)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }

            if (save_cropped_image_file(e3_ten,
out_file_e3_ten, e3_tensor)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }

            if (save_cropped_image_file(dxx, out_file_dxx_ten,
imaxx)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }


            if (save_cropped_image_file(dyy, out_file_dyy_ten,
imayy)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }


            if (save_cropped_image_file(dzz, out_file_dzz_ten,
imazz)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }


            if (save_cropped_image_file(dxy, out_file_dxy_ten,
imaxy)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }

            if (save_cropped_image_file(dxz, out_file_dxz_ten,
imaxz)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }


            if (save_cropped_image_file(dyz, out_file_dyz_ten,
imayz)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }

            if (save_cropped_image_file(base,
out_file_base_ten, imagebase)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }

            if (save_cropped_image_file(fa_ten,
out_file_fa_ten, fa_tensor)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }

            if (save_cropped_image_file(aer1_ten,
out_file_aer1_ten, aer1_tensor)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }

            if (save_cropped_image_file(aer2_ten,
out_file_aer2_ten, aer2_tensor)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }
            }

            else {

            if (save_image_file(adc_ten, out_file_adc_ten,
adc_tensor)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }


            if (save_image_file(asd_ten, out_file_asd_ten,
asd_tensor)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }


            if (save_image_file(avr_ten, out_file_avr_ten,
avr_tensor)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }


            if (save_image_file(pro_ten, out_file_pro_ten,
pro_tensor)!=0)
              {
                printf("Error saving data\n");
                return(1);
              }

            if (save_image_file(e1_ten, out_file_e1_ten,
e1_tensor)!=0)
              {
                printf("Error saving data\n");
```

```
            return(1);
          }

          if (save_image_file(e2_ten, out_file_e2_ten,
e2_tensor)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_image_file(e3_ten, out_file_e3_ten,
e3_tensor)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }
          if (save_image_file(dxx, out_file_dxx_ten,
imaxx)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_image_file(dyy, out_file_dyy_ten,
imayy)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }


          if (save_image_file(dzz, out_file_dzz_ten,
imazz)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }


          if (save_image_file(dxy, out_file_dxy_ten,
imaxy)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_image_file(dxz, out_file_dxz_ten,
imaxz)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }


          if (save_image_file(dyz, out_file_dyz_ten,
imayz)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_image_file(base, out_file_base_ten,
imagebase)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_image_file(fa_ten, out_file_fa_ten,
fa_tensor)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_image_file(aer1_ten, out_file_aer1_ten,
aer1_tensor)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_image_file(aer2_ten, out_file_aer2_ten,
aer2_tensor)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }
          }

          /* Save eigenvalues and eigenvectors to matlab
files */

          if(save_choice == 1) {

          printf("\nSaving matlab files.....\n");

          if (save_matlab_file(eval1, mat_file_eval1,
e1_tensor)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_matlab_file(eval2, mat_file_eval2,
e2_tensor)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_matlab_file(eval3, mat_file_eval3,
e3_tensor)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_matlab_file(evec11, mat_file_evec11,
eigvec11)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_matlab_file(evec12, mat_file_evec12,
eigvec12)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_matlab_file(evec13, mat_file_evec13,
eigvec13)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_matlab_file(evec21, mat_file_evec21,
eigvec21)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_matlab_file(evec22, mat_file_evec22,
eigvec22)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_matlab_file(evec23, mat_file_evec23,
eigvec23)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_matlab_file(evec31, mat_file_evec31,
eigvec31)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_matlab_file(evec32, mat_file_evec32,
eigvec32)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_matlab_file(evec33, mat_file_evec33,
eigvec33)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          if (save_matlab_file(d2, mat_file_t2,
adc_tensor)!=0)
          {
            printf("Error saving data\n");
            return(1);
          }

          }

          /* free memory from adc and anisotropy image
matrices */

          free_matrix(adc_tensor,1,n_rows,1,n_cols);
          free_matrix(asd_tensor,1,n_rows,1,n_cols);
          free_matrix(avr_tensor,1,n_rows,1,n_cols);
          free_matrix(aer1_tensor,1,n_rows,1,n_cols);
          free_matrix(aer2_tensor,1,n_rows,1,n_cols);
          free_matrix(pro_tensor,1,n_rows,1,n_cols);
          free_matrix(fa_tensor,1,n_rows,1,n_cols);
          free_matrix(e1_tensor,1,n_rows,1,n_cols);
          free_matrix(e2_tensor,1,n_rows,1,n_cols);
          free_matrix(e3_tensor,1,n_rows,1,n_cols);
          free_matrix(eigval1,1,n_rows,1,n_cols);
          free_matrix(eigval2,1,n_rows,1,n_cols);
          free_matrix(eigval3,1,n_rows,1,n_cols);
          free_matrix(eigvec11,1,n_rows,1,n_cols);
          free_matrix(eigvec12,1,n_rows,1,n_cols);
          free_matrix(eigvec13,1,n_rows,1,n_cols);
          free_matrix(eigvec21,1,n_rows,1,n_cols);
          free_matrix(eigvec22,1,n_rows,1,n_cols);
          free_matrix(eigvec23,1,n_rows,1,n_cols);
          free_matrix(eigvec31,1,n_rows,1,n_cols);
          free_matrix(eigvec32,1,n_rows,1,n_cols);
          free_matrix(eigvec33,1,n_rows,1,n_cols);
          free_matrix(ave_image,1,n_rows,1,n_cols);
          free_matrix(imaxx,1,n_rows,1,n_cols);
          free_matrix(imayy,1,n_rows,1,n_cols);
          free_matrix(imazz,1,n_rows,1,n_cols);
          free_matrix(imaxy,1,n_rows,1,n_cols);
          free_matrix(imaxz,1,n_rows,1,n_cols);
          free_matrix(imayz,1,n_rows,1,n_cols);
          free_matrix(imagebase,1,n_rows,1,n_cols);

          free_f3tensor(avet1,1,b_max,1,n_rows,1,n_cols);
          free_f3tensor(avet2,1,b_max,1,n_rows,1,n_cols);
          free_f3tensor(avet3,1,b_max,1,n_rows,1,n_cols);
          free_f3tensor(avet4,1,b_max,1,n_rows,1,n_cols);
          free_f3tensor(avet5,1,b_max,1,n_rows,1,n_cols);
          free_f3tensor(avet6,1,b_max,1,n_rows,1,n_cols);
          free_matrix(aver,1,n_rows,1,n_cols);

          }

          free_f3tensor(t1_bad,1,n_slices,1,b_max,1,n_aq_max
);
          free_f3tensor(t2_bad,1,n_slices,1,b_max,1,n_aq_max
);
          free_f3tensor(t3_bad,1,n_slices,1,b_max,1,n_aq_max
);
          free_f3tensor(t4_bad,1,n_slices,1,b_max,1,n_aq_max
);
          free_f3tensor(t5_bad,1,n_slices,1,b_max,1,n_aq_max
);
          free_f3tensor(t6_bad,1,n_slices,1,b_max,1,n_aq_max
);
          free_matrix(count_bad_t1,1,n_slices,1,b_max);
          free_matrix(count_bad_t2,1,n_slices,1,b_max);
          free_matrix(count_bad_t3,1,n_slices,1,b_max);
          free_matrix(count_bad_t4,1,n_slices,1,b_max);
          free_matrix(count_bad_t5,1,n_slices,1,b_max);
          free_matrix(count_bad_t6,1,n_slices,1,b_max);
          free_ivector(n_aq,1,no_b);
          free_ivector(b_val,1,no_b);
```

```
            return(0);

}     /* main */

/*************************************************************
************/

int   read_image_file(float **rdata, char *fname, FILE
*data_file, int slices, int aq)
{
            char    full_file_name[255],first,second, sl[10],
acq[10];
            int     row, col, rw, co;
            long    value, temp;
        float   answer;

            sprintf(sl,"%d",slices);
            sprintf(acq,"%d",aq);

            strcpy(full_file_name,in_path);
            strcat(full_file_name,st_name);
            strcat(full_file_name,slash);
            strcat(full_file_name,s);
            strcat(full_file_name,im_fdr);
            strcat(full_file_name,slash);
            strcat(full_file_name,sc);
            strcat(full_file_name,slash);
            strcat(full_file_name,fdr);
            strcat(full_file_name,sl);
            strcat(full_file_name,slash);
            strcat(full_file_name,re);
            strcat(full_file_name,s);
            strcat(full_file_name,sl);
            strcat(full_file_name,score);
            strcat(full_file_name,acq);
            strcat(full_file_name,score);
            strcat(full_file_name,fname);
            strcat(full_file_name,".img");

/*          printf("Reading from %s\n",full_file_name);  */
            if ((data_file=fopen(full_file_name,"rb"))==NULL)
return(1);

            /* extract data */
            for (row=1; row<=n_rows; row++){
              for (col=1; col<=n_cols; col++){
                first=fgetc(data_file);
                second=fgetc(data_file);
                temp=first & 0xFF;
                temp=temp<<8;
                value=temp | (second & 0xFF);
                if (value > ((32*1024)-1)) value=value-
(64*1024);
                answer=(float)value;
                rdata[row][col] = answer;
                }
              }

            fclose(data_file);
            return(0);

}  /*   read_data_file   */

/*************************************************************
************/

int   read_cropped_image_file(float **rdata, char *fname, FILE
*data_file, int slices, int aq)
{
            char    full_file_name[255],first,second, sl[10],
acq[10];
            int     row, col;
            long    value, temp;
        float   answer;

            sprintf(sl,"%d",slices);
            sprintf(acq,"%d",aq);

            strcpy(full_file_name,in_path);
            strcat(full_file_name,st_name);
            strcat(full_file_name,slash);
            strcat(full_file_name,s);
            strcat(full_file_name,im_fdr);
            strcat(full_file_name,slash);
            strcat(full_file_name,sc);
            strcat(full_file_name,slash);
            strcat(full_file_name,fdr);
            strcat(full_file_name,sl);
            strcat(full_file_name,slash);
            strcat(full_file_name,re);
            strcat(full_file_name,s);
            strcat(full_file_name,sl);
            strcat(full_file_name,score);
            strcat(full_file_name,acq);
            strcat(full_file_name,score);
            strcat(full_file_name,fname);
            strcat(full_file_name,".img");
            if ((data_file=fopen(full_file_name,"rb"))==NULL)
return(1);

            /* extract data */
            for (row=1; row<=n_rows; row++){
              for (col=1; col <=64; col++) {
                first=fgetc(data_file);
                second=fgetc(data_file);
                temp=first & 0xFF;
                temp=temp<<8;
                value=temp | (second & 0xFF);
                if (value > ((32*1024)-1)) value=value-
(64*1024);
                answer=(float)value;
              }
              for (col=65; col<=n_cols+64; col++){
                first=fgetc(data_file);
                second=fgetc(data_file);
                temp=first & 0xFF;
                temp=temp<<8;
                value=temp | (second & 0xFF);
                if (value > ((32*1024)-1)) value=value-
(64*1024);
                answer=(float)value;
                rdata[row][col-64] = answer;
                }
              for (col=n_cols+65; col <= 2*n_cols; col++){
                first=fgetc(data_file);
                second=fgetc(data_file);
                temp=first & 0xFF;
                temp=temp<<8;
                value=temp | (second & 0xFF);
                if (value > ((32*1024)-1)) value=value-
(64*1024);
                answer=(float)value;
                }
              }
            fclose(data_file);
            return(0);

}  /*   read_data_file   */

/*************************************************************
************/

float sum_dw_images(void)

{
            int    row, col;

            if(b_val[nb] == 0){

              for (row=1; row<=n_rows; row++){
                for (col=1; col<=n_cols; col++) {
                      sumr[row][col]= sumr[row][col] +
(imager[row][col]);
                }
              }
            }
            else
            {
              if(t1_bad[slices][nb][bad_count_t1+1] == bs) {
              bad_count_t1++; }
              else {
/*              printf("\nSumming t1 slice %d, b=%dsmm-2, acq
%d", slices, b_val[nb], bs);  */
                for (row=1; row<=n_rows; row++){
                  for (col=1; col<=n_cols; col++) {
                    sumt1[row][col]= sumt1[row][col] +
(imaget1[row][col]);
                  }
                }

              if(t2_bad[slices][nb][bad_count_t2+1] == bs) {
              bad_count_t2++; }
              else {
/*             printf("\nSumming t2 slice %d, b=%dsmm-2, acq
%d", slices, b_val[nb], bs);  */
                for (row=1; row<=n_rows; row++){
                  for (col=1; col<=n_cols; col++) {
                    sumt2[row][col]= sumt2[row][col] +
(imaget2[row][col]);
                  }
                }

              if(t3_bad[slices][nb][bad_count_t3+1] == bs) {
              bad_count_t3++; }
              else {
/*              printf("\nSumming t3 slice %d, b=%dsmm-2, acq
%d", slices, b_val[nb], bs);  */
                for (row=1; row<=n_rows; row++){
                  for (col=1; col<=n_cols; col++) {
                    sumt3[row][col]= sumt3[row][col] +
(imaget3[row][col]);
                  }
                }

              if(t4_bad[slices][nb][bad_count_t4+1] == bs) {
              bad_count_t4++; }
              else {
/*              printf("\nSumming t4 slice %d, b=%dsmm-2, acq
%d", slices, b_val[nb], bs);  */
                for (row=1; row<=n_rows; row++){
                  for (col=1; col<=n_cols; col++) {
                    sumt4[row][col]= sumt4[row][col] +
(imaget4[row][col]);
                  }
                }

              if(t5_bad[slices][nb][bad_count_t5+1] == bs) {
              bad_count_t5++; }
              else {
/*              printf("\nSumming t5 slice %d, b=%dsmm-2, acq
%d", slices, b_val[nb], bs);  */
                for (row=1; row<=n_rows; row++){
                  for (col=1; col<=n_cols; col++) {
                    sumt5[row][col]= sumt5[row][col] +
(imaget5[row][col]);
                  }
                }
              if(t6_bad[slices][nb][bad_count_t6+1] == bs) {
              bad_count_t6++; }
              else {
/*              printf("\nSumming t6 slice %d, b=%dsmm-2, acq
%d", slices, b_val[nb], bs);  */
                for (row=1; row<=n_rows; row++){
                  for (col=1; col<=n_cols; col++) {
                    sumt6[row][col]= sumt6[row][col] +
(imaget6[row][col]);
                  }
                }
              }
            }

}   /* sum_images */

/*************************************************************
************/

float ave_images(void)

{
            int    row, col;

            for (row=1; row<=n_rows; row++)
            {
               for (col=1; col<=n_cols; col++)
               {
                        if(b_val[1] == 0){
                        if(nb==1){
```

```
                        aver[row][col]= sumr[row][col] /
(float)n_aq[nb];
                        }
                        else{
                        avet1[nb-1][row][col]= sumt1[row][col]
/ (float)n_aq[nb];
                        avet2[nb-1][row][col]= sumt2[row][col]
/ (float)n_aq[nb];
                        avet3[nb-1][row][col]= sumt3[row][col]
/ (float)n_aq[nb];
                        avet4[nb-1][row][col]= sumt4[row][col]
/ (float)n_aq[nb];
                        avet5[nb-1][row][col]= sumt5[row][col]
/ (float)n_aq[nb];
                        avet6[nb-1][row][col]= sumt6[row][col]
/ (float)n_aq[nb];
                        ave_sum[nb-1][row][col] = (avet1[nb-
1][row][col] + avet2[nb-1][row][col] + avet3[nb-1][row][col] +
avet4[nb-1][row][col] + avet5[nb-1][row][col] + avet6[nb-
1][row][col]) / 6.0;
                        }
                        }
                        else{
                        avet1[nb][row][col]= sumt1[row][col] /
((float)n_aq[nb]-count_bad_t1[slices][nb]);
                        avet2[nb][row][col]= sumt2[row][col] /
((float)n_aq[nb]-count_bad_t2[slices][nb]);
                        avet3[nb][row][col]= sumt3[row][col] /
((float)n_aq[nb]-count_bad_t3[slices][nb]);
                        avet4[nb][row][col]= sumt4[row][col] /
((float)n_aq[nb]-count_bad_t4[slices][nb]);
                        avet5[nb][row][col]= sumt5[row][col] /
((float)n_aq[nb]-count_bad_t5[slices][nb]);
                        avet6[nb][row][col]= sumt6[row][col] /
((float)n_aq[nb]-count_bad_t6[slices][nb]);
                        ave_sum[nb][row][col] =
(avet1[nb][row][col] + avet2[nb][row][col] +
avet3[nb][row][col] + avet4[nb][row][col] +
avet5[nb][row][col] + avet6[nb][row][col]) / 6.0;
                        }

                        /* Calculate maximum and minimum image
values */

                        if(b_val[1] == 0){
                        if(nb==1){
                        if((int)aver[row][col] > max_val_r)
                        max_val_r = aver[row][col];
                        }
                        else {
                        if((int)avet1[nb-1][row][col] >
max_val_t1)
                        max_val_t1 = avet1[nb-1][row][col];
                        if((int)avet2[nb-1][row][col] >
max_val_t2)
                        max_val_t2 = avet2[nb-1][row][col];
                        if((int)avet3[nb-1][row][col] >
max_val_t3)
                        max_val_t3 = avet3[nb-1][row][col];
                        if((int)avet4[nb-1][row][col] >
max_val_t4)
                        max_val_t4 = avet4[nb-1][row][col];
                        if((int)avet5[nb-1][row][col] >
max_val_t5)
                        max_val_t5 = avet5[nb-1][row][col];
                        if((int)avet6[nb-1][row][col] >
max_val_t6)
                        max_val_t6 = avet6[nb-1][row][col];
                        if((int)ave_sum[nb-1][row][col] >
max_val_ave_sum)
                        max_val_ave_sum = ave_sum[nb-
1][row][col];
                        }
                        }
                        else{
                        if((int)avet1[nb][row][col] >
max_val_t1)
                        max_val_t1 = avet1[nb][row][col];
                        if((int)avet2[nb][row][col] >
max_val_t2)
                        max_val_t2 = avet2[nb][row][col];
                        if((int)avet3[nb][row][col] >
max_val_t3)
                        max_val_t3 = avet3[nb][row][col];
                        if((int)avet4[nb][row][col] >
max_val_t4)
                        max_val_t4 = avet4[nb][row][col];
                        if((int)avet5[nb][row][col] >
max_val_t5)
                        max_val_t5 = avet5[nb][row][col];
                        if((int)avet6[nb][row][col] >
max_val_t6)
                        max_val_t6 = avet6[nb][row][col];
                        if((int)ave_sum[nb][row][col] >
max_val_ave_sum)
                        max_val_ave_sum =
ave_sum[nb][row][col];
                        }
                    }
                }


}   /* ave_images */


/*************************************************************
**************/

float       tensor_images(void)

{
            int row,col, box_size, box_side, x_offset,
y_offset;
            float sq = 2.0;
            float cu = 3.0;
            float frac = 0.1666666666666666;
            float frac2 = 1.5;
            float **adc_tensor_temp, **asd_tensor_temp,
**avr_tensor_temp, *roi;
            float mean_bas, bas_adve, stdev_bas, bas_var,
bas_skew, bas_curt;
            float mean_tmp1, mean_tmp2, mean_tmp3, mean_tmp4,
mean_tmp5, mean_tmp6;
            float std_tmp1, std_tmp2, std_tmp3, std_tmp4,
std_tmp5, std_tmp6;
            char full_file_name[255], sl[10],vb[10], b[]="b";
            FILE *noise_file;

            box_side=32;
            box_size=box_side*box_side;
            x_offset=32;
            y_offset=48;

/*          max_val_adc_ten = 0;
            min_val_adc_ten = 32767;
            max_val_asd_ten = 0;
            min_val_asd_ten = 32767;
            max_val_avr_ten = 0;
            min_val_avr_ten = 32767;
            max_val_pro_ten = 0;
            min_val_pro_ten = 32767;
            max_val_e1_ten = 0;
            min_val_e1_ten = 32767;
            max_val_e2_ten = 0;
            min_val_e2_ten = 32767;
            max_val_e3_ten = 0;
            min_val_e3_ten = 32767;
            max_val_fa_ten = 0.0;
            min_val_fa_ten = 32767;
*/

            roi=vector(1,box_size);
            eig_val=vector(1,3);
            dtensor_image=matrix(1,3,1,3);
            eig_vect=matrix(1,3,1,3);
            adc_tensor_temp=matrix(1,n_rows,1,n_cols);
            asd_tensor_temp=matrix(1,n_rows,1,n_cols);
            avr_tensor_temp=matrix(1,n_rows,1,n_cols);


            /* Read mean noise level in baseline (lowest b)
image */
            sprintf(sl,"%d",slices);
            sprintf(vb,"%d",b_val[1]);

            mean_bas = 0.0;

            strcpy(full_file_name, in_path);
            strcat(full_file_name, st_name);
            strcat(full_file_name, slash);
            strcat(full_file_name, s);
            strcat(full_file_name, im_fdr);
            strcat(full_file_name, slash);
            strcat(full_file_name, sc);
            strcat(full_file_name, slash);
            strcat(full_file_name, fdr);
            strcat(full_file_name, sl);
            strcat(full_file_name, slash);
            strcat(full_file_name, b);
            strcat(full_file_name, vb);
            strcat(full_file_name, score);
            strcat(full_file_name, sl);
            strcat(full_file_name,".noise");

/*          printf("Reading from: %s\n",full_file_name);  */
            if ((noise_file=fopen(full_file_name,"rb"))==NULL)
return(1);

            /* extract scaling factor */

            if(b_val[1] == 0){
            fscanf(noise_file,"%f %f",&mean_tmp1,&std_tmp1);
            mean_bas=mean_tmp1;
            }
            else{
            fscanf(noise_file,"%f %f %f %f %f %f %f %f %f %f
%f %f", &mean_tmp1, &std_tmp1, &mean_tmp2, &std_tmp2,
&mean_tmp3, &std_tmp3, &mean_tmp4, &std_tmp4, &mean_tmp5,
&std_tmp5, &mean_tmp6, &std_tmp6);

            mean_bas = ((mean_tmp1 + mean_tmp2 + mean_tmp3 +
mean_tmp4 + mean_tmp5 + mean_tmp6) / 6.0);
            }

            fclose(noise_file);

            for (row=1; row<=n_rows; row++)
            {
            for (col=1; col<=n_cols; col++)
            {


            /* define diffusion tensor for each pixel */

            dtensor_image[1][1] = imaxx[row][col];
            dtensor_image[2][2] = imayy[row][col];
            dtensor_image[3][3] = imazz[row][col];
            dtensor_image[1][2] = imaxy[row][col];
            dtensor_image[1][3] = imaxz[row][col];
            dtensor_image[2][3] = imayz[row][col];
            dtensor_image[2][1] = imaxy[row][col];
            dtensor_image[3][1] = imaxz[row][col];
            dtensor_image[3][2] = imayz[row][col];

            /* Calculate eigenvalues and vectors and sort into
order */

            jacobi(dtensor_image, 3, eig_val, eig_vect);
            eigsrt(eig_val, eig_vect, 3);

            /* Store eigenvalues and eigenvectors*/

            eigval1[row][col] = eig_val[1];
            eigval2[row][col] = eig_val[2];
            eigval3[row][col] = eig_val[3];
            eigvec11[row][col] = eig_vect[1][1];
            eigvec12[row][col] = eig_vect[1][2];
            eigvec13[row][col] = eig_vect[1][3];
            eigvec21[row][col] = eig_vect[2][1];
            eigvec22[row][col] = eig_vect[2][2];
            eigvec23[row][col] = eig_vect[2][3];
            eigvec31[row][col] = eig_vect[3][1];
            eigvec32[row][col] = eig_vect[3][2];
            eigvec33[row][col] = eig_vect[3][3];

            /* if signal is below 8x the noise level set maps
to zero */

            if(imagebase[row][col] <= 8*mean_bas) {
```

```
            adc_tensor[row][col] = 0.0;
            asd_tensor[row][col] = 0.0;
            avr_tensor[row][col] = 0.0;
            e1_tensor[row][col] = 0.0;
            e2_tensor[row][col] = 0.0;
            e3_tensor[row][col] = 0.0;
            fa_tensor[row][col] = 0.0;
            pro_tensor[row][col] = 0.0;
            }

            else {

            /* Calculate anisotropy indices */

            e1_tensor[row][col] = (scale_adc*eig_val[1])+1.0;
            e2_tensor[row][col] = (scale_adc*eig_val[2])+1.0;
            e3_tensor[row][col] = (scale_adc*eig_val[3])+1.0;

            adc_tensor[row][col] = scale_adc*((eig_val[1] +
eig_val[2] + eig_val[3]) / 3.0)+1.0;
            adc_tensor_temp[row][col] = (eig_val[1] +
eig_val[2] + eig_val[3]) / (3.0);
            asd_tensor_temp[row][col] =
(sqrt(frac))*(sqrt((pow((((float)eig_val[1]) -
((float)adc_tensor_temp[row][col])), sq)) +
(pow((((float)eig_val[2]) -
((float)adc_tensor_temp[row][col])), sq)) +
(pow((((float)eig_val[3]) -
((float)adc_tensor_temp[row][col])), sq))) /
((float)adc_tensor_temp[row][col]));
            asd_tensor[row][col] = (scale_asd *
asd_tensor_temp[row][col])+1.0;
            avr_tensor[row][col] =  (scale_asd*(1.0-
(((float)eig_val[1]*(float)eig_val[2]*(float)eig_val[3])/(pow(
(((float)eig_val[1]+(float)eig_val[2]+(float)eig_val[3])/3),cu
)))))) + 1.0;
avr_tensor_temp[row][col] =
((((float)eig_val[1]*(float)eig_val[2]*(float)eig_val[3])/(pow
((((float)eig_val[1]+(float)eig_val[2]+(float)eig_val[3])/3),c
u))));

            fa_tensor[row][col] =
1.0+(scale_asd*(sqrt(frac2))*(sqrt((pow((((float)eig_val[1]) -
((float)adc_tensor_temp[row][col])), sq)) +
(pow((((float)eig_val[2]) -
((float)adc_tensor_temp[row][col])), sq)) +
(pow((((float)eig_val[3]) -
((float)adc_tensor_temp[row][col])), sq))) /
(sqrt((pow((((float)eig_val[1])), sq)) +
(pow((((float)eig_val[2])), sq)) + (pow((((float)eig_val[3])),
sq))))));

            aer1_tensor[row][col] =
1.0+(scale_asd*((eig_val[1]-eig_val[3])/(eig_val[1])));
            aer2_tensor[row][col] = 1.0+(scale_asd*(sqrt(1.0 -
pow((eig_val[3]/eig_val[1]), sq))));

            if(asd_tensor_temp[row][col] > 0.25 &&
asd_tensor_temp[row][col] < 0.75)
            pro_tensor[row][col]=1.0+(scale_asd* (1-
(3.0*(pow(((float)asd_tensor_temp[row][col]),sq)))+(2.0*(pow((
(float)asd_tensor_temp[row][col]),cu)))-
(avr_tensor_temp[row][col]))/(4.0*(pow(((float)asd_tensor_temp
[row][col]),cu))));
            else
            pro_tensor[row][col] = 0.0;

            }

            /* scale tensor component images */

            imaxx[row][col] = scale_adc*imaxx[row][col];
            imayy[row][col] = scale_adc*imayy[row][col];
            imazz[row][col] = scale_adc*imazz[row][col];
            imaxy[row][col] = scale_adc*imaxy[row][col];
            imaxz[row][col] = scale_adc*imaxz[row][col];
            imayz[row][col] = scale_adc*imayz[row][col];

            /*  Set error values for out of range pixels */

                        if (avr_tensor[row][col] < 1.0)
                        avr_tensor[row][col] = 0;
                        if (avr_tensor[row][col] >
scale_asd+1.0)
                        avr_tensor[row][col] = 0;
                        if (asd_tensor[row][col] < 1.0)
                        asd_tensor[row][col] = 0;
                        if (asd_tensor[row][col] >
scale_asd+1.0)
                        asd_tensor[row][col] = 0;
/*                      if (adc_tensor[row][col] > 32768.0)
                        adc_tensor[row][col] = 0;  */
                        if (adc_tensor[row][col] < 1.0)
                        adc_tensor[row][col] = 0;
                        if (pro_tensor[row][col] >
scale_asd+1.0)
                        pro_tensor[row][col] = 0;
                        if (pro_tensor[row][col] < 1.0)
                        pro_tensor[row][col] = 0;
                        if (fa_tensor[row][col] >
scale_asd+1.0)
                        fa_tensor[row][col] = 0;
                        if (fa_tensor[row][col] < 1.0)
                        fa_tensor[row][col] = 0;
                        if (aer1_tensor[row][col] >
scale_asd+1.0)
                        aer1_tensor[row][col] = 0;
                        if (aer1_tensor[row][col] < 1.0)
                        aer1_tensor[row][col] = 0;
                        if (aer2_tensor[row][col] >
scale_asd+1.0)
                        aer2_tensor[row][col] = 0;
                        if (aer2_tensor[row][col] < 1.0)
                        aer2_tensor[row][col] = 0;
                        if (e1_tensor[row][col] < 1.0) {
                        e1_tensor[row][col] = 0.0;
                        e2_tensor[row][col] = 0.0;
                        e3_tensor[row][col] = 0.0;  }
                        if (e2_tensor[row][col] < 1.0)  {
                        e1_tensor[row][col] = 0.0;
                        e2_tensor[row][col] = 0.0;
                        e3_tensor[row][col] = 0.0;  }
                        if (e3_tensor[row][col] < 1.0)  {
                        e1_tensor[row][col] = 0.0;
                        e2_tensor[row][col] = 0.0;
                        e3_tensor[row][col] = 0.0;  }

            /* Calculate maximum and minimum image values */

                        if (adc_tensor[row][col] >
max_val_adc_ten)
                        max_val_adc_ten =
adc_tensor[row][col];
                        if (adc_tensor[row][col] <
min_val_adc_ten)
                        min_val_adc_ten =
adc_tensor[row][col];
                        if (asd_tensor[row][col] >
max_val_asd_ten)
                        max_val_asd_ten =
asd_tensor[row][col];
                        if (asd_tensor[row][col] <
min_val_asd_ten)
                        min_val_asd_ten =
asd_tensor[row][col];
                        if (avr_tensor[row][col] >
max_val_avr_ten)
                        max_val_avr_ten =
avr_tensor[row][col];
                        if (avr_tensor[row][col] <
min_val_avr_ten)
                        min_val_avr_ten =
avr_tensor[row][col];
                        if (pro_tensor[row][col] >
max_val_pro_ten)
                        max_val_pro_ten =
pro_tensor[row][col];
                        min_val_pro_ten = 0;
                        if (fa_tensor[row][col] >
max_val_fa_ten)
                        max_val_fa_ten = fa_tensor[row][col];
                        if (fa_tensor[row][col] <
min_val_fa_ten)
                        min_val_fa_ten = fa_tensor[row][col];
                        if (aer1_tensor[row][col] >
max_val_aer1_ten)
                        max_val_aer1_ten =
aer1_tensor[row][col];
                        if (aer1_tensor[row][col] <
min_val_aer1_ten)
                        min_val_aer1_ten =
aer1_tensor[row][col];
                        if (aer2_tensor[row][col] >
max_val_aer2_ten)
                        max_val_aer2_ten =
aer2_tensor[row][col];
                        if (aer2_tensor[row][col] <
min_val_aer2_ten)
                        min_val_aer2_ten =
aer2_tensor[row][col];
                        if (imaxx[row][col] > max_val_dxx_ten)
                        max_val_dxx_ten = imaxx[row][col];
                        if (imaxx[row][col] < min_val_dxx_ten)
                        min_val_dxx_ten = imaxx[row][col];
                        if (imayy[row][col] > max_val_dyy_ten)
                        max_val_dyy_ten = imayy[row][col];
                        if (imayy[row][col] < min_val_dyy_ten)
                        min_val_dyy_ten = imayy[row][col];
                        if (imazz[row][col] > max_val_dzz_ten)
                        max_val_dzz_ten = imazz[row][col];
                        if (imazz[row][col] < min_val_dzz_ten)
                        min_val_dzz_ten = imazz[row][col];
                        if (imaxy[row][col] > max_val_dxy_ten)
                        max_val_dxy_ten = imaxy[row][col];
                        if (imaxy[row][col] < min_val_dxy_ten)
                        min_val_dxy_ten = imaxy[row][col];
                        if (imaxz[row][col] > max_val_dxz_ten)
                        max_val_dxz_ten = imaxz[row][col];
                        if (imaxz[row][col] < min_val_dxz_ten)
                        min_val_dxz_ten = imaxz[row][col];
                        if (imayz[row][col] > max_val_dyz_ten)
                        max_val_dyz_ten = imayz[row][col];
                        if (imayz[row][col] < min_val_dyz_ten)
                        min_val_dyz_ten = imayz[row][col];
                        if (imagebase[row][col] >
max_val_base_ten)
                        max_val_base_ten =
imagebase[row][col];
                        if (imagebase[row][col] <
min_val_base_ten)
                        min_val_base_ten =
imagebase[row][col];
                        if (e1_tensor[row][col] >
max_val_e1_ten)
                        max_val_e1_ten = e1_tensor[row][col];
                        if (e1_tensor[row][col] <
min_val_e1_ten)
                        min_val_e1_ten = e1_tensor[row][col];
                        if (e2_tensor[row][col] >
max_val_e2_ten)
                        max_val_e2_ten = e2_tensor[row][col];
                        if (e2_tensor[row][col] <
min_val_e2_ten)
                        min_val_e2_ten = e2_tensor[row][col];
                        if (e3_tensor[row][col] >
max_val_e3_ten)
                        max_val_e3_ten = e3_tensor[row][col];
                        if (e3_tensor[row][col] <
min_val_e3_ten)
                        min_val_e3_ten = e3_tensor[row][col];
            }
            }

            free_matrix(adc_tensor_temp,1,n_rows,1,n_cols);
            free_matrix(asd_tensor_temp,1,n_rows,1,n_cols);
            free_matrix(avr_tensor_temp,1,n_rows,1,n_cols);
            free_vector(eig_val,1,3);
            free_matrix(dtensor_image,1,3,1,3);
            free_matrix(eig_vect,1,3,1,3);
            free_vector(roi,1,box_size);
}

/*************************************************************
**************/

float  multivariate(int samp_strat)

/* Perform multivariate linear regression to calculate
diffusion tensor */
{
            int i, j, row, col, N, br, bc, nb, k, dep, p, q,
count;
            float **B, **M, **IM, **BT, **tmp, **SB, **S,
**BTS, **S_temp, **x_diag;
```

```
            float *x, *alpha, ***x_temp, max, tp;
            int rw1, rw2, rw3, rw4, rw5, rw6;
            float sq=2.0;
/*
            printf("b_bas = %i \n",b_bas);
            printf("samp_strat = %i \n",samp_strat);
            printf("fov_choice = %i \n",fov_choice);
*/

            if(b_val[1] == 0) {
            br = (6*(no_b-1))+1;
            bc = 7;
            }
            else
            {
            br = (6*(b_max));
            bc = 7;
            }

            B = matrix(1,br,1,bc);
            M = matrix(1,bc,1,bc);
            IM = matrix(1,bc,1,bc);
            BT = matrix(1,bc,1,br);
            tmp = matrix(1,bc,1,br);
            SB = matrix(1,br,1,bc);
            S = matrix(1,br,1,br);
            S_temp = matrix(1,br,1,br);
            x_diag = matrix(1,br,1,br);
            BTS = matrix(1,bc,1,br);
            x = vector(1,br);
            alpha = vector(1,bc);
            x_temp = f3tensor(1,n_rows,1,n_cols,1,br);

            /* Define covariance matrix sigma (S) as 1/(std
dev) */
            /* Need to multiply by mean^2 intensity in each
pixel to obtain S */

            if(no_b > 2) {
            covariance(S_temp);
            }

            for(nb=1;nb<=no_b;nb++) {

            /* reset tensor row numbers */
            rw1=0, rw2=0, rw3=0, rw4=0, rw5=0, rw6=0;

            /* define tensor row numbers */
            if(b_val[1] == 0){
            if(nb>1) {
            rw1=((nb-2)*6)+2;
            rw2=((nb-2)*6)+3;
            rw3=((nb-2)*6)+4;
            rw4=((nb-2)*6)+5;
            rw5=((nb-2)*6)+6;
            rw6=((nb-2)*6)+7;
            }
            }
            else {
            rw1=((nb-1)*6)+1;
            rw2=((nb-1)*6)+2;
            rw3=((nb-1)*6)+3;
            rw4=((nb-1)*6)+4;
            rw5=((nb-1)*6)+5;
            rw6=((nb-1)*6)+6;
            }

            /* B Matrix Definitions */

            if(b_val[1] == 0) {

            B[1][1]=0.00, B[1][2]=0.01, B[1][3]=0.04,
B[1][4]=2*0.00;
            B[1][5]=2*0.00, B[1][6]=2*0.02, B[1][7]=1.0;
            }

            if(samp_strat == 2) {

            if(slew == 1) {

            if(fov_choice == 4) {

            if(b_val[nb] == 100) {
            B[rw1][1]=49.03, B[rw1][2]=0.01, B[rw1][3]=49.31,
B[rw1][4]=2*0.35;
            B[rw1][5]=2*49.16, B[rw1][6]=2*0.35,
B[rw1][7]=1.0;
            B[rw2][1]=49.03, B[rw2][2]=0.01, B[rw2][3]=49.31,
B[rw2][4]=-2*0.35;
            B[rw2][5]=-2*49.16, B[rw2][6]=2*0.35,
B[rw2][7]=1.0;
            B[rw3][1]=0.00, B[rw3][2]=49.02, B[rw3][3]=49.31,
B[rw3][4]=2*0.00;
            B[rw3][5]=2*0.00, B[rw3][6]=2*49.16,
B[rw3][7]=1.0;
            B[rw4][1]=0.00, B[rw4][2]=49.02, B[rw4][3]=48.75,
B[rw4][4]=2*0.00;
            B[rw4][5]=-2*0.00, B[rw4][6]=-2*48.88,
B[rw4][7]=1.0;
            B[rw5][1]=49.03, B[rw5][2]=49.02, B[rw5][3]=0.04,
B[rw5][4]=2*49.02;
            B[rw5][5]=2*0.49, B[rw5][6]=2*0.49, B[rw5][7]=1.0;
        B[rw6][1]=49.03, B[rw6][2]=49.02,
B[rw6][3]=0.04,B[rw6][4]=-2*49.02;
            B[rw6][5]=-2*0.49, B[rw6][6]=2*0.49,
B[rw6][7]=1.0;
            }
            if(b_val[nb] == 300) {
            B[rw1][1]=147.08, B[rw1][2]=0.01,
B[rw1][3]=147.57, B[rw1][4]=2*0.60;
            B[rw1][5]=2*147.32, B[rw1][6]=2*0.61,
B[rw1][7]=1.0;
            B[rw2][1]=147.08, B[rw2][2]=0.01,
B[rw2][3]=147.57, B[rw2][4]=-2*0.60;
            B[rw2][5]=-2*147.32, B[rw2][6]=2*0.61,
B[rw2][7]=1.0;
            B[rw3][1]=0.00, B[rw3][2]=147.08,
B[rw3][3]=147.57, B[rw3][4]=2*0.00;
            B[rw3][5]=2*0.00, B[rw3][6]=2*147.32,
B[rw3][7]=1.0;
            B[rw4][1]=0.00, B[rw4][2]=147.08,
B[rw4][3]=146.60, B[rw4][4]=2*0.00;
            B[rw4][5]=-2*0.00, B[rw4][6]=-2*146.83,
B[rw4][7]=1.0;
            B[rw5][1]=147.08, B[rw5][2]=147.08,
B[rw5][3]=0.04, B[rw5][4]=2*147.08;
            B[rw5][5]=2*0.84, B[rw5][6]=2*0.84, B[rw5][7]=1.0;
        B[rw6][1]=147.08, B[rw6][2]=147.08,
B[rw6][3]=0.04,B[rw6][4]=-2*147.08;
            B[rw6][5]=-2*0.84, B[rw6][6]=2*0.84,
B[rw6][7]=1.0;
            }
            if(b_val[nb] == 700){
            B[rw1][1]=343.14, B[rw1][2]=0.01,
B[rw1][3]=343.89, B[rw1][4]=2*0.91;
            B[rw1][5]=2*343.51, B[rw1][6]=2*0.92,
B[rw1][7]=1.0;
            B[rw2][1]=343.14, B[rw2][2]=0.01,
B[rw2][3]=343.89, B[rw2][4]=-2*0.91;
            B[rw2][5]=-2*343.51, B[rw2][6]=2*0.92,
B[rw2][7]=1.0;
            B[rw3][1]=0.00, B[rw3][2]=343.14,
B[rw3][3]=343.89, B[rw3][4]=2*0.00;
            B[rw3][5]=2*0.00, B[rw3][6]=2*343.51,
B[rw3][7]=1.0;
            B[rw4][1]=0.00, B[rw4][2]=343.14,
B[rw4][3]=342.40, B[rw4][4]=2*0.00;
            B[rw4][5]=-2*0.00, B[rw4][6]=-2*342.76,
B[rw4][7]=1.0;
            B[rw5][1]=343.14, B[rw5][2]=343.14,
B[rw5][3]=0.04, B[rw5][4]=2*343.14;
            B[rw5][5]=2*1.29, B[rw5][6]=2*1.29, B[rw5][7]=1.0;
            B[rw6][1]=343.14, B[rw6][2]=343.14,B[rw6][3]=0.04,
B[rw6][4]=-2*343.14;
            B[rw6][5]=-2*1.29, B[rw6][6]=2*1.29,
B[rw6][7]=1.0;
            }
            if(b_val[nb] == 800){
            B[rw1][1]=392.14, B[rw1][2]=0.01,
B[rw1][3]=392.94, B[rw1][4]=2*0.98;
            B[rw1][5]=2*392.53, B[rw1][6]=2*0.98,
B[rw1][7]=1.0;
            B[rw2][1]=392.14, B[rw2][2]=0.01,
B[rw2][3]=392.94, B[rw2][4]=-2*0.98;
            B[rw2][5]=-2*392.53, B[rw2][6]=2*0.98,
B[rw2][7]=1.0;
            B[rw3][1]=0.00, B[rw3][2]=392.14,
B[rw3][3]=392.94, B[rw3][4]=2*0.00;
            B[rw3][5]=2*0.00, B[rw3][6]=2*392.53,
B[rw3][7]=1.0;
            B[rw4][1]=0.00, B[rw4][2]=392.14,
B[rw4][3]=391.35, B[rw4][4]=2*0.00;
            B[rw4][5]=-2*0.00, B[rw4][6]=-2*391.35,
B[rw4][7]=1.0;
            B[rw5][1]=392.14, B[rw5][2]=392.14,
B[rw5][3]=0.04, B[rw5][4]=2*392.14;
            B[rw5][5]=2*1.37, B[rw5][6]=2*1.37, B[rw5][7]=1.0;
            B[rw6][1]=392.14, B[rw6][2]=392.14,B[rw6][3]=0.04,
B[rw6][4]=-2*392.14;
            B[rw6][5]=-2*1.37, B[rw6][6]=2*1.37,
B[rw6][7]=1.0;
            }
            if(b_val[nb] == 900){
            B[rw1][1]=441.20, B[rw1][2]=0.01,
B[rw1][3]=442.05, B[rw1][4]=2*1.04;
            B[rw1][5]=2*441.62, B[rw1][6]=2*1.04,
B[rw1][7]=1.0;
            B[rw2][1]=441.20, B[rw2][2]=0.01,
B[rw2][3]=442.05, B[rw2][4]=-2*1.04;
            B[rw2][5]=-2*441.62, B[rw2][6]=2*1.04,
B[rw2][7]=1.0;
            B[rw3][1]=0.00, B[rw3][2]=441.20,
B[rw3][3]=442.05, B[rw3][4]=2*0.00;
            B[rw3][5]=2*0.00, B[rw3][6]=2*441.62,
B[rw3][7]=1.0;
            B[rw4][1]=0.00, B[rw4][2]=441.20,
B[rw4][3]=440.37, B[rw4][4]=2*0.00;
            B[rw4][5]=-2*0.00, B[rw4][6]=-2*440.78,
B[rw4][7]=1.0;
            B[rw5][1]=441.20, B[rw5][2]=441.20,
B[rw5][3]=0.04, B[rw5][4]=2*441.20;
            B[rw5][5]=2*1.46, B[rw5][6]=2*1.46, B[rw5][7]=1.0;
            B[rw6][1]=441.20, B[rw6][2]=441.20,B[rw6][3]=0.04,
B[rw6][4]=-2*441.20;
            B[rw6][5]=-2*1.46, B[rw6][6]=2*1.46,
B[rw6][7]=1.0;
            }
            if(b_val[nb] == 1000){
            B[rw1][1]=490.16, B[rw1][2]=0.01,
B[rw1][3]=491.06, B[rw1][4]=2*1.09;
            B[rw1][5]=2*490.61, B[rw1][6]=2*1.10,
B[rw1][7]=1.0;
            B[rw2][1]=490.16, B[rw2][2]=0.01,
B[rw2][3]=491.06, B[rw2][4]=-2*1.09;
            B[rw2][5]=-2*490.61, B[rw2][6]=2*1.10,
B[rw2][7]=1.0;
            B[rw3][1]=0.00, B[rw3][2]=490.16,
B[rw3][3]=491.06, B[rw3][4]=2*0.00;
            B[rw3][5]=2*0.00, B[rw3][6]=2*490.61,
B[rw3][7]=1.0;
            B[rw4][1]=0.00, B[rw4][2]=490.16,
B[rw4][3]=489.29, B[rw4][4]=2*0.00;
            B[rw4][5]=-2*0.00, B[rw4][6]=-2*489.73,
B[rw4][7]=1.0;
            B[rw5][1]=490.16, B[rw5][2]=490.16,
B[rw5][3]=0.04, B[rw5][4]=2*490.16;
            B[rw5][5]=2*1.54, B[rw5][6]=2*1.54, B[rw5][7]=1.0;
            B[rw6][1]=490.16, B[rw6][2]=490.16,B[rw6][3]=0.04,
B[rw6][4]=-2*490.16;
            B[rw6][5]=-2*1.54, B[rw6][6]=2*1.54,
B[rw6][7]=1.0;
            }
            if(b_val[nb] == 1200){
            B[rw1][1]=588.21, B[rw1][2]=0.01,
B[rw1][3]=589.20, B[rw1][4]=2*1.20;
            B[rw1][5]=2*588.70, B[rw1][6]=2*1.20,
B[rw1][7]=1.0;
            B[rw2][1]=588.21, B[rw2][2]=0.01,
B[rw2][3]=589.20, B[rw2][4]=-2*1.20;
            B[rw2][5]=-2*588.70, B[rw2][6]=2*1.20,
B[rw2][7]=1.0;
            B[rw3][1]=0.00, B[rw3][2]=588.21,
B[rw3][3]=589.20, B[rw3][4]=2*0.00;
            B[rw3][5]=2*0.00, B[rw3][6]=2*588.70,
B[rw3][7]=1.0;
            B[rw4][1]=0.00, B[rw4][2]=588.21,
B[rw4][3]=587.25, B[rw4][4]=2*0.00;
            B[rw4][5]=-2*0.00, B[rw4][6]=-2*587.73,
B[rw4][7]=1.0;
            B[rw5][1]=588.21, B[rw5][2]=588.21,
B[rw5][3]=0.04, B[rw5][4]=2*588.21;
            B[rw5][5]=2*1.68, B[rw5][6]=2*1.68, B[rw5][7]=1.0;
            B[rw6][1]=588.21, B[rw6][2]=588.21,B[rw6][3]=0.04,
B[rw6][4]=-2*588.21;
```

```
            B[rw6][5]=-2*1.68, B[rw6][6]=2*1.68,
B[rw6][7]=1.0;
            }
            if(b_val[nb] == 1300){
            B[rw1][1]=637.39, B[rw1][2]=0.01,
B[rw1][3]=638.42, B[rw1][4]=2*1.25;
            B[rw1][5]=2*637.90, B[rw1][6]=2*1.25,
B[rw1][7]=1.0;
            B[rw2][1]=637.39, B[rw2][2]=0.01,
B[rw2][3]=638.42, B[rw2][4]=-2*1.25;
            B[rw2][5]=-2*637.90, B[rw2][6]=2*1.25,
B[rw2][7]=1.0;
            B[rw3][1]=0.00, B[rw3][2]=637.39,
B[rw3][3]=638.42, B[rw3][4]=2*0.00;
            B[rw3][5]=2*0.00, B[rw3][6]=2*637.90,
B[rw3][7]=1.0;
            B[rw4][1]=0.00, B[rw4][2]=637.39,
B[rw4][3]=636.38, B[rw4][4]=2*0.00;
            B[rw4][5]=-2*0.00, B[rw4][6]=-2*636.38,
B[rw4][7]=1.0;
            B[rw5][1]=637.39, B[rw5][2]=637.39,
B[rw5][3]=0.04, B[rw5][4]=2*637.39;
            B[rw5][5]=2*1.75, B[rw5][6]=2*1.75, B[rw5][7]=1.0;
            B[rw6][1]=637.39, B[rw6][2]=637.39,B[rw6][3]=0.04,
B[rw6][4]=-2*637.39;
            B[rw6][5]=-2*1.75, B[rw6][6]=2*1.75,
B[rw6][7]=1.0;
            }
            }
            }
            }

            if(samp_strat == 1) {

            if(fov_choice == 4) {

            if(b_val[nb] == 700)  {
            B[rw1][1]=672.47, B[rw1][2]=0.01, B[rw1][3]=0.04,
B[rw1][4]=2*1.28;
            B[rw1][5]=2*1.80, B[rw1][6]=2*0.02, B[rw1][7]=1.0;
            B[rw2][1]=0.00, B[rw2][2]=672.47, B[rw2][3]=0.04,
B[rw2][4]=2*0.00;
            B[rw2][5]=2*0.00, B[rw2][6]=2*1.80, B[rw2][7]=1.0;
            B[rw3][1]=0.00, B[rw3][2]=0.01, B[rw3][3]=673.52,
B[rw3][4]=2*0.00;
            B[rw3][5]=2*0.00, B[rw3][6]=2*1.29, B[rw3][7]=1.0,
            B[rw4][1]=343.14, B[rw4][2]=343.14,
B[rw4][3]=0.04,B[rw4][4]=2*343.14;
            B[rw4][5]=2*1.29, B[rw4][6]=2*1.29, B[rw4][7]=1.0;
            B[rw5][1]=343.14, B[rw5][2]=0.01,
B[rw5][3]=343.89, B[rw5][4]=2*0.91;
            B[rw5][5]=2*343.51, B[rw5][6]=2*0.92,
B[rw5][7]=1.0;
            B[rw6][1]=0.00, B[rw6][2]=343.14,
B[rw6][3]=343.89, B[rw6][4]=2*0.00;
            B[rw6][5]=2*0.00, B[rw6][6]=2*343.51,
B[rw6][7]=1.0;
            }
            }
            }

            }           /* end loop nb to no_b */

            for (row=1; row<=n_rows; row++)
            {

            for (col=1; col<=n_cols; col++)
            {


            for(k=1; k<=br; k++) {
            x[k]=0.0;
            }

            if(b_val[1] == 0){
            x[1] = aver[row][col]+1.0;
            for(nb=1;nb<=b_max;nb++){
            x[((nb-1)*6)+2] = avet1[nb][row][col]+1.0;
            x[((nb-1)*6)+3] = avet2[nb][row][col]+1.0;
            x[((nb-1)*6)+4] = avet3[nb][row][col]+1.0;
            x[((nb-1)*6)+5] = avet4[nb][row][col]+1.0;
            x[((nb-1)*6)+6] = avet5[nb][row][col]+1.0;
            x[((nb-1)*6)+7] = avet6[nb][row][col]+1.0;
            }
            }
            else
            {
            for(nb=1;nb<=b_max;nb++){
            x[((nb-1)*6)+1] = avet1[nb][row][col]+1.0;
            x[((nb-1)*6)+2] = avet2[nb][row][col]+1.0;
            x[((nb-1)*6)+3] = avet3[nb][row][col]+1.0;
            x[((nb-1)*6)+4] = avet4[nb][row][col]+1.0;
            x[((nb-1)*6)+5] = avet5[nb][row][col]+1.0;
            x[((nb-1)*6)+6] = avet6[nb][row][col]+1.0;
            }
            }

            max=0.0;
            if(no_b > 2) {
              for(i=1; i<=br;i++) {
            if(x[i] > max) max=x[i];
              }
            }
            if(max == 0.0) {
              for(i=1; i<=br; i++) {
                for(j=1; j<=br; j++) {
                  if(i==j) {
                  S[i][j] = 1.0;
                  }
                  else {
                  S[i][j] = 0.0;
                  }
                }
              }
            }
            else {
              for(i=1; i<=br; i++) {
                for(j=1; j<=br; j++) {
                  if(i==j) {
            x_diag[i][j] = pow(x[i],sq);
                  }
                  else {
            x_diag[i][j] = 0.0;
                  }
                }
              }
```

```
            /* Obtain covariance matrix S by multiplying
S_temp and x_diag */

            multmat(x_diag, S_temp, S, br, br, br);

            }

            for (i=1; i<=br; i++) {
              if (x[i] != 0)
                x[i] = -log(x[i]);
            }

            transpose(B, BT, br, bc);                    /*
transpose of B is BT */

            multmat(S, B, SB, br, br, bc); /*multiply S by B
return as SB */

            multmat(BT, SB, M, bc, br, bc);  /*multiply BT by
SB return as M */

            for(i=1;i<=bc;i++) {
            max=0.0;
            count=0;
              for(j=1;j<=bc;j++)
                      if((tp=fabs(M[i][j])) > max) max=tp;
              if (max == 0.0) {
              count=count+1;
              for(p=1; p<=br; p++) {
                for(q=1; q<=br; q++) {
                  if(p==q) {
                  S[p][q] = 1.0;
                  }
                  else {
                  S[p][q] = 0.0;
                  }
                }
              }
            multmat(S, B, SB, br, br, bc); /*multiply S by B
return as SB */

            multmat(BT, SB, M, bc, br, bc);  /*multiply BT by
SB return as M */
            }
            }

            inverse(M, IM, bc);                    /*inverse of
M returned in IM */

            multmat(BT, S, BTS, bc, br, br);   /*multiply BT
by S return as BTS */

            multmat(IM, BTS, tmp, bc, bc, br); /*multiply IM
and BTS return as tmp*/

            multmatvec(tmp, x, alpha, bc, br);  /*matrix tmp *
vect x */

            imaxx[row][col] = alpha[1];
            imayy[row][col] = alpha[2];
            imazz[row][col] = alpha[3];
            imaxy[row][col] = alpha[4];
            imaxz[row][col] = alpha[5];
            imayz[row][col] = alpha[6];
            imagebase[row][col] = exp(-alpha[7]);

            }
/*          printf("%d", row);  */
            }

            if(count >= 1) {
            printf("%d singular matrices found....\n",count);
}


/*          for(col=1;col<=256;col++){
            printf("base: %e  ",imagebase[64][col]);
            printf("ave : %e  ",aver[64][col]);  }  */

            free_vector(alpha,1,bc);
            free_vector(x,1,br);
            free_matrix(B,1,br,1,bc);
            free_matrix(M,1,bc,1,bc);
            free_matrix(IM,1,bc,1,bc);
            free_matrix(BT,1,bc,1,br);
            free_matrix(tmp,1,bc,1,br);
            free_matrix(SB,1,br,1,bc);
            free_matrix(S,1,br,1,br);
            free_matrix(S_temp,1,br,1,br);
            free_matrix(x_diag,1,br,1,br);
            free_matrix(BTS,1,bc,1,br);
            free_f3tensor(x_temp,1,n_rows,1,n_cols,1,br);
}

/*************************************************************
**************/

float covariance(float **cov)

/* Calculates the covariance matrix by sampling the background
noise */

{
            int row, col, dep, box_size, box_side, numbr,
numbc, matsze;
            int x_offset, y_offset;
            float
*mean_t1,*mean_t2,*mean_t3,*mean_t4,*mean_t5,*mean_t6, mean_r;
            float
*stdev_t1,*stdev_t2,*stdev_t3,*stdev_t4,*stdev_t5,*stdev_t6;
            float stdev_r, sq=2.0;
            float t1ave, t2ave, t3ave, t4ave, t5ave, t6ave,
rave;
            float t1adve, t2adve, t3adve, t4adve, t5adve,
t6adve, radve;
            float t1sdev, t2sdev, t3sdev, t4sdev, t5sdev,
t6sdev, rsdev;
            float t1var, t2var, t3var, t4var, t5var, t6var,
rvar;
            float t1skew, t2skew, t3skew, t4skew, t5skew,
t6skew, rskew;
            float t1curt, t2curt, t3curt, t4curt, t5curt,
t6curt, rcurt;
            char full_file_name[255], s1[10],vb[10], b[]="b";
            FILE *noise_file;
```

```
            /* Define size & position of ROI for noise
measurement (in pixels) */
            box_side=32;
            box_size=box_side*box_side;
            x_offset=32;
            y_offset=48;

            if(b_val[1] == 0) {
            matsze = (6*(no_b-1))+1;
            }
            else {
            matsze = (6*(no_b));
            }

            /* Define matrices and vectors */

            mean_t1=vector(1,b_max);
            stdev_t1=vector(1,b_max);
            mean_t2=vector(1,b_max);
            stdev_t2=vector(1,b_max);
            mean_t3=vector(1,b_max);
            stdev_t3=vector(1,b_max);
            mean_t4=vector(1,b_max);
            stdev_t4=vector(1,b_max);
            mean_t5=vector(1,b_max);
            stdev_t5=vector(1,b_max);
            mean_t6=vector(1,b_max);
            stdev_t6=vector(1,b_max);

            /* Reset matrices and vectors */

            for(row=1; row<=b_max; row++) {
            mean_t1[row]=0;
            stdev_t1[row]=0;
            mean_t2[row]=0;
            stdev_t2[row]=0;
            mean_t3[row]=0;
            stdev_t3[row]=0;
            mean_t4[row]=0;
            stdev_t4[row]=0;
            mean_t5[row]=0;
            stdev_t5[row]=0;
            mean_t6[row]=0;
            stdev_t6[row]=0;
            }

            for(dep=1; dep<=b_max; dep++) {

            /* read in noise data for each b_value */

            sprintf(sl,"%d",slices);
            sprintf(vb,"%d",b_val[dep]);

            strcpy(full_file_name, in_path);
            strcat(full_file_name, st_name);
            strcat(full_file_name, slash);
            strcat(full_file_name, s);
            strcat(full_file_name, im_fdr);
            strcat(full_file_name, slash);
            strcat(full_file_name, sc);
            strcat(full_file_name, slash);
            strcat(full_file_name, fdr);
            strcat(full_file_name, sl);
            strcat(full_file_name, slash);
            strcat(full_file_name, b);
            strcat(full_file_name, vb);
            strcat(full_file_name, score);
            strcat(full_file_name, sl);
            strcat(full_file_name,".noise");

/*          printf("Reading from: %s\n",full_file_name);  */
            if ((noise_file=fopen(full_file_name,"rb"))==NULL)
return(1);

            /* extract scaling factor */

            if(b_val[dep] == 0){
            fscanf(noise_file,"%f %f",&rave,&rsdev);
            }
            else{
            fscanf(noise_file,"%f %f %f %f %f %f %f %f %f %f %f
%f %f ", &t1ave, &t1sdev, &t2ave, &t2sdev, &t3ave, &t3sdev,
&t4ave, &t4sdev, &t5ave, &t5sdev, &t6ave, &t6sdev);
            }

/*          printf("Image Scale Factor:  %e\n",*scale); */

            fclose(noise_file);
                mean_t1[dep]=t1ave;
                stdev_t1[dep]=t1sdev;
                mean_t2[dep]=t2ave;
                stdev_t2[dep]=t2sdev;
                mean_t3[dep]=t3ave;
                stdev_t3[dep]=t3sdev;
                mean_t4[dep]=t4ave;
                stdev_t4[dep]=t4sdev;
                mean_t5[dep]=t5ave;
                stdev_t5[dep]=t5sdev;
                mean_t6[dep]=t6ave;
                stdev_t6[dep]=t6sdev;
            }

            /* Construct covariance matrix */

            if(b_val[1] == 0) {

              for(row=1; row<=matsze; row++) {
                for(col=1; col<=matsze; col++) {
                  if(row == col) {
                    for(numbr=1; numbr<=no_b; numbr++) {
                        if(numbr==1) {
            cov[1][1] = 1/(pow((rsdev),sq));
                        }
                        else {
            cov[((numbr-2)*6)+2][((numbr-
2)*6)+2]=1/(pow((stdev_t1[numbr-1]),sq));
            cov[((numbr-2)*6)+3][((numbr-
2)*6)+3]=1/(pow((stdev_t2[numbr-1]),sq));
            cov[((numbr-2)*6)+4][((numbr-
2)*6)+4]=1/(pow((stdev_t3[numbr-1]),sq));
            cov[((numbr-2)*6)+5][((numbr-
2)*6)+5]=1/(pow((stdev_t4[numbr-1]),sq));
            cov[((numbr-2)*6)+6][((numbr-
2)*6)+6]=1/(pow((stdev_t5[numbr-1]),sq));
            cov[((numbr-2)*6)+7][((numbr-
2)*6)+7]=1/(pow((stdev_t6[numbr-1]),sq));
                        }
                    }
                  }
                  else {
            cov[row][col] = 0.0;
                  }
                }
              }
              else {
                for(row=1; row<=matsze; row++) {
                  for(col=1; col<=matsze; col++) {
                    if(row == col) {
                      for(numbr=1; numbr<=no_b; numbr++) {
            cov[((numbr-1)*6)+1][((numbr-
1)*6)+1]=1/(pow((stdev_t1[numbr]),sq));
            cov[((numbr-1)*6)+2][((numbr-
1)*6)+2]=1/(pow((stdev_t2[numbr]),sq));
            cov[((numbr-1)*6)+3][((numbr-
1)*6)+3]=1/(pow((stdev_t3[numbr]),sq));
            cov[((numbr-1)*6)+4][((numbr-
1)*6)+4]=1/(pow((stdev_t4[numbr]),sq));
            cov[((numbr-1)*6)+5][((numbr-
1)*6)+5]=1/(pow((stdev_t5[numbr]),sq));
            cov[((numbr-1)*6)+6][((numbr-
1)*6)+6]=1/(pow((stdev_t6[numbr]),sq));
                      }
                    }
                    else {
            cov[row][col] = 0.0;
                    }
                  }
                }
              }

              free_vector(mean_t1,1,b_max);
              free_vector(stdev_t1,1,b_max);
              free_vector(mean_t2,1,b_max);
              free_vector(stdev_t2,1,b_max);
              free_vector(mean_t3,1,b_max);
              free_vector(stdev_t3,1,b_max);
              free_vector(mean_t4,1,b_max);
              free_vector(stdev_t4,1,b_max);
              free_vector(mean_t5,1,b_max);
              free_vector(stdev_t5,1,b_max);
              free_vector(mean_t6,1,b_max);
              free_vector(stdev_t6,1,b_max);

}

/*************************************************************
**************/

int   save_image_file(char *fname, FILE *out_file, float
**calc)

/* save the calculated image file with the given file name */

{
            int     row, col;
            char    full_file_name[255], label[30];
            char    high,low;
            FILE    *data_file;

            strcpy(full_file_name, in_path);
            strcat(full_file_name, st_name);
            strcat(full_file_name, slash);
            strcat(full_file_name, s);
            strcat(full_file_name, im_fdr);
            strcat(full_file_name, slash);
            strcat(full_file_name, map);
            strcat(full_file_name, slash);
            strcat(full_file_name, re);
            strcat(full_file_name, fname);
            strcat(full_file_name,".img");
            if ((out_file=fopen(full_file_name,"ab"))==NULL)
return(1);

            /* save data */
            for (row=1; row<=n_rows; row++)
            {
                        for (col=1; col <= n_cols; col++)
                        {

            high=(int)calc[row][col]/256;

            low=(int)calc[row][col]%256;
                                    fputc(high,out_file);
                                    fputc(low,out_file);
                        }
            }
            fclose(out_file);
            return(0);

}    /* save_image_file */

/*************************************************************
**************/

int   save_f3_file(char *fname, FILE *out_file, float ***calc)

/* save the calculated image file with the given file name */

{
            int     row, col;
            char    full_file_name[255], label[30];
            char    high,low;
            FILE    *data_file;

            strcpy(full_file_name, in_path);
            strcat(full_file_name, st_name);
            strcat(full_file_name, slash);
            strcat(full_file_name, s);
            strcat(full_file_name, im_fdr);
            strcat(full_file_name, slash);
            strcat(full_file_name, map);
            strcat(full_file_name, slash);
            strcat(full_file_name, re);
            strcat(full_file_name, fname);
            strcat(full_file_name,".img");
            if ((out_file=fopen(full_file_name,"ab"))==NULL)
return(1);

            /* save data */
            for (row=1; row<=n_rows; row++)
            {
                        for (col=1; col <= n_cols; col++)
                        {
```

```
                                        if(b_val[1] == 0) {
                                        high=(int)calc[nb-
1][row][col]/256;
                                        low=(int)calc[nb-
1][row][col]%256;
                                        }
                                        else {

                high=(int)calc[nb][row][col]/256;

                low=(int)calc[nb][row][col]%256;
                                        }
                                        fputc(high,out_file);
                                        fputc(low,out_file);
                        }
                }
                fclose(out_file);
                return(0);

}    /* save_f3_file */

/*************************************************************
**************/

int   save_cropped_f3_file(char *fname, FILE *out_file, float
***calc)

/* save the calculated image file with the given file name */

{
                int     row, col;
                char    full_file_name[255], label[30];
                char    high,low;
                FILE    *data_file;

                strcpy(full_file_name, in_path);
                strcat(full_file_name, st_name);
                strcat(full_file_name, slash);
                strcat(full_file_name, s);
                strcat(full_file_name, im_fdr);
                strcat(full_file_name, slash);
                strcat(full_file_name, map);
                strcat(full_file_name, slash);
                strcat(full_file_name, re);
                strcat(full_file_name, fname);
                strcat(full_file_name,".img");
                if ((out_file=fopen(full_file_name,"ab"))==NULL)
return(1);

                /* save data */

                for (row=1; row<=n_rows; row++){
                                for (col=1; col<=64; col++){
                                        if(b_val[1] == 0){
                        calc[nb-1][row][col]=calc[nb-
1][row][col];
                        calc[nb-1][row][col]=calc[nb-
1][row][col];
                                        }
                                        else {
                        calc[nb][row][col]=calc[nb][row][col];
                        calc[nb][row][col]=calc[nb][row][col];
                                        }
                  }
                  for (col=65; col<=192; col++){
                                        if(b_val[1] == 0){
                                        high=(int)calc[nb-
1][row][col]/256;
                                        low=(int)calc[nb-
1][row][col]%256;
                                        }
                                        else {

                high=(int)calc[nb][row][col]/256;

                low=(int)calc[nb][row][col]%256;
                                        }
                                        fputc(high,out_file);
                                        fputc(low,out_file);
                  }
                                for (col=193; col<=256; col++){
                                        if(b_val[1] == 0){
                        calc[nb-1][row][col]=calc[nb-
1][row][col];
                        calc[nb-1][row][col]=calc[nb-
1][row][col];
                                        }
                                        else {
                        calc[nb][row][col]=calc[nb][row][col];
                        calc[nb][row][col]=calc[nb][row][col];
                                        }
                  }
                }

                for (row=1; row<=n_rows; row++)
                {
                        for (col=1; col <= n_cols; col++)
                        {
                                        if(b_val[1] == 0) {
                                        high=(int)calc[nb-
1][row][col]/256;
                                        low=(int)calc[nb-
1][row][col]%256;
                                        }
                                        else {

                high=(int)calc[nb][row][col]/256;

                low=(int)calc[nb][row][col]%256;
                                        }
                                        fputc(high,out_file);
                                        fputc(low,out_file);
                        }
                }
                fclose(out_file);
                return(0);

}    /* save_cropped_f3_file */

/*************************************************************
**************/

int   save_cropped_image_file(char *fname, FILE *out_file,
float **calc)

/* save the calculated image file with the given file name */

{
                int     row, col;
                char    full_file_name[255], label[30];
                char    high,low;
                FILE    *data_file;

                strcpy(full_file_name, in_path);
                strcat(full_file_name, st_name);
                strcat(full_file_name, slash);
                strcat(full_file_name, s);
                strcat(full_file_name, im_fdr);
                strcat(full_file_name, slash);
                strcat(full_file_name, map);
                strcat(full_file_name, slash);
                strcat(full_file_name, re);
                strcat(full_file_name, fname);
                strcat(full_file_name,".img");
                if ((out_file=fopen(full_file_name,"ab"))==NULL)
return(1);

                /* save data */

                for (row=1; row<=n_rows; row++){
                                for (col=1; col<=64; col++){
                        calc[row][col]=calc[row][col];
                        calc[row][col]=calc[row][col];
                  }
                  for (col=65; col<=192; col++){

                high=(int)calc[row][col]/256;

                low=(int)calc[row][col]%256;
                                        fputc(high,out_file);
                                        fputc(low,out_file);
                  }
                                for (col=193; col<=256; col++){
                        calc[row][col]=calc[row][col];
                        calc[row][col]=calc[row][col];
                  }
                }
                fclose(out_file);
                return(0);

}    /* save_image_file */

/*************************************************************
**************/

int  save_cropped_matlab_file(char *fname, FILE *mat_file,
float **data)

/* save the given data set in a form readable by matlab with
extension .mdf */

{
                int row, col;
                char full_file_name[255], mat[]="matlab", sl[10];

                sprintf(sl,"%d",slices);

                strcpy(full_file_name, in_path);
                strcat(full_file_name, st_name);
                strcat(full_file_name, slash);
                strcat(full_file_name, s);
                strcat(full_file_name, im_fdr);
                strcat(full_file_name, slash);
                strcat(full_file_name, mat);
                strcat(full_file_name, slash);
                strcat(full_file_name, re);
                strcat(full_file_name, fname);
                strcat(full_file_name, score);
                strcat(full_file_name, sl);
                strcat(full_file_name,".mdf");

/*              printf("Save file as %s\n", full_file_name);    */
                mat_file=fopen(full_file_name,"wb");

                for(row=1; row<=n_rows; row++)
                {
                  for(col=1; col<=64; col++) {
                  data[row][col] = data[row][col];
                  }
                  for(col=65; col<=192; col++)
                  {
                    fprintf(mat_file,"%f ",data[row][col]);
                  }
                  fputc(0X0A,mat_file);
                  for(col=193; col<=256; col++) {
                  data[row][col] = data[row][col];
                  }
                }
                fputc(0X0A,mat_file);

                fclose(mat_file);
                return(0);

}

/* save_cropped_matlab_file */

/*************************************************************
**************/
int  save_matlab_file(char *fname, FILE *mat_file, float
**data)

/* save the given data set in a form readable by matlab with
extension .mdf */

{
                int row, col;
                char full_file_name[255], mat[]="matlab", sl[10];

                sprintf(sl,"%d",slices);

                strcpy(full_file_name, in_path);
                strcat(full_file_name, st_name);
                strcat(full_file_name, slash);
                strcat(full_file_name, s);
                strcat(full_file_name, im_fdr);
                strcat(full_file_name, slash);
                strcat(full_file_name, mat);
                strcat(full_file_name, slash);
                strcat(full_file_name, re);
                strcat(full_file_name, fname);
                strcat(full_file_name, score);
                strcat(full_file_name, sl);
                strcat(full_file_name,".mdf");
```

```
/*          printf("Save file as %s\n", full_file_name);   */
            mat_file=fopen(full_file_name,"wb");

            for(row=1; row<=n_rows; row++)
            {
              for(col=1; col<=n_cols; col++)
              {
                fprintf(mat_file,"%f ",data[row][col]);
              }
              fputc(0X0A,mat_file);
            }
            fputc(0X0A,mat_file);

            fclose(mat_file);
            return(0);

}

/* save_matlab_file */

/*************************************************************
***************/

void nrerror(char error_text[])


/* Print an error message and exit the program
*/

{

            fprintf(stderr,"Numerical Recipes run-time
error...\n");

            fprintf(stderr,"%s\n",error_text);

            fprintf(stderr,"...now exiting to system...\n");

            exit(1);

}       /* nrerror */


/*************************************************************
***********/


int **imatrix(int nrl,int nrh,int ncl,int nch)


{

            int i,**m;


            m=(int **)malloc((unsigned) (nrh-
nrl+1)*sizeof(int*));

            if (!m) nrerror("allocation failure 1 in
imatrix()");

            m -= nrl;


            for(i=nrl;i<=nrh;i++) {

                        m[i]=(int *)malloc((unsigned) (nch-
ncl+1)*sizeof(int));

                        if (!m[i]) nrerror("allocation failure
2 in imatrix()");

                        m[i] -= ncl;

            }

            return m;

}


/*************************************************************
**********/


void free_imatrix(int **m,int nrl,int nrh,int ncl,int nch)

{

            int i;


            for(i=nrh;i>=nrl;i--) free((char*) (m[i]+ncl));

            free((char*) (m+nrl));

}


/*************************************************************
******************/
double **dmatrix(int nrl,int nrh,int ncl,int nch)

/* Declare a dynamic, two dimensional double precision matrix
with

   elements [nrl..nrh][ncl..nch]      */


{

            int i;

            double **m;

            m=(double **) malloc((unsigned) (nrh-
nrl+1)*sizeof(double*));

            if (!m) nrerror("allocation failure 1 in
dmatrix()");

            m -= nrl;


            for(i=nrl;i<=nrh;i++) {

                        m[i]=(double *) malloc((unsigned)
(nch-ncl+1)*sizeof(double));

                        if (!m[i]) nrerror("allocation failure
2 in dmatrix()");

                        m[i] -= ncl;

            }

            return m;

}      /* dmatrix */


/*************************************************************
************/


void free_dmatrix(double **m,int nrl,int nrh,int ncl,int nch)


/* Free memory previously allocated by dmatrix()      */


{

            int i;


            for(i=nrh;i>=nrl;i--) free((char*) (m[i]+ncl));

            free((char*) (m+nrl));

}     /* free_dmatrix   */

/*************************************************************
*****************/

/* A program to write an analyze header file given the
appropriate input */
/* Written by Biomedical Imaging Resource */
/* Adapted for our uses on 19th March 1998 by PAA */

void make_hdr(char *hdr_name, short int nrw, short int cw,
short int nsl, short int nv, char *type, int max_val, int
min_val, float x_dim, float y_dim, float z_dim)
{

struct header_key
            {
              int sizeof_hdr;
              char data_type[10];
              char db_name[18];
              int extents;
              short int session_error;
              char regular;
              char hkey_un0;
            };

struct image_dimension
            {
            short int dim[8];
            short int unused8;
            short int unused9;
            short int unused10;
            short int unused11;
            short int unused12;
            short int unused13;
            short int unused14;
            short int datatype;
            short int bitpix;
            short int dim_un0;
            float pixdim[8];
            float funused8;
            float funused9;
            float funused10;
            float funused11;
            float funused12;
            float funused13;
            float compressed;
            float verified;
            int glmax,glmin;
            char vox_units[30];
            char cal_units[4];
            float vox_offset;
            float cal_max;
            float cal_min;
            };

struct data_history
            {
            char descrip[80];
            char aux_file[24];
            char orient;
            char originator[10];
            char generated[10];
            char scannum[10];
            char patient_id[10];
            char exp_date[10];
            char exp_time[10];
            char hist_un0[3];
            int views;
            int vols_added;
            int start_field;
            int field_skip;
            int omax, omin;
```

```
            int smax,smin;
            };

struct dsr
            {
            struct header_key hk;
            struct image_dimension dime;
            struct data_history hist;
            };

            int i;
            struct dsr hdr;
            FILE *fp;
            static char DataTypes[9][12] =
{"UNKNOWN","BINARY","CHAR","SHORT","INT","FLOAT","COMPLEX","DO
UBLE","RGB"};
            static int DataTypeSizes[9] =
{0,1,8,16,32,32,64,64,24};
            char full_file_name[255];

            strcpy(full_file_name, in_path);
            strcat(full_file_name, st_name);
            strcat(full_file_name, slash);
            strcat(full_file_name, s);
            strcat(full_file_name, im_fdr);
            strcat(full_file_name, slash);
            strcat(full_file_name, map);
            strcat(full_file_name, slash);
            strcat(full_file_name, re);
            strcat(full_file_name, hdr_name);
            strcat(full_file_name,".hdr");

            memset(&hdr,0,sizeof(struct dsr));
            for(i=0;i<8;i++)
            hdr.dime.pixdim[i] = 0.0;

            hdr.dime.vox_offset = 0.0;
            hdr.dime.cal_max = 0.0;
            hdr.dime.cal_min = 0.0;

            hdr.dime.datatype = -1;

            for(i=1;i<8;i++)
              if(!strcmp(type,DataTypes[i]))
              {
                hdr.dime.datatype = (1<<(i-1));
                hdr.dime.bitpix = DataTypeSizes[i];
                break;
              }

            if(hdr.dime.datatype <= 0)
            {
              printf("<%s> is an unacceptable datatype \n\n",
type);
              exit(0);
            }

            if((fp=fopen(full_file_name,"w"))==0)
            {
              printf("unable to create: %s\n",full_file_name);
              exit(0);
            }

            hdr.dime.dim[0] = 4;  /* all analyze images are
taken as 4 dimensional*/
            hdr.hk.regular = 'r';
            hdr.hk.sizeof_hdr = sizeof(struct dsr);

            hdr.dime.dim[1] = nrw; /* slice width in pixels */
            hdr.dime.dim[2] = ncw; /* slice height in pixels
*/
            hdr.dime.dim[3] = nsl; /* volume depth in slices
*/
            hdr.dime.dim[4] = nv; /* number of volumes per
file */

            hdr.dime.glmax = max_val; /* maximum voxel value
*/
            hdr.dime.glmin = min_val; /* minimum voxel value
*/

            /* set the voxel dimension fields;
                        A value of 0.0 for these fields
implies that the value is
                        unknown.  change these values to what
is appropriate for the
                        sequence or pass additional command
line arguments */

            hdr.dime.pixdim[1] = x_dim; /* voxel x dimension
*/
            hdr.dime.pixdim[2] = y_dim; /* voxel y dimension
*/
            hdr.dime.pixdim[3] = z_dim; /* voxel z dimension,
slice thickness */

            /* Assume zero offset in .img file, byte at which
pixel data starts
                        in the image file */

            hdr.dime.vox_offset = 0.0;

            /* Planar Orientation; */
            /* Movie flag OFF : 0=transverse, 1=coronal,
2=saggital
               Movie flag ON : 3=transverse,
4=coronal,5=saggital */

            hdr.hist.orient = 0;

            /* up to 3 characters for the voxels units label
ie mm., um, cm. */

            strcpy(hdr.dime.vox_units,"calculated image");

            /* up to 7 characters for the calibration units
label; ie HU */

            strcpy(hdr.dime.cal_units," ");

            /* Calibration maximum and minimum values;
               values of 0.0 for both fields imply that no
               calibration max and min values are used */

            hdr.dime.cal_max = 0.0;
            hdr.dime.cal_min = 0.0;

            fwrite(&hdr,sizeof(struct dsr),1,fp);
            fclose(fp);

            }

/*************************************************************
***************/

void jacobi(float **a, int n, float d[], float **v)

/*  Numerical recipes program to calculate the eigenvalues and
eigenvectors of the matrix a, outputting the eigenvalues in
the column vector d and the eigenvector components in the
matrix v */

{
            int j,iq,ip,i;
            float tresh,theta,tau,t,sm,s,h,g,c,*b,*z;

            b=vector(1,n);
            z=vector(1,n);
            for (ip=1;ip<=n;ip++) {
                        for (iq=1;iq<=n;iq++) v[ip][iq]=0.0;
                        v[ip][ip]=1.0;
            }
            for (ip=1;ip<=n;ip++) {
                        b[ip]=d[ip]=a[ip][ip];
                        z[ip]=0.0;
            }
            for (i=1;i<=50;i++) {
                        sm=0.0;
                        for (ip=1;ip<=n-1;ip++) {
                                    for (iq=ip+1;iq<=n;iq++)
                                                sm +=
fabs(a[ip][iq]);
                        }
                        if (sm == 0.0) {
                                    free_vector(z,1,n);
                                    free_vector(b,1,n);
                                    return;
                        }
                        if (i < 4)
                                    tresh=0.2*sm/(n*n);
                        else
                                    tresh=0.0;
                        for (ip=1;ip<=n-1;ip++) {
                                    for (iq=ip+1;iq<=n;iq++) {

            g=100.0*fabs(a[ip][iq]);
                                                if (i > 4 &&
(float)(fabs(d[ip])+g) == (float)fabs(d[ip])
                                                                &&
(float)(fabs(d[iq])+g) == (float)fabs(d[iq]))

            a[ip][iq]=0.0;
                                                else if
(fabs(a[ip][iq]) > tresh) {

            h=d[iq]-d[ip];
                                                                if
((float)(fabs(h)+g) == (float)fabs(h))

            t=(a[ip][iq])/h;
                                                else {

            theta=0.5*h/(a[ip][iq]);

            t=1.0/(fabs(theta)+sqrt(1.0+theta*theta));

            if (theta < 0.0) t = -t;
                                                                }

            c=1.0/sqrt(1+t*t);

            s=t*c;

            tau=s/(1.0+c);

            h=t*a[ip][iq];

            z[ip] -= h;

            z[iq] += h;

            d[ip] -= h;

            d[iq] += h;

            a[ip][iq]=0.0;

            for (j=1;j<=ip-1;j++) {

            ROTATE(a,j,ip,j,iq)
                                                                }

            for (j=ip+1;j<=iq-1;j++) {

            ROTATE(a,ip,j,j,iq)
                                                                }

            for (j=iq+1;j<=n;j++) {

            ROTATE(a,ip,j,iq,j)
                                                                }

            for (j=1;j<=n;j++) {

            ROTATE(v,j,ip,j,iq)
                                                                }
                                                }
                                    }
                        }
                        for (ip=1;ip<=n;ip++) {
                                    b[ip] += z[ip];
                                    d[ip]=b[ip];
                                    z[ip]=0.0;
                        }
            }
            nrerror("Too many iterations in routine jacobi");
}

            #undef ROTATE
            #undef NRANSI

/*************************************************************
***************/
```

```
void eigsrt(float d[], float **v, int n)

/* Numerical recipes routine for sorting eigenvalues and
eigenvectors into descending numerical order. d is the
eigenvalue column matrix and v is the matrix of eigenvectors
*/

{
            int k,j,i;
            float p;

            for (i=1;i<n;i++) {
                        p=d[k=i];
                        for (j=i+1;j<=n;j++)
                                    if (d[j] >= p) p=d[k=j];
                        if (k != i) {
                                    d[k]=d[i];
                                    d[i]=p;
                                    for (j=1;j<=n;j++) {
                                                p=v[j][i];

            v[j][i]=v[j][k];
                                                v[j][k]=p;
                                    }
                        }
            }
}

/*************************************************************
***************/

float *vector(int nl, int nh)

/* allocate a float vector with subscript range v[nl..nh] */

{
            float *v;

            v=(float *)malloc((size_t) ((nh-
nl+1+NR_END)*sizeof(float)));
            if (!v) nrerror("allocation failure in vector()");
            return v-nl+NR_END;
}

/*************************************************************
***************/

void free_vector(float *v, int nl, int nh)

/* free a float vector allocated with vector() */
{
            free((FREE_ARG) (v+nl-NR_END));
}

/*************************************************************
***************/

void lubksb(float **a, int n, int *indx, float b[])
{
            int i,ii=0,ip,j;
            float sum;

            for (i=1;i<=n;i++) {
                        ip=indx[i];
                        sum=b[ip];
                        b[ip]=b[i];
                        if (ii)
                                    for (j=ii;j<=i-1;j++) sum
-= a[i][j]*b[j];
                        else if (sum) ii=i;
                        b[i]=sum;
            }
            for (i=n;i>=1;i--) {
                        sum=b[i];
                        for (j=i+1;j<=n;j++) sum -=
a[i][j]*b[j];
                        b[i]=sum/a[i][i];
            }
}

/*************************************************************
***************/

void ludcmp(float **a, int n, int *indx, float *d)
{
            int i,imax,j,k;
            float big,dum,sum,temp;
            float *vv;

            vv=vector(1,n);
            *d=1.0;
            for (i=1;i<=n;i++) {
                        big=0.0;
                        for (j=1;j<=n;j++)
                                    if ((temp=fabs(a[i][j])) >
big) big=temp;
                        if (big == 0.0) {
                         nrerror("Singular matrix in routine
ludcmp");
                        }
                        vv[i]=1.0/big;
            }
            for (j=1;j<=n;j++) {
                        for (i=1;i<j;i++) {
                                    sum=a[i][j];
                                    for (k=1;k<i;k++) sum -=
a[i][k]*a[k][j];
                                    a[i][j]=sum;
                        }
                        big=0.0;
                        for (i=j;i<=n;i++) {
                                    sum=a[i][j];
                                    for (k=1;k<j;k++)
                                                sum -=
a[i][k]*a[k][j];
                                    a[i][j]=sum;
                                    if ( (dum=vv[i]*fabs(sum))
>= big) {
                                                big=dum;
                                                imax=i;
                                    }
                        }
                        if (j != imax) {
                                    for (k=1;k<=n;k++) {

            dum=a[imax][k];
            a[imax][k]=a[j][k];
                                                a[j][k]=dum;
                                    }
                                    *d = -(*d);
                                    vv[imax]=vv[j];
                        }
                        indx[j]=imax;
                        if (a[j][j] == 0.0) a[j][j]=TINY;
                        if (j != n) {
                                    dum=1.0/(a[j][j]);
                                    for (i=j+1;i<=n;i++)
a[i][j] *= dum;
                        }
            }
            free_vector(vv,1,n);
}
#undef TINY
#undef NRANSI

/*************************************************************
***************/

float   inverse(float **matrix, float **inverse, int N)

/*  Calculates the inverse of a matrix */

{
            int i,j,*indx;
            float d, *col;

            indx=ivector(1,N);
            col=vector(1,N);

            ludcmp(matrix,N,indx,&d);
            for(j=1;j<=N;j++) {
              for(i=1;i<=N;i++)  col[i]=0.0;
              col[j]=1.0;
              lubksb(matrix,N,indx,col);
              for(i=1;i<=N;i++) inverse[i][j]=col[i];

            free_ivector(indx,1,N);
            free_vector(col,1,N);
            }
}

/*************************************************************
***************/

float   transpose(float **matrix, float **transpose, int r,
int c)

/*  Calculates the transpose of the given matrix */

{
            int i,j;

            for(i=1; i<=r; i++)
            {
            for(j=1; j<=c; j++)
            {
            transpose[j][i] = matrix[i][j];
            }
            }

}

/*************************************************************
***************/

float   multmat(float **a, float **b, float **c, int numrows1,
int numcols1, int numcols2)

/* Multiplies matrix a[i,j] by b[j,k] and returns the product
in c */

{
            int row1,col1,row2,col2;

            for(row1=1; row1<=numrows1; row1++) {
              for(col2=1; col2<=numcols2; col2++) {

                 /* Initialise output matrix */

                 c[row1][col2] = 0;

                 for(col1=1; col1<=numcols1; col1++){
                 c[row1][col2] += a[row1][col1] *
b[col1][col2];
                 }
              }
            }
}

/*************************************************************
***************/

float   multmatvec(float **a, float *b, float *c, int
nummrows, int nummcols)

/* Multiplies matrix a[i,j] by b[j] and returns the product in
c */

{
            int mrow,mcol, vrow, numvrows;

            numvrows = nummcols;

            for(mrow=1; mrow<=nummrows; mrow++) {

                /* Initialise output matrix */

                c[mrow] = 0;

                for(vrow=1, mcol=1; vrow<=numvrows; vrow++,
mcol++) {
                c[mrow] += a[mrow][mcol] * b[vrow];
                }
            }
}

/*************************************************************
***************/

int *ivector(int nl, int nh)

/* allocate an int vector with subscript range v[nl..nh] */
```

```
{

            int *v;

            v=(int *)malloc((size_t) ((nh-
nl+1+NR_END)*sizeof(int)));
            if (!v) nrerror("allocation failure in
ivector()");
            return v-nl+NR_END;

}

/*************************************************************
***************/

void free_ivector(int *v, int nl, int nh)

/* free an int vector allocated with ivector() */

{

            free((FREE_ARG) (v+nl-NR_END));
}

/*************************************************************
***************/

void make_dir(char *dir_name)
{
            strcpy(dir_make,"mkdir ");
            strcat(dir_make,in_path);
            strcat(dir_make,st_name);
            strcat(dir_make,slash);
            strcat(dir_make,s);
            strcat(dir_make,im_fdr);
            strcat(dir_make,slash);
            strcat(dir_make,dir_name);

            system(dir_make);
}

/*************************************************************
***************/

float **matrix(int nrl, int nrh, int ncl, int nch)

/* allocate a float matrix with subscript range
m[nrl..nrh][ncl..nch] */

{
            int i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
            float **m;

            /* allocate pointers to rows */
            m=(float **)
malloc((size_t)((nrow+NR_END)*sizeof(float*)));
            if (!m) nrerror("allocation failure 1 in
matrix()");
            m += NR_END;
            m -= nrl;

            /* allocate rows and set pointers to them */
            m[nrl]=(float *)
malloc((size_t)((nrow*ncol+NR_END)*sizeof(float)));
            if (!m[nrl]) nrerror("allocation failure 2 in
matrix()");
            m[nrl] += NR_END;
            m[nrl] -= ncl;

            for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

            /* return pointer to array of pointers to rows */
            return m;
}

/*************************************************************
***************/

float ***f3tensor(int nrl, int nrh, int ncl, int nch, int ndl,
int ndh)
/* allocate a float 3tensor with range
t[nrl..nrh][ncl..nch][ndl..ndh] */
{
            long i,j,nrow=nrh-nrl+1,ncol=nch-ncl+1,ndep=ndh-
ndl+1;
            float ***t;

            /* allocate pointers to pointers to rows */
            t=(float ***)
malloc((size_t)((nrow+NR_END)*sizeof(float**)));
            if (!t) nrerror("allocation failure 1 in
f3tensor()");
            t += NR_END;
            t -= nrl;

            /* allocate pointers to rows and set pointers to
them */
            t[nrl]=(float **)
malloc((size_t)((nrow*ncol+NR_END)*sizeof(float*)));
            if (!t[nrl]) nrerror("allocation failure 2 in
f3tensor()");
            t[nrl] += NR_END;
            t[nrl] -= ncl;

            /* allocate rows and set pointers to them */
            t[nrl][ncl]=(float *)
malloc((size_t)((nrow*ncol*ndep+NR_END)*sizeof(float)));
            if (!t[nrl][ncl]) nrerror("allocation failure 3 in
f3tensor()");
            t[nrl][ncl] += NR_END;
            t[nrl][ncl] -= ndl;

            for(j=ncl+1;j<=nch;j++) t[nrl][j]=t[nrl][j-
1]+ndep;
            for(i=nrl+1;i<=nrh;i++) {
                        t[i]=t[i-1]+ncol;
                        t[i][ncl]=t[i-1][ncl]+ncol*ndep;
                        for(j=ncl+1;j<=nch;j++)
t[i][j]=t[i][j-1]+ndep;
            }

            /* return pointer to array of pointers to rows */
            return t;
}

/*************************************************************
***************/

void free_matrix(float **m, int nrl, int nrh, int ncl, int
nch)

/* free a float matrix allocated with matrix() */

{
            free((FREE_ARG) (m[nrl]+ncl-NR_END));
            free((FREE_ARG) (m+nrl-NR_END));
}

/*************************************************************
***************/

void free_f3tensor(float ***t, int nrl, int nrh, int ncl, int
nch,
            int ndl, int ndh)
/* free a float f3tensor allocated by f3tensor() */
{
            free((FREE_ARG) (t[nrl][ncl]+ndl-NR_END));
            free((FREE_ARG) (t[nrl]+ncl-NR_END));
            free((FREE_ARG) (t+nrl-NR_END));
}

/*************************************************************
***************/

void moment(float data[], int n, float *ave, float *adev,
float *sdev, float *var, float *skew, float *curt)

/* Given an array of data[1..n], this routine returns its mean
ave, average */
/* deviation adev, standard deviation sdev, variance var,
skewness skew and */
/* kurtosis curt. */

{
            void nrerror(char error_text[]);
            int j;
            float ep = 0.0,s,p;

            if(n<=1) nrerror("n must be at least 2 in
moment");
            s=0.0;
            for(j=1;j<=n;j++) s+=data[j];
            *ave=s/n;
            *adev=(*var)=(*skew)=(*curt)=0.0;
            for(j=1;j<=n;j++) {
              *adev += fabs(s=data[j]-(*ave));
              *var += (p=s*s);
              *skew += (p*=s);
              *curt += (p*=s);
            }
            *adev /= n;
            *var=(*var-ep*ep/n)/(n-1);
            *sdev=sqrt(*var);
            if(*var){
              *skew /= (n*(*var)*(*sdev));
              *curt=(*curt)/(n*(*var)*(*var))-3.0;
            } else nrerror("No skew/kurtosis when variance = 0
(in moment)");
}

/*************************************************************
***************/
```

A2

List of Publications

A2.1 Papers

P.A. Armitage, M.E. Bastin, I. Marshall, J.M. Wardlaw, J. Cannon. Diffusion anisotropy measurements in ischaemic stroke of the human brain. MAGMA 1998;6(1):28-36.

M.E. Bastin, **P.A. Armitage**, I. Marshall. A theoretical study of the effect of experimental noise on the measurement of anisotropy in diffusion imaging. Mag Reson Imaging 1998;16(7):773-785.

Macmillan CSA, Wild JW, Andrews PJD, Marshall I, **Armitage PA**, Wardlaw JM, Easton VJ, Cannon J. Accuracy of a miniature intracranial pressure monitor, its function during MR scanning and assessment of image artefact generation. Neurosurgery 1999;45(1):188-93.

A2.2 Conference Proceedings

P.A. Armitage, M.E. Bastin, I. Marshall. Selecting an appropriate anisotropy index for displaying diffusion data with improved contrast and sensitivity. MAGMA 1999; 8 (suppl. 1): 105.

P.A. Armitage, M.E. Bastin, I. Marshall. Optimisation of two-point diffusion-weighting strategies for diffusion tensor imaging of the human brain. MAGMA 1999; 8 (suppl 1): 179.

P.A. Armitage. Quantifying the likely errors arising when using the orthogonal or tetrahedral encoding schemes to sample diffusion anisotropy. In : Book of abstracts: Seventh Annual Meeting of the International Society for Magnetic Resonance in Medicine, Berkeley, CA: ISMRM 1999:1797.

C.S.A. Macmillan, J.M. Wild, J.M. Wardlaw, P.J.D. Andrews, I. Marshall, V.J. Easton, **P.A. Armitage**. Neuronal injury demonstrated by early quantitative magnetic resonance spectroscopy following acute brain injury. In : Book of abstracts: Seventh Annual Meeting of the International Society for Magnetic Resonance in Medicine, Berkeley, CA: ISMRM 1999:224.

P.A. Armitage. A theoretical comparison of the orthogonal, tetrahedral and tensor encoding schemes for diffusion anisotropy imaging. MAGMA 1998;6 (suppl. 1):117.

P.A. Armitage. A prolateness index for further characterising diffusion anisotropy. MAGMA 1998;6 (suppl. 1):119.

M.E. Bastin, **P.A. Armitage**, I. Marshall. The signal-to-noise ratio in diffusion imaging. In: Book of abstracts: Fourth Annual National Conference of the Institute of Physics and Engineering in Medicine: IPEM 1998:186.

M.E. Bastin, **P.A. Armitage**, I. Marshall. The effect of experimental noise on the calculation of diffusion anisotropy in diffusion imaging. Cerebrovascular Disease 1998;8 (suppl. 4):74.

C.S.A. Macmillan, J.M. Wild, **P.A. Armitage**, J.M. Wardlaw, M.E. Bastin, I. Marshall, J. Cannon, P.J. Andrews. Value of combined proton MRS and diffusion imaging following traumatic brain injury. Anaesthesia 2000 1998;1: 95-96.

M.E. Bastin, **P.A. Armitage**, I. Marshall. The effect of experimental noise on the calculation of anisotropy in diffusion tensor and diffusion weighted imaging. In: Book of abstracts: Sixth Annual Meeting of the International Society for Magnetic Resonance in Medicine, Vol. 2. Berkeley, CA: ISMRM 1998;1220.

P.A. Armitage, M.E. Bastin, I. Marshall, J.M. Wardlaw, J. Cannon. Effects of image registration on diffusion anisotropy measurements in ischaemic stroke. In: Book of abstracts: Sixth Annual Meeting of the International Society for Magnetic Resonance in Medicine, Vol. 2. Berkeley, CA: ISMRM 1998;1237.

P.A. Armitage, I. Marshall, J.M. Wardlaw, J. Cannon. Diffusion anisotropy in ischaemic stroke. MAGMA 1997;5(2):159.

M.E. Bastin, **P.A. Armitage**, I. Marshall. Estimation of magnet uniformity for diffusion tensor imaging. In: Book of abstracts: Third Annual National Conference of the Institute of Physics and Engineering in Medicine: IPEM 1997:193.

J.M. Wardlaw, I. Marshall, M.S. Dennis, **P.A. Armitage**, C. Counsell. Diffusion-weighted MRI is useful for demonstrating small infarcts not visible by other means. Cerebrovascular Diseases 1997;7:35.

P.A. Armitage, M.E. Bastin, I. Marshall. Anisotropic diffusion-weighted imaging of ischaemic stroke. In: Book of abstracts: Third Annual National Conference of the Institute of Physics and Engineering in Medicine: IPEM 1997:152.

Abbreviations

ADC	-	Apparent Diffusion Coefficient
a-p	-	Anterior to Posterior
a.u.	-	Arbitrary Units
BOLD	-	Blood Oxygen Level Dependent
CNS	-	Central Nervous System
CPMG	-	Carr-Purcell-Meiboom-Gill
CRI	-	Clinical Research Initiative
CT	-	Computed Tomography
CSF	-	Cerebrospinal Fluid
DAI	-	Diffusion Anisotropy Indices
DICOM	-	Digital Imaging and Communications in Medicine
DNR	-	Diffusion to Noise Ratio
DTI	-	Diffusion Tensor Imaging
DW	-	Diffusion-Weighted
DWI	-	Diffusion-Weighted Imaging
DWIs	-	Diffusion-Weighted Images
EMF	-	Electromotive Force
EPI	-	Echo Planar Imaging

FFT	-	Fast Fourier Transform
FID	-	Free Induction Decay
FLAIR	-	Fluid Attenuated Inversion Recovery
FLASH	-	Fast Low Angle Shot
fMRI	-	Functional Magnetic Resonance Imaging
FONAR	-	Field Focused Nuclear Magnetic Resonance
FOV	-	Field Of View
FSE	-	Fast Spin Echo
FWHM	-	Full Width at Half Maximum
Gd-DTPA	-	Gadolinium Diethylenetriamine Pentaacetic Acid
GRASE	-	Gradient And Spin Echo
h-f	-	Head to Feet
ICP	-	Intracranial Pressure
IDL	-	Iterative Data Language
ITU	-	Intensive Care Unit
IVIM	-	Intravoxel Incoherent Motion
LSR	-	Lothian Stroke Register
LU	-	Lower-Upper (triangular matrices)
l-r	-	Left to Right
MR	-	Magnetic Resonance
MRC	-	Medical Research Council
MRI	-	Magnetic Resonance Imaging
MS	-	Multiple Sclerosis
NAA	-	N-Acetylaspartate
NMR	-	Nuclear Magnetic Resonance
PACI	-	Partial Anterior Circulation Infarct
PCA	-	Phase Contrast Angiography
PET	-	Positron Emission Tomography
rCBF	-	Regional Cerebral Blood Flow
rCBV	-	Regional Cerebral Blood Volume
RF	-	Radio Frequency

RGB	-	Red-Green-Blue
RI	-	Rotationally Invariant
ROI	-	Region Of Interest
RV	-	Rotationally Variant
SAH	-	Subarachnoid Haemorrhage
SAR	-	Specific Absorption Rate
SHEFC	-	Scottish Higher Education Funding Council
SNR	-	Signal-to-Noise Ratio
SPECT	-	Single Photon Positron Emission Tomography
SPM	-	Statistical Parametric Mapping
STEAM	-	Stimulated Echo Acquisition Mode
s/n	-	Stroke to Normal Ratio
TBI	-	Traumatic Brain Injury
TCD	-	Transcranial Doppler Ultrasound
TOF	-	Time Of Flight
TSO	-	Time from Stroke Onset
VOI	-	Voxel Of Interest
WGH	-	Western General Hospital
2DFT	-	Two Dimensional Fourier Transform

Acknowledgements

I would like to thank my supervisor Dr Ian Marshall (Senior Lecturer, Dept. of Medical Physics) for providing support during the course of this work and particularly for his efforts with the development of the diffusion imaging sequences and post-processing software on the Siemens scanner, which provided the basis for much of the initial work. I would also like to thank Dr Mark Bastin (Lecturer, Dept. of Medical Physics) for his close collaboration and constructive criticism, primarily regarding the work performed in the later stages of the project. Much of the clinical impetus for the project has come courtesy of Dr Joanna Wardlaw (Research Neuroradiologist, Dept. of Clinical Neurosciences), to whom I am extremely grateful for enabling the technical developments to be carried through and tested in the clinical setting.

I would also like to thank the other members of the research group who have provided much needed advice at times of distress, particularly; Jim Wild for his useful advice both before and after his move to colder territories; Uwe Koehler for his extremely useful computing tips; Malcolm Robertson for his multi-lingual resources; and finally Jim Cannon and latterly Evelyn Cowie for organising and scanning the patients. I am also grateful to the secretarial staff, particularly Irene Craig (Medical Physics) and Ann Deary (Clinical Neurosciences), for their much appreciated assistance.

I would also like to acknowledge the help and support of the following people and groups:

Prof. Norman McDicken: For providing advice and conference funding.

Martin Connell: For comprehensive UNIX computing support.

Prof. Charles Warlow: For providing financial assistance in attending conferences.

Dr Robin Sellar: For allowing access to the Siemens clinical scanner and for providing conference funding.

Dr Carol MacMillan: For enthusiastically recruiting and organising the head injury patient study.

Dr Peter Hoskins: For providing advice throughout the course of this work as part of the departments 'Mentor' system.

David Perry: For providing PC computing support.

David Signorini, *Val Easton* and *Steff Lewis*: for valuable statistical advice.

Medical Illustration: For providing high quality posters and slides.

I would like to express my gratitude to the UK Medical Research Council for funding this work and the SHEFC Brain Imaging Research Centre for Scotland for providing access to the Elscint scanner and associated facilities.

I would like to thank my family; particularly my mother and father for supporting me in all my endeavours. Finally, I would like to thank the many friends who have contributed to making my time in Edinburgh (and elsewhere) so enjoyable; particularly Alison, Andy, Bec, Bernard, Caroline, Chris, Graham, Henry, Ingo, Jeff, Jo, Jon, Karen, Katie, Michelle, Mike, Paul, Rob, Steph, Steve, Tom, Vicky, all the staff in DCN X-ray, the football boys and 'Maggies' regulars.

Declaration

I hereby declare that this thesis has been written, and the work performed, by myself. Where work is published that has been performed by other people, either associated with the research group or elsewhere, an appropriate acknowledgement is made in the text.

References

1. Bloch F, Hansen WW, Packard ME. Phys Rev 1946;69:127.
2. Purcell EM, Torrey HC, Pound RV. Phys Rev 1946;69:37.
3. Proctor WG, Yu FC. Phys Rev 1950;77:717.
4. Dickinson WC. Phys Rev 1950;77:736.
5. Hahn EL. Spin echoes. Phys Rev 1950;80:580-594.
6. Lauterbur PC. Image formation by induced local interaction: examples employing nuclear magnetic resonance zeugmatography. Nature 1973;243:190-191.
7. Mansfield P, Grannell PK. NMR 'diffraction' in solids? J Phys C 1973;6:L422-L426.
8. Damadian R, Goldsmith M, Minkoff L. Physiol Chem Phys 1977;9:97.
9. Mansfield P, Pykett IL, Morris PG, Coupland RE. Br J Radiol 1978;51:921.
10. EdelsteinWA, Hutchinson JMS, Johnson G, Redpath TW. Spin-warp NMR imaging and applications of human whole body imaging. Phys Med Biol 1980;25:751-756.
11. Kumar A, Welti D, Ernst R. NMR Fourier zeugmatography. J Magn Reson 1975;18:69-83.
12. Mansfield P. Multi-planar image formation using NMR spin echoes. J Phys C 1977;10:L55-L58.
13. Ordidige RJ, Coxon R, Howseman A, Chapman B, Turner R, Stehling M, Mansfield P. Snapshot head imaging at 0.5T using the echo planar technique. Magn Reson Med 1988;8:110-115.
14. Carr HY, Purcell EM. Effects of diffusion on free precession in nuclear magnetic resonance experiments. Phys Rev 1954;94:630-638.
15. Torrey HC. Bloch equations with diffusion terms. Phys Rev 1956;104:563-565.
16. Stejskal EO, Tanner JE. Spin diffusion measurements: spin echoes in the presence of a time-dependent field gradient. J Chem Phys 1965;42:288-292.

17. Taylor DG, Bushell MC. The spatial mapping of translational diffusion coefficients by the NMR imaging technique. Phys Med Biol 1985;30:345-349.
18. Merboldt KD, Hanicke W, Frahm J. Self-diffusion NMR imaging using stimulated echoes. J Magn Reson 1985;64:479-486.
19. Le Bihan D, Breton E, Lallemand D, Grenier P, Cabanis E, Laval-Jeantet M. MR imaging of intravoxel incoherent motions: application to diffusion and perfusion in neurologic disorders. Radiology 1986;161:401-407.
20. Turner R, Le Bihan D. Single-shot diffusion imaging at 2.0 Tesla. J Magn Reson 1990;86:445-452.
21. van Gelderen P, de Vleeschouwer MH, DesPres D, Pekar J, van Zijl PC, Moonen CT. Water diffusion and acute stroke. Magn Reson Med 1994;31:154-163.
22. Weishmann UC, Clark CA, Symms MR, Barker GJ, Birnie KD, Shorvon SD. Water diffusion in the human hippocampus in epilepsy. Magn Reson Imaging 1999;17:29-36.
23. Tsuruda JS, Chew WM, Moseley ME, Norman D. Diffusion-weighted MR imaging of the brain: value of differentiating between extraaxial cysts and epidermoid tumors. AJNR 1990;11:925-931.
24. Basser PJ, Mattiello J, Le Bihan D. Estimation of the effective self-diffusion tensor from the NMR spin echo. J Magn Reson B 1994;103:247-254.
25. Basser PJ, Mattiello J, Le Bihan D. MR diffusion tensor spectroscopy and imaging. Biophys J 1994;66:259-267.
26. Lim KO, Hedehus M, Moseley M, de Crespigny A, Sullivan EV, Pfefferbaum A. Compromised white matter tract integrity in schizophrenia inferrd from diffusion tensor imaging. Arch Gen Psychiatry 1999;56:367-374.
27. Le Bihan D, Turner R, Douek P, Patronas N. Diffusion MR imaging: clinical applications. AJR 1992;159:591-599.
28. Hounsfield GN. Computerised transverse axial scanning (tomography) I. description of system. Br J Radiol 1973;46:1023-1047.
29. Morris PG. Nuclear magnetic resonance imaging in medicine and biology. Oxford: Clarendon Press;1986. p. 41-50.
30. Bryan RN, Levy LM, Whitlow WD, Killian JM, Prezios TJ, Rosario JA. Diagnosis of acute cerebral infarction: comparison of CT and MRI. AJNR 1991;12:611-620.
31. Kuhl DE, Edwards RQ. Image separation of radio-isotope scanning. Radiology 1963;80:653-662.
32. Ter-Pogossian MM, Phelps ME, Hoffman EJ. A positron emission transaxial tomograph for nuclear medicine imaging (PET). Radiology 1975;114:89-98.
33. Aaslid R, Markwalder TH, Nomes H. Non-invasive trans-cranial doppler ultrasound recording flow velocity in basal cerebral arteries. J Neurosurg 1982;57:769-774.
34. Nishimura DG. Time-of-flight angiography. Magn Reson Med 1990;14:194-201.
35. Dumoulin CL, Souza SP, Walker MF, Wagle W. Three-dimensional phase contrast angiography. Magn Reson Med 1989;9:139-149.
36. Hausemann R, Lewin JS, Laub G. Phase-contrast MR angiography with reduced acquisition time: new concepts in sequence design. J Magn Reson Imaging 1991;1:399-404.

37. Le Bihan D, Turner R. The capillary network: a link between IVIM and classical perfusion. Magn Reson Med 1992;27:171-178.
38. Rosen BR, Belliveau JW et al. Perfusion imaging with NMR contrast agents. Magn Reson Med 1990;14:249-265.
39. Boxerman JL, Hamberg LM, Rosen BR, Weisskoff RM. MR contrast due to intravascular magnetic susceptibility perturbations. Magn Reson Med 1995;34:555-566.
40. Schmitt F, Stehling MK, Turner R. Echo-planar imaging. Berlin: Springer-Verlag;1998. p. 423-434.
41. Detre JA, Williams DS, Leigh JS et al. Perfusion Imaging. Magn Reson Med 1992;23:37-45.
42. Williams DS, Detre JA, Leigh JS et al. Magnetic resonance imaging of perfusion using spin inversion of arterial water. Proc Natl Acad Sci USA 1992;89:212-216.
43. Pauling L, Coryell C. The magnetic properties and structure of hemoglobin. Proc Natl Acad Sci USA 1936;22:210-216.
44. Turner R, Le Bihan D, Moonen CTW, DesPres D, Frank J. Echo planar time course MRI of cat brain oxygenation changes. Magn Reson Med 1991;22:159-166.
45. Sokoloff L, Reinich M, Kumal C. The 14-C deoxyglucose method for the measurement of cerebral glucose utilisation. Theory procedure and normal values in the concious and anaesthetised rat. Neurochem 1977;28:897-916.
46. Broich K, Alavi A, Kushner M. Positron emission tomography in cerebrovascular disorders. Seminars in Nuclear Medicine 1992;4:224-232.
47. Patt SL, Sykes BD. T1 water eliminated fourier transform NMR spectroscopy. Chem Phys 1972;56:3182-3184.
48. Hore PJ. Solvent suppression in fourier transform NMR. J Magn Reson 1983;55:283-300.
49. Brown TR, Kincaid BM, Ugurbil D. NMR chemical shift imaging in three dimensions. Natl Acad Sci, USA 1982;79:3523-3526.
50. Maudlsey AA, Hilal SK, Perman WH, Simon HE. Spatially resolved high resolution spectroscopy by four dimensional NMR. J Magn Reson 1983;51:147-152.
51. Sotak CH, Freeman DM. A method for volume localised lactate editing using zero quantum coherence created in a stimulated echo pulse sequence. J Magn Reson 1988;77:382-388.
52. Abragam A. The principles of nuclear magnetism. Oxford: Oxford University Press;1960.
53. Farrar TC, Becker ED. Pulse and Fourier transform NMR. New York: Academic Press; 1971.
54. Meiboom S, Gill D. Rev Scient Instrum 1958;29:688-
55. Schmitt F, Warach S, Wielopolski P, Edelman RR. Clinical applications and techniques of echo-planar imaging. MAGMA 1994;2:259-266.
56. Howseman AM, Stehling MK, Chapman B, Coxon R, Turner R et al. Improvements in snap-shot nuclear magnetic resonance imaging. Br J Radiol 1988;61:822-888.
57. Pykett I, Rzedzian R. Instant images of the body by magnetic resonance. Magn Reson Med 1987;5:563-571.

58. Margosian P, Schmitt F, Purdy D. Faster MR imaging: imaging with half the data. Health Care Instrum 1986;1:195-197.
59. Haacke EM, Lindskog ED, Lin W. Partial-Fourier imaging. A fast, iterative POCS technique capable of local phase recovery. J Magn Reson 1991;92:126-145.
60. Meyer C, Hu B, Nishimura D, Macovski A. Fast spiral coronary artery imaging. Magn Reson Med 1992;28:202-213.
61. Meyer C, Macovski A. Square spiral fast imaging: interleaving and off-resonance effects. In: Proceedings of the Society of Magnetic Resonance in Medicine, 1987. p. 230.
62. Jehensen P, Westphal M, Schuff N. Analytical method for the compensation of eddy-current effects induced by pulsed magnetic field gradients in NMR systems. J Magn Reson 1990;90:264-278.
63. Turner R, Bowley RM. Passive screening of switched magnetic field gradients. J Phys E (Sci Instr) 1986;19(10):876-879.
64. Cory DG. Measurement of translational displacement probabilities by NMR: an indicator of compartmentation. Magn Reson Med 1990;14:435-444.
65. Eisberg RM, Lerner LS. Physics: foundations and applications. Kogakusha: McGraw-Hill;1982. p. 837.
66. Schmitt F, Stehling MK, Turner R. Echo-planar imaging. Berlin: Springer-Verlag;1998. p. 311.
67. Mills R. Self diffusion of normal and heavy water in the range 1 - 45°. J Phys Chem 1973;77:685-688.
68. Atkins PW. Physical chemistry. Oxford: University Press;1986. p. 678.
69. Le Bihan D. Molecular diffusion nuclear magnetic resonance imaging. Magn Reson Quarterly 1991;7:1-30.
70. Jost W. Diffusion in solids, liquids, gases. New York: Academic Press; 1960.
71. Moseley ME, Cohen Y, Kucharczyk J, Mintorovitch J, Asgari HS, Wendland MF, Tsuruda J, Norman D. Diffusion-weighted MR imaging of anisotropic water diffusion in the cat central nervous system. Radiology 1990;176:439-445.
72. Turner R, Le Bihan D, Maier J, Vavrek R, Hedges LK, Pekar J. Echo-planar imaging of intravoxel incoherent motions. Radiology 1990;177:407-414.
73. Chenevert TL, Brunberg JA, Pipe JG. Anisotropic diffusion within human white matter: demonstration with NMR techniques in vivo. Radiology 1990;177:401-405.
74. Cleveland GG, Chang DC, Hazelwood CF. Nuclear magnetic resonance measurement of skeletal muscle: anisotropy of the diffusion coefficient of the intracellular water. Biophys J 1976;16:1043-1053.
75. Le Bihan D, Turner R, Douek P, Patronas N. Diffusion MR imaging: clinical applications. AJR 1992;159:591-599.
76. Stejskal EO. Use of spin echoes in a pulsed magnetic-field gradient to study anisotropic restricted diffusion and flow. J Chem Phys 1965;43:3597-3603.
77. Onsager L. Phys Rev 1931;37:405.
78. Onsager L. Phys Rev 1931;38:2265.
79. Goldman M. Quantum description of high-resolution NMR in liquids. Clarendon Press;1988. p. 12-15.

80. Price WS, Kuchel PW. Effect of nonrectangular field gradient pulses in the Stejskal and Tanner (diffusion) pulse sequence. J Magn Reson 1991;94:133-139.
81. Mattiello J, Basser PJ, Le Bihan D. Analytical expressions for the *b* matrix in NMR diffusion imaging and spectroscopy. J Magn Reson 1994;A108:131-141.
82. Mattiello J, Basser PJ, Le Bihan D. The *b* matrix in diffusion tensor echo-planar imaging. Magn Reson Med 1997;37:292-300.
83. Basser PJ, Pierpaoli C. A simplified method to measure the diffusion tensor from seven MR images. Magn Reson Med 1998;39:928-934.
84. Basser PJ, Pierpaoli C. Microstructural and physiological features of tissues elucidated by quantitative-diffusion-tensor MRI. J Magn Reson 1996;B111:209-219.
85. Conturo TE, McKinstry RC, Akbudak E, Robinson BH. Encoding of anisotropic diffusion with tetrahedral gradients: a general mathematical diffusion formalism and experimental results. Magn Reson Med 1996;35:399-412.
86. Basser PJ. Inferring microstructural features and the physiological state of tissues from diffusion-weighted images. NMR Biomed 1995;8:333-344.
87. Norris DG, Niendorf T, Leibfritz D. Healthy and infarcted brain tissues studied at short diffusion times: The origins of the apparent restriction and the reduction in apparent diffusion coefficient. NMR Biomed 1994;7:304-310.
88. Norris DG, Niendorf T, Hoehn-Berlage M, Hohno K, Schneider EJ, Hainz P, Hropot M, Leibfritz D. Incidence of apparent restricted diffusion in three different models of cerebral infarction. Magn Reson Imaging 1994;12:1175-1182.
89. Ehman RL, Felmlee JP. Adaptive technique for high-definition MR imaging of moving structures. Radiology 1989;173:255-263.
90. Ordidge RJ, Helpern JA, Qing ZX, Knight RA, Nagesh V. Correction of motional artefacts in diffusion-weighted MR images using navigator echoes. Magn Reson Imaging 1994;12:455-460.
91. Anderson AW, Gore JC. Analysis and correction of motion artefacts in diffusion-weighted imaging. Magn Reson Med 1994;32:379-387.
92. de Crespigny AJ, Marks MP, Enzmann DR, Moseley ME. Navigated diffusion imaging of normal and ischemic human brain. Magn Reson Med 1995;33:720-728.
93. Latta P, Jellus V, Budinsky L, Mlynarik V, Tkac I, Luypaert R. Motion artefacts reduction in DWI using navigator echoes: A robust and simple correction scheme. In: Proceedings of the International Society for Magnetic Resonance in Medicine, 1996. p. 1351.
94. Marshall I, Wardlaw JM. Comparison of postprocessing techniques for navigated diffusion weighted imaging. In: Proceedings of the International Society for Magnetic Resonance in Medicine, 1997. p. 1725.
95. Gibbs SJ, Carpenter TA, Hall LD. Diffusion imaging with unshielded gradients. J Magn Reson 1992;98:183-191.
96. Gibbs SJ, Johnson CS. A PFG NMR experiment for accurate diffusion and flow studies in the presence of eddy currents. J Magn Reson 1991;93:395-402.
97. Haselgrove JC, Moore JR. Correction for distortion of echo-planar images used to calculate the apparent diffusion coefficient. Magn Reson Med 1996;36:960-964.

98. Alexander AL, Tsuruda JS, Parker DL. Elimination of eddy current artifacts in diffusion-weighted echo-planar images: The use of bipolar gradients. Magn Reson Med 1997;38:1016-1021.
99. Jezzard P, Barnett AS, Pierpaoli C. Characterization of and correction for eddy current artifacts in echo planar diffusion imaging. Magn Reson Med 1998;39:801-812.
100. Bastin ME. Correction of eddy current induced artefacts in diffusion tensor imaging using iterative cross-correlation. Magn Reson Imaging 1999;17:1011-1024.
101. Kwong K, McKinstry RC, Chien D, Crawley AO, Pearlman JD, Rosen BR. CSF-suppressed quantitative single-shot diffusion imaging. Magn Reson Med 1991;21:157-163.
102. Falconer JC, Narayana PA. Cerebrosipnal fluid-suppressed high-resolution diffusion imaging of human brain. Magn Reson Med 1997;37:119-123.
103. Tanner JE. Use of the stimulated echo in NMR diffusion studies. J Chem Phys 1970;52:2523-2526.
104. Frahm J, Merboldt KD, Hanicke W, Haase A. Stimulated echo imaging. J Magn Reson 1985;64:81-93.
105. Canet D, Diter B, Belmajdoub A, Braondeau J, Boubel JC, Elbayed K. Self-diffusion measurements using a radiofrequency field gradient. J Magn Reson 1989;81:1-12.
106. Karczmar GS, Twieg DB, Lawry TJ, Matson GB, Weiner MW. Detection of motion using B1 gradients. Magn Reson Med 1988;7:111-116.
107. Merboldt KD, Hanicke W, Gyngell ML, Frahm J, Bruhn H. Rapid NMR imaging of molecular self-diffusion using a modified CE-FAST sequence. J Magn Reson 1989;82:115-121.
108. Merboldt KD, Hanicke W, Gyngell ML, Frahm J, Bruhn H. The influence of flow and motion in MRI of diffusion using a modified CE-FAST sequence. Magn Reson Med 1989;12:198-208.
109. Le Bihan D. Intravoxel incoherent motion imaging using steady-state free precession. Magn Reson Med 1988;7:346-351.
110. Merboldt KD, Hanicke W, Bruhn H, Gyngell ML, Frahm J. Diffusion imaging of the human brain in vivo using high speed STEAM MRI. Magn Reson Med 1992;23:179-192.
111. Warach S, Chien D, Li W, Ronthal M, Edelman RR. Fast magnetic resonance diffusion-weighted imaging of acute human stroke. Neurology 1992;42:1717-1723.
112. Liu G, van Gelderen P, Duyn J, Moonen CTW. Single-shot diffusion MRI of human brain on a conventional clinical instrument. Magn Reson Med 1996;35:671-677.
113. Frahm J, Haase A, Matthaei D. Rapid NMR imaging of dynamic processes using the FLASH technique. Magn Reson Med 1986;3:321-327.
114. Lee H, Price RR. Diffusion imaging with the MP-RAGE sequence. J Magn Reson Imaging 1994;4:837-842.
115. Pierpaoli C, Basser PJ. Toward a quantitative assessment of diffusion anisotropy. Magn Reson Med 1996;36:893-906.

116. Le Bihan D, Breton E, Lallemand D, Aubin ML, Vignaud J, Laval-Jeantet M. Separation of diffusion and perfusion in intravoxel incoherent motion MR imaging. Radiology 1988;168:497-505.
117. Neeman M, Freyer JP, Sillerud LO. J Magn Reson 1990;90:303.
118. Cetas TC. Will thermometric tomography become practical for hyperthermia treatment monitoring? Cancer Res 1984;44:4805-4808.
119. Jolesz FA, Bleier AR, Jakab P, Ruenzel PW, Huttl K, Jako GJ. MR imaging of laser tissue interactions. Radiology 1988;168:249-253.
120. Jolesz FA, Higuchi N, Bleier AR, Jakab P. MRI of laser effects on tissue. Magn Reson Imaging 1990;8:156.
121. Gibbs SJ, Carpenter TA, Hall LD. Magnetic resonance imaging of thermal convection. J Magn Reson 1993;A105:209-214.
122. Szafer A, Zhong J, Gore JC. Theoretical model for water diffusion in tissues. Magn Reson Med 1995;33:697-712.
123. Niendorf T, Dijkhuizen RM, Norris DG, van Lookeren Campagne M, Nicolay K. Biexponential diffusion attenuation in various states of brain tissue: implications for diffusion-weighted imaging. Magn Reson Med 1996;36:847-857.
124. van Dusschoten D, Moonen CTW, Adrie de Jager P, van As H. Unravelling diffusion constants in biological tissue by combining Carr-Purcell-Meiboom-Gill imaging and pulsed field gradient NMR. Magn Reson Med 1996;36:907-913.
125. Douek P. MR color mapping of myelin fiber orientation. J Comput Assist Tomogr 1991;15:923-929.
126. Maier SE, Gudbjartsson H, Hsu L, Jolesz FA. Diffusion anisotropy imaging of stroke. In: Proceedings of the International Society for Magnetic Resonance in Medicine, 1997. p. 573.
127. Armitage PA, Bastin ME, Marshall I, Wardlaw JM, Cannon J. Diffusion anisotropy measurements in ischaemic stroke of the human brain. MAGMA 1998;6:28-36.
128. Pierpaoli C, Jezzard P, Basser PJ, Barnett A, Di Chiro G. Diffusion tensor MR imaging of the human brain. Radiology 1996;201:637-648.
129. Bastin ME, Armitage PA, Marshall I. A theoretical study of the effect of experimental noise on the measurement of anisotropy in diffusion imaging. Magn Reson Imaging 1998;16:773-785.
130. Gonzalez RG, Schaefer PW, Buonanno FS, Schwamm LH, Buzik RF, Rordorf G, Wang B, Sorensen AG, Koroshetz WJ. Diffusion-weighted MR imaging: diagnostic accuracy in patients imaged within 6 hours of stroke symptom onset. Radiology 1999;210:155-162.
131. Yamada N, Imakita S, Sakuma T. Value of diffusion-weighted imaging and apparent diffusion coefficient in recent cerebral infarctions: a correlative study with contrast-enhanced T1-weighted imaging. AJNR 1999;20:193-198.
132. Yoneda Y, Tokui K, Hanihara T, Kitagaki H, Tabuchi M, Mori E. Diffusion-weighted magnetic resonance imaging: Detection of ischaemic injury 39 minutes after onset in a stroke patient. An Neurol 1999;45:794-797.
133. van Everdingen KJ, van der Grond J, Kappelle LJ, Ramos LM, Mali WP. Diffusion-weighted magnetic resonance imaging in acute stroke. Stroke 1998;29:1783-1790.

134. Nagesh V, Welch KM, Windham JP, Patel S, Levine SR, Hearshen D, Peck D, Robbins K, D'Olhaberriague L, Soltanian-Zadeh H, Boska MD. Time course of ADCw changes in ischaemic stroke: beyond the human eye! Stroke 1998;29:1783-1790.
135. Schwamm LH, Koroshetz WJ, Sorensen AG, Wang B, Copen WA, Budzik R, Rordorf G, Buonanno FS, Schaefer PW, Gonzalez RG. Time course of lesion development in patients with acute stroke: serial diffusion- and hemodynamic-weighted magnetic resonance imaging. Stroke 1998;29:2268-2276.
136. Singer MB, Chong J, Lu D, Schonewille WJ, Tuhrim S, Atlas SW. Diffusion-weighted MRI in acute subcortical infarction. Stroke 1998;29:133-136.
137. Schlaug G, Siewert B, Benfield A, Edelman RR, Warach S. Time course of apparent diffusion coefficient (ADC) abnormality in human stroke. Neurology 1997;49:113-119.
138. Marks MP, de Crespigny AJ, Lentz D, Enzmann DR, Albers GW, Moseley ME. Acute and chronic stroke: navigated spin-echo diffusion-weighted MR imaging. Radiology 1996;199:403-408.
139. Larsson HB, Thomsen C, Frederiksen J, Stubgaard M, Henriksen O. In vivo magnetic resonance diffusion measurement in the brain of patients with multiple sclerosis. Magn Reson Imaging 1992;10:7-12.
140. Christiansen P, Gideon P, Thomsen C, Stubgaard M, Henriksen O, Larsson HB. Increased water self-diffusion in chronic plaques and in apparently normal white matter in patients with multiple sclerosis. Acta Neurol Scand 1993;87:195-199.
141. Krabbe K, Gideon P, Wagn P, Hansen U, Thomsen C, Madsen F. MR diffusion imaging of intracranial tumours. Neuroradiology 1997;39:483-489.
142. Brunberg JA, Chenevert TL, McKeever PE, Ross DA, Junck LR, Muraszko KM, Dauser R, Pipe JG, Betley AT. In vivo MR determination of water diffusion coefficients and diffusion anisotropy: correlation with structural alteration in gliomas of the cerebral hemispheres. AJNR 1995;16:361-371.
143. Bastin ME, Delgado M, Whittle IR, Cannon J, Wardlaw JM. The use of diffusion tensor imaging in quantifying the effect of dexamethasone on brain tumours. Neuroreport 1999;10:1385-1391.
144. Gideon P, Thomsen C, Gjerris F, Sorensen PS, Henriksen O. Increased self-diffusion of brain water in hydrocephalus measured by MR imaging. Acta Radiol 1994;35:514-519.
145. Buchsbaum MS, Tang CY, Peled S, Gudbjartsson H, Lu D, Hazlett EA, Downhill J, Haznedar M, Fallon JH, Atlas SW. MRI white matter diffusion anisotropy and PET metabolic rate in schizophrenia. Neuroreport 1998;9:425-430.
146. Hoehn-Berlage M, Eis M, Back T, Kohno K, Yamashita K. Changes of relaxation times (T_1, T_2) and apparent diffusion coefficient after permanent middle cerebral artery occlusion in the rat: Temporal evolution, regional extent, and comparison with histology. Magn Reson Med 1995;34:824-834.
147. Anderson AW, Zhong J, Petroff OAC, Szafer A, Ransom BR, Pritchard JW, Gore JC. Effects of osmotically driven cell volume changes on diffusion-weighted imaging of the rat optic nerve. Magn Reson Med 1996;35:162-167.
148. Minkoff L, Damadian R, Thomas TE, Hu N, Goldsmith M, Koutcher J, Stanford M. Physiol Chem Phys 1977;9:101.

149. Mansfield P, Chapman BLW. Active magnetic screening of gradient coils in NMR imaging. J Magn Reson 1986;66:573-576.
150. Turner R. Gradient coil design: a review of methods. Magn Reson Imaging 1993;11:903-920.
151. Bleaney BI, Bleaney B. Electricity and magnetism. Oxford: Clarendon Press; 1965.
152. Jeffreys H, Jeffreys B. Methods of mathematical physics. Cambridge: University Press; 1956.
153. Golay MJE. Rev Scient Instrum 1958;29:313.
154. Hoult DI, Richards RE. Proc Roy Soc 1975;A344:311.
155. Bailles DR, Bryant DJ, Case HA, Collins AG, Cox IJ, Hall AS, Harman RR, Khenia S, McArthur P, Ross BD, Young IR. In vivo implementation of three-dimensional phase encoded spectroscopy with a correction for field inhomogeneity. J Magn Reson 1988;77:460-470.
156. Webb P, Spielman D, Macovski A. Inhomogeneity correction for in vivo spectroscopy by high resolution water referencing. Magn Reson Med 1992;23:1-11.
157. Axel L, Morton D. Correction of phase wrapping in MRI. Med Phys 1989;1,2:284-297.
158. Wen H, Jaffer FA. An in vivo automated shimming method taking into account shim current constraints. Magn Reson Med 1995;34:898-904.
159. Ginsberg DM, Melchner MJ. Rev Scient Instrum 1970;41:122.
160. Baines T, Mansfield P. J Phys E: Scient Instrum 1973;6:462.
161. Turner R, Le Bihan D, Chesnick S. Echo-planar imaging of diffusion and perfusion. Magn Reson Med 1991;19:247-253.
162. O'Sullivan JD. A fast sinc function gridding algorithm for Fourier inversion in computer tomography. IEEE Trans Med Imaging 1985;4(4):200-207.
163. Zakhor A, Weisskoff R, Rzedzian R. Optimal sampling and reconstruction of MRI signals resulting from sinusoidal gradients. IEEE Trans Signal Processing 1991;19(9):2056-2065.
164. Trussel HJ, Arnder LL, Moran PR, Williams RC. Correction for nonuniform sampling distortions in magnetic resonance imagery. IEEE Trans Med Imaging 1991;7(1):32-44.
165. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical recipes in C: The art of scientific computing. Cambridge: University Press;1992.
166. Cooley JW, Tukey JW. Math Computation 1965;19:295.
167. Warach S, Gaa J, Siewert B, Wielopolski P, Edelman RR. Acute human stroke studied by whole brain echo planar diffusion-weighted magnetic resonance imaging. Ann Neurol 1995;37:231-241.
168. Ulug AM, Beauchamp N, Bryan RN, van Zijl PCM. Absolute quantitation of diffusion constants in human stroke. Stroke 1997;28:483-490.
169. Zelaya F, Flood N, Chalk JB, Wang D, Doddrell DM, Strugnell W, Benson M, Ostergaard L, Semple J, Eagle S. An evaluation of the time dependence of the anisotropy of the water diffusion tensor in acute human ischemia. Magn Reson Imaging 1999;17:331-348.
170. Edelstein WA, Bottomley PA, Pfeifer LM. A signal-to-noise calibration procedure for NMR imaging systems. Med Phys 1984;11:180-185.

171. Clark CA, Barker GJ, Tofts PS. Magnetic resonance diffusion imaging of the human cervical spinal cord in vivo. In: Proceedings of the International Society for Magnetic Resonance in Medicine, 1998. p. 529.
172. Shrager RI, Basser PJ. Anisotropically weighted MRI. Magn Reson Med 1998;40:160-165.
173. Papadakis NG, Xing DA, Houston GC, Smith JM, Smith MI, James MF, Parsons AA, Huang CLH, Hall LD, Carpenter TA. A study of rotationally invariant and symmetric indices of diffusion anisotropy. Magn Reson Imaging 1999;17:881-892.
174. Shimony JS, McKinstry RC, Akbudak E, Aronovitz JA, Snyder AZ, Conturo TE. Quantitative diffusion tensor anisotropy imaging: normative human cerebral data. In: Proceedings of the International Society for Magnetic Resonance in Medicine, 1998. p. 1241.
175. Xing D, Papadakis NG, Huang CLH, Lee VM, Carpenter TA, Hall LD. Optimised diffusion-weighting for measurement of apparent diffusion coefficient (ADC) in human brain. Magn Reson Imaging 1997;15:771-784.
176. Laubach HJ, Jakob PM, Loevblad KO, Baird AE, Bovo MP, Edelman RR, Warach S. A phantom for diffusion-weighted imaging of acute stroke. J Magn Reson Imaging 1998;8:1349-1354.
177. Pierpaoli C. Oh no! One more method for color mapping of fiber tract direction using diffusion MR imaging data. In: Proceedings of the International Society for Magnetic Resonance in Medicine, 1997. p. 1741.
178. Jones DK, Williams SCR, Horsfield MA. Full representation of white-matter fibre direction on one map via diffusion tensor analysis. In: Proceedings of the International Society for Magnetic Resonance in Medicine, 1997. p. 1743.
179. Basser PJ. Quantifying errors in fiber-tract direction and diffusion tensor field maps resulting from MR noise. In: Proceedings of the International Society for Magnetic Resonance in Medicine, 1997. p. 1740.
180. Jones DK, Simmons A, Williams SCR, Horsfield MA. Non-invasive assessment of axonal fibre connectivity in the human brain via diffusion tensor MRI. Magn Reson Med 1999;42:37-41.
181. Basser PJ. Fiber-tractography via diffusion tensor MRI (DT-MRI). In: Proceedings of the International Society for Magnetic Resonance in Medicine, 1998. p. 1226.
182. Cantor CR, Schimmel PR. Biophysical chemistry vol. 2. San Francisco: W.H. Freeman and Co.;1980.
183. Basser PJ, Pierpaoli C. Estimating the principal diffusivities (eigenvalues) of the effective diffusion tensor. In: Proceedings of the International Society for Magnetic Resonance in Medicine, 1997. p. 1739.
184. Martin KM, Papadakis NG, Huang CLH. Hall LD, Carpenter TA. The reduction of the sorting bias in the eigenvalues of the diffusion tensor. Magn Reson Imag 1999;17:893-901.
185. Ulug AM, van Zijl PCM. Orientation-independent diffusion imaging without tensor diagonalization: anisotropy definitions based on physical attributes of the diffusion ellipsoid. J Magn Reson Imag 1999;9:804-813.
186. Orrison WW, Lewine DJ, Sanders JA, Hartshorne MF. Functional brain imaging. St. Louis: Mosby-Year Book, Inc.;1995. p. 266.

187. Armitage PA. A prolateness index for further characterising diffusion anisotropy. In: Proceedings of the European Society for Magnetic Resonance in Medicine and Biology 1998. p. 291.
188. Schmitt F, Stehling MK, Turner R. Echo-planar imaging. Berlin: Springer-Verlag;1998.
189. Sorensen AG, Buonanno FS, Gonzalez RG et al. Hyperacute stroke: evaluation with combined multisection diffusion-weighted and hemodynamically weighted echo-planar imaging. Radiology 1996;199:391-401.
190. Lutsep H, Albers G, de Crespigny A, Kamat GN, Marks MP, Moseley ME. Clinical utility of diffusion-weighted magnetic resonance imaging in the assessment of ischaemic stroke. Ann Neurol 1997;41:574-580.
191. Kampfl A, Franz G, Aichner F, Pfausler B, Haring HP, Felber S, Luz G, Schocke M, Schmutzhard E. The persistent vegetative state after closed head injury: clinical and magnetic resonance imaging findings in 42 patients. J Neurosurg 1998;88:809-816.
192. Parizel PM, Ozsarlak, van Goethem JE, van den Hauwe L, Dillen C, Verlooy J, Cosyns P, De Schepper AM. Imaging findings in diffuse axonal injury after closed head trauma. Eur Radiol 1998;8:960-965.
193. Ashikaga R, Araki Y, Ishida O. MRI of head injury using FLAIR. Neuroradiology 1997;39:239-242.
194. Wild JM, MacMillan CS, Wardlaw JM, Marshall I, Cannon J, Easton VJ, Andrews PJ. 1H spectroscopic imaging of acute head injury - evidence of diffuse axonal injury. MAGMA 1999;8:109-115.
195. Ito H, Ishii K, Onuma T, Kawashima R, Fukuda H. Cerebral perfusion changes in traumatic brain injury; IMP SPECT studies. Ann Nucl Med 1997;11:167-172.
196. Macmillan CSA, Wild JW, Andrews PJD, Marshall I, Armitage PA, Wardlaw JM, Easton VJ, Cannon J. Accuracy of a miniature intracranial pressure monitor, its function during MR scanning and assessment of image artefact generation. Neurosurgery 1999;45(1):188-93.