From Diffusion to the Diffusion Tensor

4

Thijs Dhollander

Learning Points

- The diffusion coefficient; the apparent diffusion coefficient (ADC); their relation to the pair of acquired diffusion-weighted images (DWI) and nondiffusion-weighted images (B0); and a few general properties of the ADC in relation to the amount of diffusion weighting (the *b*-value).
- The concept of directionality of the diffusion sensitizing gradient; the concept of anisotropy in the acquired data; and inferring useful information on the orientation of axon bundles based on raw DWIs, normalized DWIs, ADC maps, and spherical polar plots of the latter two quantities.
- The apparent diffusion tensor; evaluating a tensor model; the link between the diffusion tensor and the acquired DWIs/B0; the meaning and interpretation of the tensor elements; and visualizing the tensors by spherical polar plots (ADC peanuts).
- Eigendecomposition of the tensor; the meaning and interpretation of eigenvalues and eigenvectors; and interpreting maps of eigenvalues and directionally encoded color (DEC) maps of eigenvectors.
- Basic visualization of tensors by glyphs; the meaning of the tensor ellipsoid; and maps of common measures such as mean diffusivity (MD), fractional anisotropy (FA), DEC FA, and a few other shape measures.
- The issue of tensor fitting; and specific tensor fitting methods: linear least squares (LLS), weighted linear least squares (WLLS), nonlinear least squares (NLS), and robust estimation of tensors by outlier rejection (RESTORE).

T. Dhollander, PhD (✉)
The Florey Institute of Neuroscience and Mental Health, Melbourne Brain Centre, 245 Burgundy Street, Heidelberg, VIC 3084, Australia

Medical Imaging Research Center (MIRC), KU Leuven, Leuven, Belgium
e-mail: thijs.dhollander@florey.edu.au

The (Self-)Diffusion Coefficient

Measuring (Self-)Diffusion in the MR Scanner

Sit down, relax, and grab yourself a glass of water. Now put it in a nearby MR scanner (or rather, imagine doing this). Acquire a (nondiffusion-weighted) T2-weighted image as well as one of those fancy new diffusion-weighted images. Now let's see if we can recover the diffusion coefficient of water from these two images. More accurately, we're talking about the *self diffusion coefficient* here: it quantifies the freedom of movement of any single molecule of water, in

W. Van Hecke et al. (eds.), *Diffusion Tensor Imaging*, DOI 10.1007/978-1-4939-3118-7_4

the glass of water. We'll simply refer to it as *D*. Also note that we need (at least) two images: the diffusion-weighted image would appear exactly the same as the T2-weighted image, if it were not *weighed down* by the appearance of diffusion; i.e., we're interested in the relative difference between both images. Since *D* should be the same in the entire glass of water, we simply choose one voxel. The intensity of the diffusion-weighted image in this voxel will be referred to as *S*, while the (non-diffusion-weighted) T2-weighted image's intensity equals S_0. As explained in Chap. 3, the process of diffusion should have caused *attenuation* in *S*, so *S* should always be smaller than S_0. The *decay* of *S* relative to S_0 is given by the so-called Stejskal-Tanner equation [1]:

$$S = S_0 \bullet e^{-b \bullet D} \tag{4.1}$$

The *b*-factor in this equation captures all the relevant *scanning parameters* and was introduced to take abstraction of them [2]. In general, it can be seen as the amount of *diffusion weighting* that is applied; i.e. how sensitive the acquisition is to diffusion. Its value is typically set and reported in s/mm^2. As a realistic value for our simple experiment at hand, we could have chosen e.g. 800 s/mm^2. We can rewrite this equation so it becomes

$$-\ln\left(\frac{S}{S_0}\right) = b \bullet D \tag{4.2}$$

The left side of the equation only contains measurements we obtained from the scanner (*S* and S_0), while the right side contains the scanning parameters in function of which we did so (all contained within the *b*-factor) and a constant (*D* for our glass of water at room temperature). From this we learn that a *logarithmic transform* (i.e., "$-\ln(\ldots)$") of our *normalized measurement* (i.e., "S/S_0") depends *linearly* on the applied diffusion weighting (i.e., *b*). As they are both simply related by a factor *D*, plotting $-\ln(S/S_0)$ in function of *b* yields a straight line through the origin, as shown in (Fig. 4.1). The slope of this line equals *D*:

$$D = \frac{-\ln\left(\frac{S}{S_0}\right)}{b} \tag{4.3}$$

The value of *D* is typically reported in mm^2/s. For our glass of water, $D = 2.2 \times 10^{-3}$ mm^2/s should be realistic at room temperature. If we were lying in the scanner ourselves and performed the above calculation for a voxel of cerebrospinal fluid (CSF) in the ventricles of our brain, a value of about 3.1×10^{-3} mm^2/s is to be expected. While CSF consists mostly (99 %) of water, the difference can be explained by our *body temperature*, which is of course higher than the normal room temperature. One point is enough to fully fix the slope of the line in (Fig. 4.1) and per consequence also determine *D*. If we would have performed the measurement using a different *b-value*, we would obtain another point on this exact same line. Using a *higher b*-value would result in *more decay*, and thus a *lower value for S* (this can be most easily appreciated by looking at Eq. 4.1). A *lower value for S* means a *higher value for* $-\ln(S/S_0)$. Consequently, we are simply considering a point further up the same line in (Fig. 4.1).

Conclusions

We are now able to calculate the *self-diffusion coefficient D* of free water (be it in a glass or as CSF in the ventricles), using measurements from

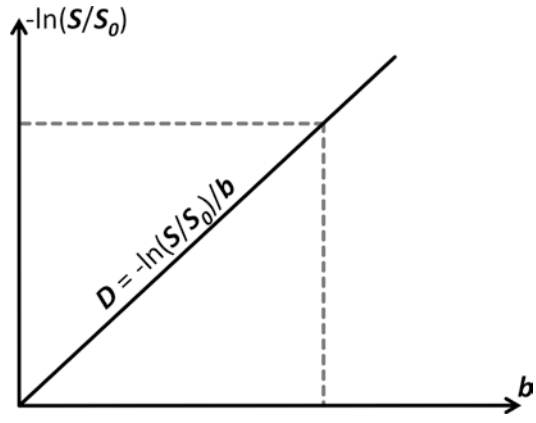


Fig. 4.1 In case of free diffusion (e.g., in a glass of water, or CSF in the ventricles of the brain), the plot of $-\ln(S/S_0)$ in function of *b*-value is a *straight line* through the origin. One point (*grey short dashed lines*) is enough to fully fix this line. It requires two images (*S* and S_0) as well as knowledge of the *b*-value used to acquire *S*. The slope of the resulting line equals the self-diffusion coefficient *D*

a MR scanner and the Stejskal-Tanner equation. The *minimum requirements* are a nondiffusion-weighted image (S_0), a diffusion-weighted image (S) and knowledge of the b-value that was used for performing the acquisition of the diffusion-weighted image.

The Apparent Diffusion Coefficient

Apparent Complications

Feeling confident about the newly gained ability to obtain D from the two images we acquired of our brain, we also attempt to perform the same calculation in a voxel of gray matter. Suddenly, however, we are confronted with a resulting value of about 0.9×10^{-3} mm^2/s. *Apparently*, the self-diffusion coefficient of water has changed, just because we measured it in the gray matter. Maybe something went wrong with the scan? We perform the acquisition again for a couple of different b-values. Carefully dotting out the obtained values of $-\ln(S/S_0)$ in function of b and connecting everything loosely by hand, we obtain a curve such as the one depicted in (Fig. 4.2). *Apparently*, the self-diffusion coefficient of water now even changes in function of our chosen acquisition parameters. Using Eq. (4.3) to calculate D equates to connecting a certain measured point on this curve with the origin by a *straight line* (as shown in Fig. 4.2) and assuming its slope still equals D. Since the obtained values are clearly lower than expected and they also seem to vary in function of b, such a value is referred to as an *apparent diffusion coefficient* (*ADC*) [3]. It's calculated from the measurements in exactly the same way as D:

$$\mathrm{ADC} = \frac{-\ln(\frac{S}{S_0})}{b} \tag{4.4}$$

To understand the behavior of the obtained ADC in regions containing tissue (e.g., gray matter), we need to look into how the acquisition of a diffusion-weighted image works. An existing (e.g., T2-weighted) sequence is modified by adding a

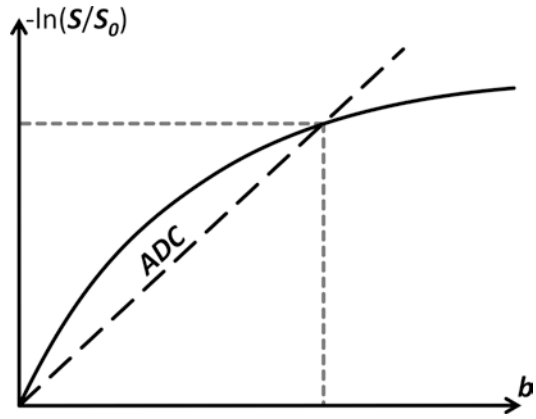


Fig. 4.2 In case of hindered/restricted diffusion (in tissue, e.g. the grey matter of the brain), the plot of $-\ln(S/S_0)$ in function of b-value is a *curve* through the origin. Based on one point (*grey short dashed lines*), we can calculate an apparent diffusion coefficient (ADC). Just like D, it equals the slope of the line that connects this point to the origin (*black long dashed line*)

couple of diffusion-sensitizing gradients. By taking abstraction of any complicated MR physics, we could say the MR scanner actually performs a simple experiment in each voxel: it takes a snapshot of all the water molecules, waits a bit, and then takes another snapshot. During the short waiting time, however, the molecules have the opportunity to *diffuse* a bit. Per consequence, a relative *displacement* of each molecule can take place in between both snapshots. The expected signal of the original (e.g., T2-weighted) sequence is *attenuated* in function of the amount of *displacement* of all water molecules in the voxel (as well as the amount of applied diffusion weighting). From the measurements of such an experiment (relative to a nondiffusion-weighted image), the Stejskal-Tanner equation is able to reliably calculate D… if and only if nothing disturbs the experiment. However, in tissue, such as gray matter, there are *cell membranes* all over the place. Because the water molecules happen to bump into the cells—i.e., they are *hindered*—they have a harder time to diffuse further away during the experiment. Water inside the cells may even be *restricted* to a confined space. And thus, our calculation of D will *apparently* yield a lower outcome, which is why we call it the ADC instead. The time the experiment allows the molecules to

diffuse is one of the parameters that makes up the *b*-value. It's easy to imagine that a larger diffusion time will allow more molecules to hit some of these cell membranes. Hence, the effect of the *hindered/restricted* diffusion on our measurement will increase with *b*-value; yet another reason to refer to the outcome of our calculations using the term "ADC". This is also illustrated in (Fig. 4.3): using a *larger b*-value renders the measurement of *S less sensitive to* (*truly*) *free diffusion*, in favor of hindered/restricted diffusion. As such, *S* will be *less attenuated* and the value of $-\ln(S/S_0)$ will be *smaller* than expected, yielding a *downward curvature* when plotting $-\ln(S/S_0)$ in function of *b*. This finally leads to an important property of the ADC in tissue, as indicated in (Fig. 4.3): using a *higher b*-value results in a *lower ADC*.

Apparent Advantages

At this point, you might start to wonder what the point is of trying to find out *D* in voxels containing tissue, only to end up having to deal with a deceiving ADC instead. However, you have to look at it from the bright side: we now effectively have access to a *probe* that tells us something about these *cells* that hinder/restrict diffusion. That's right: even though our voxel size might be quite crude ($2\times2\times2$ mm^3 or larger is not unusual), the measured values are sensitive to differences in structure at a *micrometer* scale! We are not interested in the ADC for the purpose of quantifying diffusion itself, but rather to *investigate properties of the tissue* that apparently caused the diffusion process to behave in the way that we measure.

Before moving on, let's investigate one more property of the ADC that teaches us something else about its capacity in distinguishing different tissues. Consider the setting in (Fig. 4.4): it presents again $-\ln(S/S_0)$ in function of *b*, but this time for measurements at two different locations (e.g., in the brain). Looking at the plots and applying what we just learned, we can safely say that the voxel at position p_g contains more hindering/restricting tissue than the voxel at position p_f. The former voxel's plot shows greater curvature,

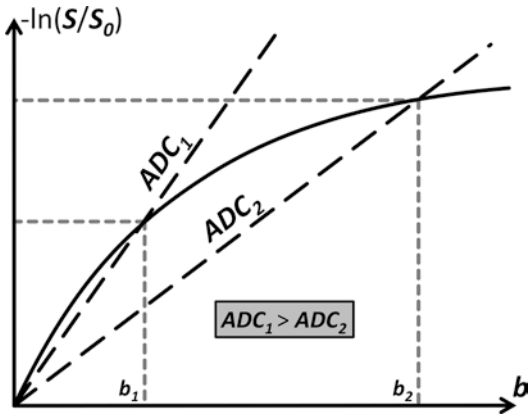


Fig. 4.3 The ADC is dependent on the *b*-value used to acquire *S*. Due to the downwards curvature of the plot of $-\ln(S/S_0)$ in function of *b*-value, a larger *b*-value results in a lower ADC. An explanation lies, e.g., in the fact that increasing the diffusion time allows more molecules to bump into cell membranes. This will on average decrease their final displacement, resulting in a reduced amount of attenuation of *S* and finally leading to a lower value for $-\ln(S/S_0)$ than expected in a free (non-hindered/restricted) environment

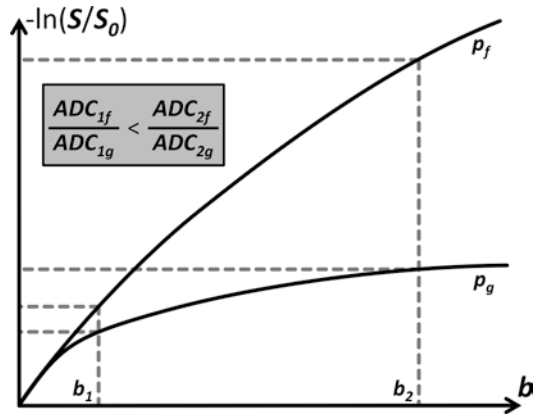


Fig. 4.4 The contrast of the ADC, e.g., between two different tissues at voxel positions p_f and p_g, is dependent on the *b*-value used to acquire *S*. Due to different tissue properties, both plots of $-\ln(S/S_0)$ in function of *b*-value show a different curvature. The tissue at p_f imposes less hindrance/restriction on the diffusion as compared to the tissue at p_g and thus the accompanying curve is closer to a *straight line*. Hence, ADC_f is larger than ADC_g for a given *b*-value. Increasing the *b*-value also results in an increase of the relative difference between both ADC values, i.e., an increase of contrast

while the latter better approximates the straight line we would expect in case of free diffusion. Due to this difference in curvature of both plots, the *relative difference* in magnitude of $-\ln(S/S_0)$, and thus also ADC, *increases for larger b-values*. In other words, using a *larger b*-value results in a *better contrast* when calculating an ADC map. This fact of course begs the question why we should still limit ourselves to a certain *b*-value. That is, why not use an absurdly high *b*-value for maximal contrast? The two most important factors that generally contribute to the *b*-value are the strength of the applied diffusion-sensitizing gradients and the time that we allow the water to diffuse during the experiment. The former is limited by what we can achieve with available hardware. The latter is fully under our control. Allowing a too long diffusion time, however, might result in other more macroscopic motion to be captured and thus confounding our measurements. Even if this would not be the case, we also have to recall that *S* only *decays* further in function of *b*-value (remember Eq. 4.1 again?). The *noise level* of our measurements, on the other hand, *does not decrease*; that is, using a *higher b*-value yields a *lower signal-to-noise ratio* (*SNR*)!

Conclusions

We have learned why the MR measurements in combination with the Stejskal-Tanner equation are not suited to calculate the true self-diffusion coefficient of water in voxels containing tissue, e.g., where diffusion is *hindered* or even *restricted*. The obtained *apparent diffusion coefficient* (*ADC*), on the other hand, can provide interesting information about the microstructure of the tissue under investigation. The *minimum requirements* for obtaining it are again a nondiffusion-weighted image (S_0), a diffusion-weighted image (S) and knowledge of the *b*-value that was used for performing the acquisition of the diffusion-weighted image. The ADC is, however, dependent on the *b*-value: a *higher b*-value results in a *lower ADC*. It also *improves the contrast* (e.g., of the ADC map), but at the cost of a *reduced SNR*. Due to these dependencies, interpreting/reporting the ADC only makes sense when the *b*-value is also specified. Finally, comparing ADC values or maps originating from acquisitions with different *b*-values does *not* make a lot of sense.

Gradient Directions and Anisotropy

Anisotropic Complications

Up to now, we've been silently ignoring yet another important fact that will complicate everything even more. It concerns that diffusion-sensitizing gradient: it's about time we started taking into account that it's applied along a certain *direction*. Nothing to worry about, if it were not for the fact that our measurements are only sensitive to diffusion along this direction. Actually, that is not fully correct; it's better to say that they are *only sensitive to diffusion with a component along this direction*. Before we start talking further about directions, let's settle on some reference frame. We define three (perpendicular) axes through the brain as follows: x runs from *left to right*, y from *back to front*, and z from *bottom to top*. So suppose we would apply the diffusion gradient along the direction of x, what are the implications then? It basically means that the measurements are *fully sensitive* to diffusion *along x*, but *gradually less sensitive* to diffusion along directions that *increasingly deviate* from x, up to the point where they are *completely insensitive* to diffusion along directions *perpendicular* to x (i.e., directions in the yz-plane).

But why should we worry about directionality of diffusion anyway? In our earliest experiments with a glass of water or CSF in the ventricles, we shouldn't: diffusion takes place *equally in all directions*. In tissue randomly containing cells—imagine a bunch of spherical cells packed together—diffusion is hindered and restricted, yet probably more or less *equally in all directions*. So again, there's nothing to worry about: as we are in both cases studying *isotropic* measurements, it is sufficient to only measure along a single direction. Our findings (e.g. calculating the

ADC) should have been the same for measurements along any other direction. But let's consider the more interesting case of the white matter in the brain: it consists of long coherent bundles of axons, almost resembling a bunch of cylindrical tubes packed closely together (see Chap. 3). One can imagine that water molecules in between and inside these tubes have *an easier time diffusing along them rather than perpendicular to them*. We thus say that diffusion in the white matter is *anisotropic*.

But how relevant is this? Is this anisotropy large enough to be measured; i.e. can we see it in our diffusion-weighted images? To answer this question, we'll introduce some real data. In (Fig. 4.5), we start by presenting a classic T1-weighted and T2-weighted image for reference. Next is the nondiffusion-weighted image: it's again a T2-weighted image, but it already shows the *lower spatial resolution* at which DWI datasets are typically acquired. In this case, the voxel size equals $2.2 \times 2 \times 2$ mm^3. Because the image is not diffusion-weighted, but it is acquired as part of a DWI dataset, we also sometimes (informally) refer to it as the "B0" (it equals a diffusion-weighted image with a b-value of 0). For convenience, we already applied a whole brain mask to it. On the second row of (Fig. 4.5), three diffusion-weighted images (DWIs) are shown (also masked). They were all acquired

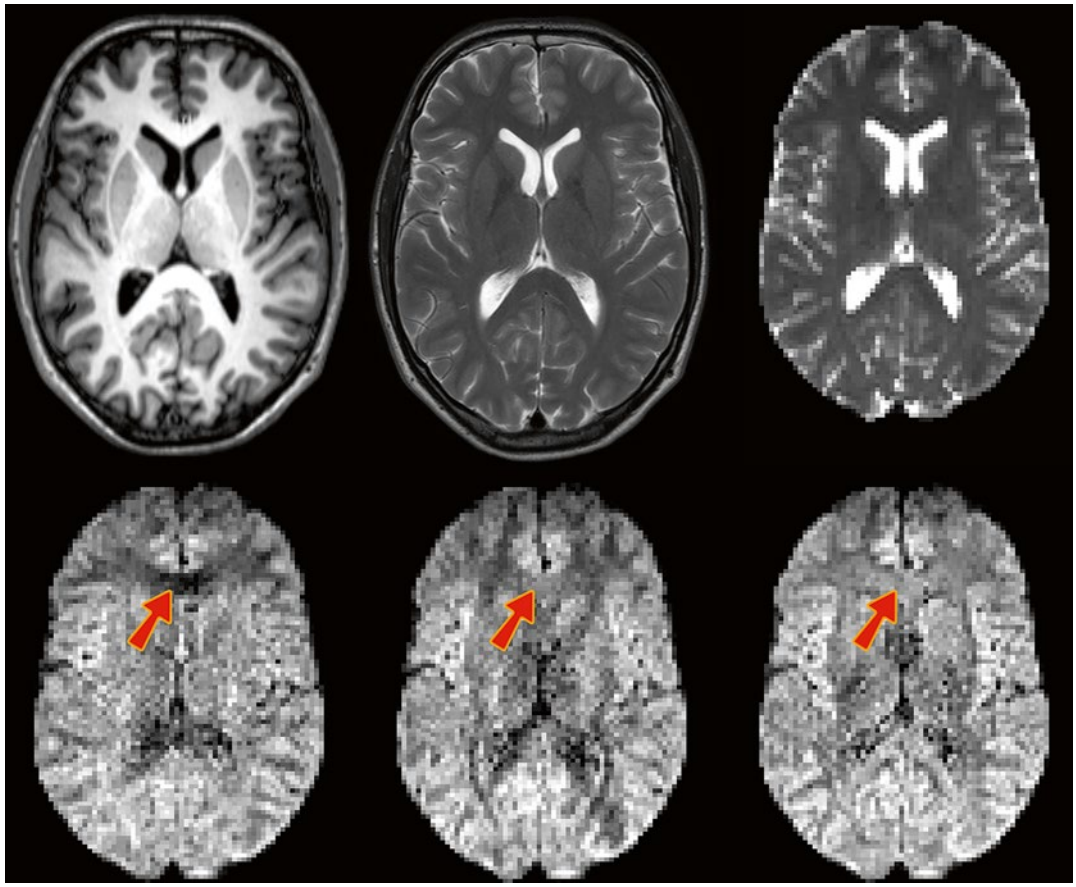


Fig. 4.5 *Top row*: T1-weighted image, T2-weighted image, B0 image ("diffusion-weighted image" with a b-value of 0, i.e., non diffusion-weighted). *Bottom row*: diffusion-weighted images (DWIs) acquired by applying gradients along the direction of x, y and z. The *arrows* indicate a region in the genu of the corpus callosum (GCC), where the anisotropy can be easily seen and understood

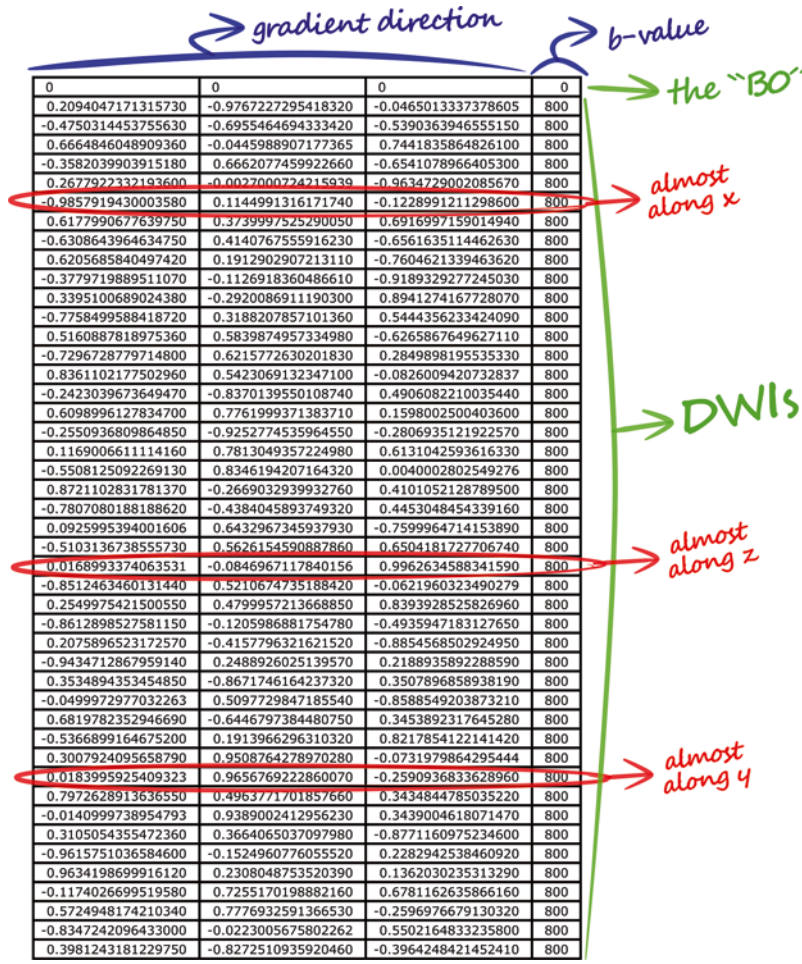
using exactly the *same amount* of diffusion weighting: $b = 800$ s/mm^2. The diffusion-sensitizing gradients are, however, applied *along different directions*: respectively along the direction of x, y and z. Differences in contrast can clearly be seen, which consequently confirms that we will have to account for *anisotropy* in our measurements.

Anisotropic Advantages

Just as when we introduced the ADC, we'll also try to use this fact to our advantage: we now have access to a probe that might even provide us with information on the *anisotropy of the microstructure* that hinders and restricts the process of diffusion. Applying what we have learned from this chapter up to this point, let's see if we can already figure out something useful from these three diffusion-weighted images. Consider the indicated region in the genu of the corpus callosum (GCC): it has *a low DWI-intensity along x*, but a (relatively) *higher DWI-intensity along y and z*. Because we know that more diffusion causes increased decay of S (the DWI-intensity), we can conclude from these images that there is *more free diffusion along x*, while there is *more hindrance and restriction along y and z*. Translating this to "reality", we might infer that *a bundle of tubelike axons runs along the left-right axis* in this region, connecting both hemispheres of the brain. Note that we are applying *inductive* reasoning here: we know that such a left-right oriented structure would result in such a pattern of diffusion and thus also such DWI measurements along these three directions, yet we reason that the latter measurements were effectively caused by the former structure. Considering we only measured along three directions, that's a pretty strong conclusion. Of course, inherently we might have also applied possible anatomical knowledge and the fact that a structure along this direction makes sense considering the spatial/anatomical neighborhood of the region (i.e., the region is in between both hemispheres).

Getting a Grip on the Information Overload

In practice, however, we will typically perform the acquisition using more than three different gradient directions. In the previous example, we were just lucky that the structure under investigation accidently happened to run along one of the three mutually perpendicular directions that we sampled. If it would instead be running at any other oblique angle, these three measurements would clearly be *inadequate* to determine its direction. In our experiment at hand, however, we actually acquired DWIs for a total of *45 different gradient directions*! The specifics of such an acquisition are presented in the *gradient table*, that contains *one row for each acquired image*, representing its gradient direction and b-value. The gradient table for our current experiment is provided in (Fig. 4.6). As it was already quite tedious to infer information by mentally combining three images, considering 45 DWIs all at once is nearly impossible. To begin with, we will no longer visualize the original DWIs, as they are still only representing a (partially) decayed T2-weighted signal. Because of this, these DWIs suffer so-called *T2 shine-through*: a higher intensity might not (only) result due to hindered/restricted diffusion; it might also be caused by an originally high T2 intensity (e.g., in areas containing CSF). Therefore, it is more evident to consider DWIs after *normalization by the B0* (i.e., the normalized measurements "S/S_0"). Such normalized versions of our original DWIs for the three x, y, and z gradient directions are shown in (Fig. 4.7). Next up is the actual challenge of visualizing the information of all 45 normalized DWIs in a conveniently organized way. Rather than showing 45 separate images, we could try to combine all information of each single voxel and visualize it within that particular voxel. Since the different values of S/S_0 are a *function of the gradient direction*, a *spherical polar plot* is the perfect candidate for the job. In such a plot, the radius of a sphere is locally manipulated to equal the function value at that three-dimensional angle. We also smoothly interpolated the values



0	0	0	0
0.2094047171315730	-0.9767227295418320	-0.0465013337378605	800
-0.4750314453755630	-0.6955464694333420	-0.5390363946555150	800
0.6664846048909360	-0.0445988907177365	0.7441835864826100	800
-0.3582039903915180	0.6662077459922660	-0.6541078966405300	800
0.2677922332193600	-0.0027000724215939	-0.9634729002085670	800
-0.9857919430003580	0.1144991316171740	-0.1228991211298600	800
0.6177990677639750	0.3739997525290050	0.6916997159014940	800
-0.6308643964634750	0.4140767555916230	-0.6561635114462630	800
0.6205685840497420	0.1912902907213110	-0.7604621339463620	800
-0.3779719889511070	-0.1126918360486610	-0.9189329277245030	800
0.3395100689024380	-0.2920086911190300	0.8941274167728070	800
-0.7758499588418720	0.3188207857101360	0.5444356233424090	800
0.5160887818975360	0.5839874957334980	-0.6265867649627110	800
-0.7296728779714800	0.6215772630201830	0.2849898195535330	800
0.8361102177502960	0.5423069132347100	-0.0826009420732837	800
-0.2423039673649470	-0.8370139550108740	0.4906082210035440	800
0.6098996127834700	0.7761999371383710	0.1598002500403600	800
-0.2550936809864850	-0.9252774535964550	-0.2806935121922570	800
0.1169006611114160	0.7813049357224980	0.6131042593616330	800
-0.5508125092269130	0.8346194207164320	0.0040002802549276	800
0.8721102831781370	-0.2669032939932760	0.4101052128789500	800
-0.7807080188188620	-0.4384045893749320	0.4453048454339160	800
0.0925995394001606	0.6432967345937930	-0.7599964714153890	800
-0.5103136738555730	0.5626154590887860	0.6504181727706740	800
0.0168993374063531	-0.0846967117840156	0.9962634588341590	800
-0.8512463460131440	0.5210674735188420	-0.0621960323490279	800
0.2549975421500550	0.4799957213668850	0.8393928525826960	800
-0.8612898527581150	-0.1205986881754780	-0.4935947183127650	800
0.2075896523172570	-0.4157796321621520	-0.8854568502924950	800
-0.9434712867959140	0.2488926025139570	0.2188935892288590	800
0.3534894353454850	-0.8671746164237320	0.3507896858938190	800
-0.0499972977032263	0.5097729847185540	-0.8588549203873210	800
0.6819782352946690	-0.6446797384480750	0.3453892317645280	800
-0.5366899164675200	0.1913966296310320	0.8217854122141420	800
0.3007924095658790	0.9508764278970280	-0.0731979864295444	800
0.0183995925409323	0.9656769222860070	-0.2590936833628960	800
0.7972628913636550	0.4963771701857660	0.3434844785035220	800
-0.0140999738954793	0.9389002412956230	0.3439004618071470	800
0.3105054355472360	0.3664065037097980	-0.8771160975234600	800
-0.9615751036584600	-0.1524960776055520	0.2282942538460920	800
0.9634198699916120	0.2308048753520390	0.1362030235313290	800
-0.1174026699519580	0.7255170198882160	0.6781162635866160	800
0.5724948174210340	0.7776932591366530	-0.2596976679130320	800
-0.8347242096433000	-0.0223005675802262	0.5502164833235800	800
0.3981243181229750	-0.8272510935920460	-0.3964248421452410	800

Fig. 4.6 The gradient table contains one row for each acquired image. The x, y and z components of the gradient direction are provided in the first three columns, while the b-value is given in the last one. A b-value of 0 indicates a B0 image; the gradient direction is irrelevant in such a case. The *red encircled rows* refer to the DWIs presented in Fig. 4.5

between the 45 directions in order to achieve the final visualization in (Fig. 4.7). Note that, due to the multitude of information on display, we have to zoom in up to a reasonable level to show everything with the required amount of detail. We choose to further focus on the region of the GCC that was the subject of our earlier thought experiment. Furthermore, a little extra *color* was added to the plot: each point on the surface of the spherical polar plots is colored according to its *direction*: *red* is assigned to x, *green* to y and *blue* to z. In (the middle of) the GCC, we spot larger values for green (y) and blue (z), and smaller values for red (x). By linking *larger values* to *hindrance/restriction*, we can thus confirm our hypothesis of an axonal bundle connecting left and right.

Associating larger values with less diffusion still feels a bit awkward, to say the least. So why don't we simply employ the ADC values? Easy enough: just calculate the 45 ADC maps from the normalized DWIs using Eq. (4.4). We present these maps—again for the three x, y, and z gradient directions—in (Fig. 4.8). This time, *larger values* equal *more free diffusion*. We can just as well create a spherical polar plot of the 45 ADC values in each voxel, which is again provided for the GCC region in (Fig. 4.8). *Larger values* are now conveniently oriented along the direction of the *greatest amount of free diffusion*, and colored accordingly. Finally, remember that property of the contrast increasing with b-value? Of course, it also applies for measurements (and ADC values) acquired using different gradient directions: using

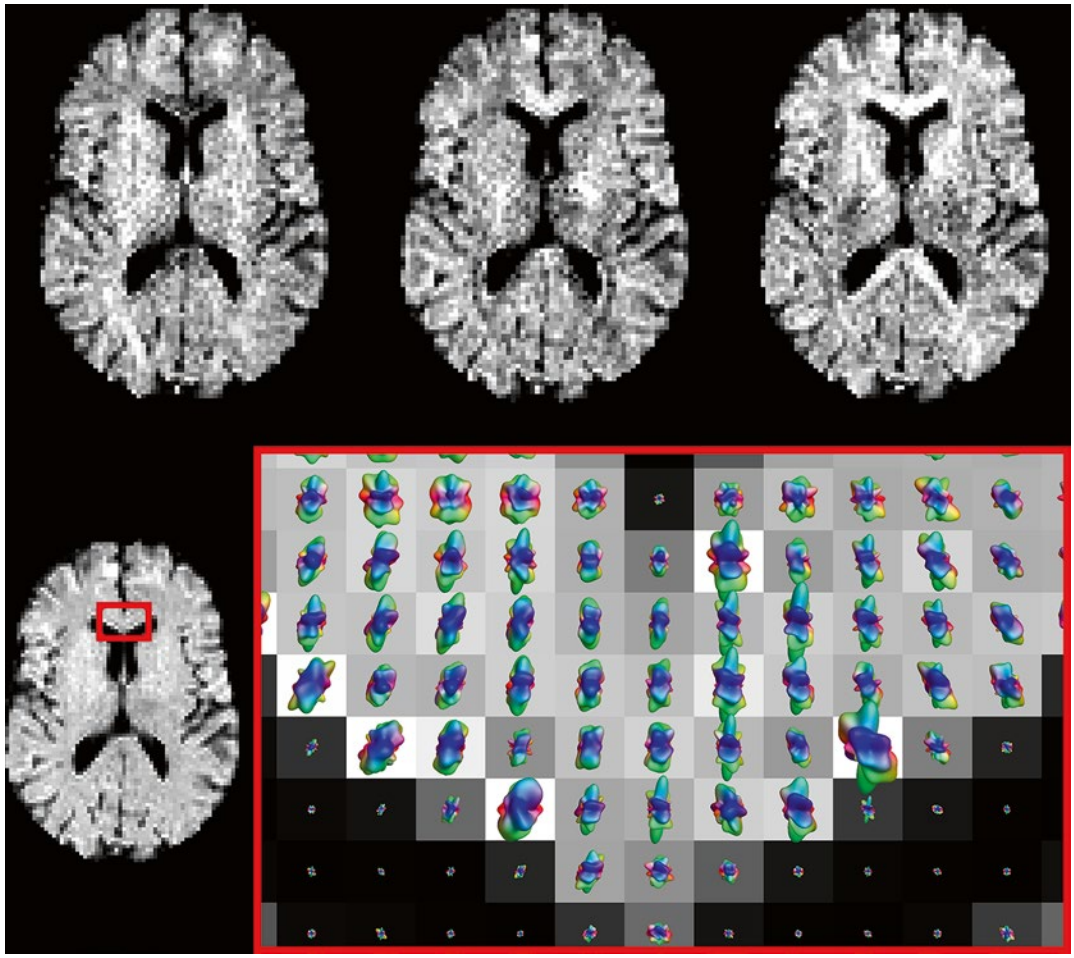


Fig. 4.7 *Top row*: DWIs for the x, y, and z gradient directions, after normalization by the B0 (i.e., the normalized measurements "S/S_0"). *Bottom row*: Spherical polar plots of the normalized DWI values in a region of the GCC, overlaid on a map of the average normalized DWI value

a *higher b*-value will *increase the contrast* in these spherical polar plots. However, as we reasoned before, the SNR will also drop.

Conclusions

We started taking into account the fact that the diffusion-sensitizing gradient is applied along a certain *direction*. The resulting DWI measurement is only sensitive to *diffusion with a component along this direction*. From DWIs acquired using different gradient directions, we could conclude that *anisotropic diffusion* takes place in the *white matter* up to a measurable extent. Again using this to our advantage, we now have a probe for the *anisotropy of microstructure* in each voxel. There are different ways to visualize data resulting from acquisitions using many different gradient directions, yet the most convenient option was *a spherical polar plot of the ADC values* in each voxel: such a visualization shows *larger values* along the direction exhibiting the *greatest amount of free diffusion*. Optional *color coding* is typically done according to a directional scheme: *red for x* (*left-right*), *green for y* (*back-front*), *and blue for z* (*bottom-top*). The *requirements* for investigating the anisotropic nature of diffusion are a B0

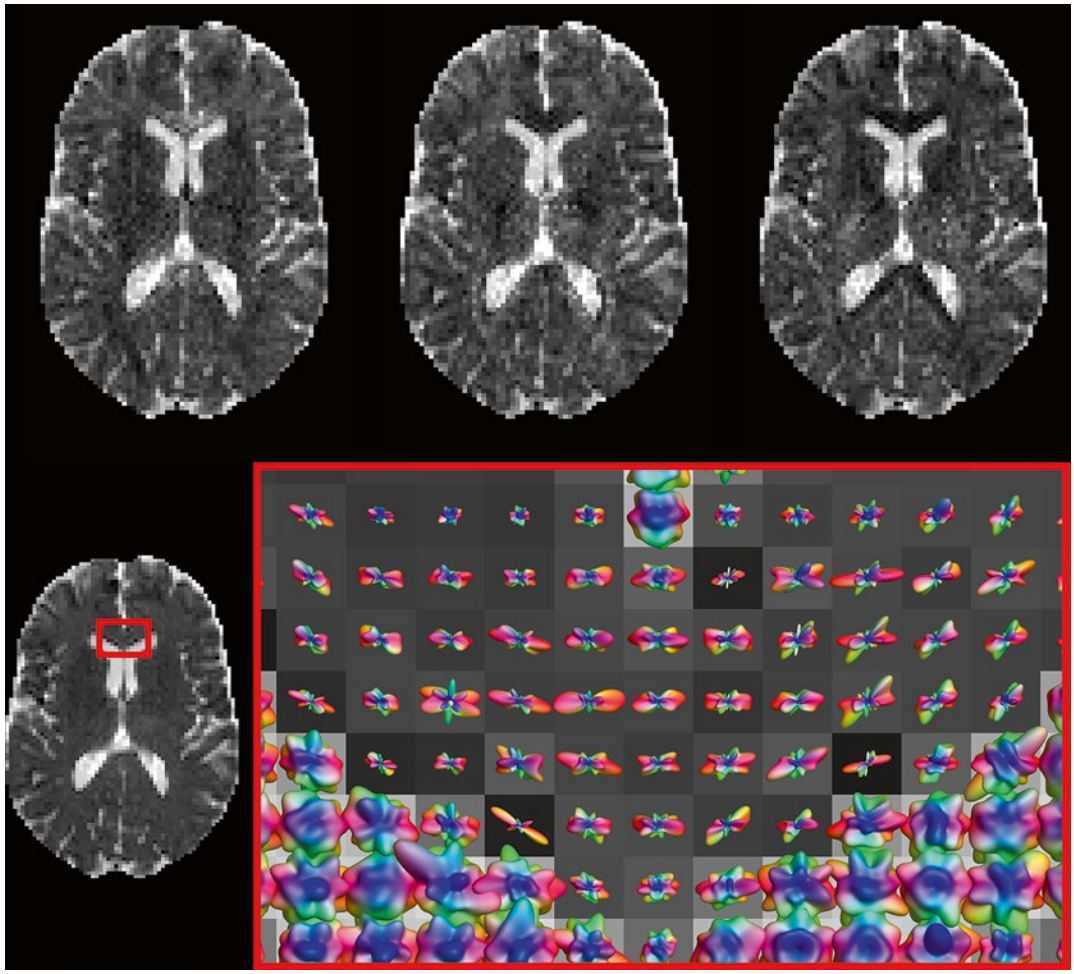


Fig. 4.8 *Top row*: ADC maps for the *x*, *y*, and *z* gradient directions. *Bottom row*: Spherical polar plots of the ADC values in a region of the GCC, overlaid on a map of the average ADC value

(nondiffusion-weighted image), a number of DWIs and knowledge of the *b*-value *and* ***gradient directions*** that were used for performing the acquisition of the DWIs. ***This latter point is very, very important***! Did we just stress that enough? Because it is (very, very important): without the accompanying *b*-value *and* ***gradient directions***, the full set of carefully acquired DWIs is nigh *useless*; we wouldn't be able to associate the (normalized) measurements nor the ADC values with any directions. This vital piece of information should thus be stored with the data; it is often summarized in a gradient table, as shown in (Fig. 4.6). On the number of gradient directions: more measurements are of course always better, yet require more scanning time. And finally, a *higher b*-value yields *better contrast*—also in the spherical polar plots of, e.g., the ADC values—but will *reduce SNR*.

The (Apparent) Diffusion Tensor

Motivation and Implications of Modeling

Looking back at the spherical polar plots of the ADC values in (Fig. 4.8), we notice that they appear quite noisy. That's not surprising, since they simply present a logarithmic transformation

of the original (normalized) data: nothing is modeled, all the measurement noise is still showing (albeit logarithmically transformed … remember this, as it will happen to bug us later on). And thus models were invented. Without going into the how and why of some historical choices that have been made in model development, we'll just introduce the (legendary) *diffusion tensor model* [4] that is central to the theory and practice of DTI. In this context, to be exact, we should refer to it as the *apparent* diffusion tensor. This name refers to the fact that we will employ a tensor to represent/model *the values of the ADC* in function of (gradient) direction, in each voxel. This means that, once we have somehow determined the correct parameters of this model in each voxel, we can evaluate it for as many directions as we like in order to visualize it again as a spherical polar plot of (*modeled*) *ADC values*. As the diffusion tensor model has *only six parameters* (compare this to the 45 ADC values we just obtained from our dataset in each voxel!), it will greatly simplify the features of our directional profile of the ADC.

Understanding DTI, in Theory: The Maths!

Mathematics … it's not as hard as it sounds, so let's just get to it then! From this point on, we will represent a (gradient) direction by a three-element column *vector* **g**, and the apparent diffusion tensor **D** by a 3×3 *symmetric matrix*:

$$\mathbf{g} = \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{xy} & D_{yy} & D_{yz} \\ D_{xz} & D_{yz} & D_{zz} \end{bmatrix} \tag{4.5}$$

For the mathematics (and software that employs it) to work out well, **g** should be a *unit vector*. As stated before, the tensor **D** has *only six free parameters* (the tensor elements D_{xx}, D_{yy}, D_{zz}, D_{xy}, D_{xz}, D_{yz}) because its matrix is *symmetric*: the elements above and below the main diagonal are the same. We'll get into the meaning of these separate tensor elements later. Given such a tensor **D**, we can "*evaluate*" it for a given direction **g** by using the following expression:

$$\begin{aligned} \mathbf{g}^{\mathrm{T}}\mathbf{D}\mathbf{g} = {} & g_x^2 D_{xx} + g_y^2 D_{yy} + g_z^2 D_{zz} \\ & + 2g_x g_y D_{xy} + 2g_x g_z D_{xz} + 2g_y g_z D_{yz} \end{aligned} \tag{4.6}$$

where $\mathbf{g}^{\mathrm{T}}$ is the transpose of **g**. The right side of the equation simply shows what you would obtain if you did the symbolic math by hand using the vector and tensor element symbols from Eq. (4.5). The outcome of this expression—if we were to fill in some specific numbers representing the vector and tensor elements—is thus a single scalar number: *the value of our tensor model, along a given direction*. As we will now employ such a tensor to symbolize the ADC values, we can simply plug it into the good old Stejskal-Tanner Eq. (4.1) to obtain the following expression:

$$S = S_0 \bullet e^{-b \bullet \mathbf{g}^{\mathrm{T}}\mathbf{D}\mathbf{g}} \tag{4.7}$$

Don't take this one lightly: **this is the *essence* of DTI**. It provides the direct relationship between the chosen experimental parameters (b and **g**), the measurements (S and S_0), and the parameters of the diffusion tensor model (**D**). It now effectively includes the gradient direction **g** that we took abstraction of before, while the vehicle to describe the ADC is no longer a single number, but a *tensor* that can describe values that vary in function of (gradient) direction. Just like we did before with the Stejskal-Tanner equation, we can rewrite Eq. (4.7) to single out the parts that equate to the ADC:

$$\mathbf{g}^{\mathrm{T}}\mathbf{D}\mathbf{g} = \frac{-\ln\left(\frac{S}{S_0}\right)}{b} \tag{4.8}$$

The right-hand side equals the expression of the ADC that we introduced before (i.e. Eq. 4.4), while the left-hand side simply says that we would like to see this ADC value arising from our model **D** when evaluated for the gradient direction **g** that this particular ADC value relates to. Completely writing out the left-hand side

expression using Eq. (4.6) finally yields the following result:

$$\begin{aligned} & g_x^2 D_{xx} + g_y^2 D_{yy} + g_z^2 D_{zz} \\ & +2g_x g_y D_{xy} + 2g_x g_z D_{xz} + 2g_y g_z D_{yz} \\ & = -\ln(S / S_0) / b \end{aligned} \tag{4.9}$$

If we now perform a DWI experiment as before (acquiring S and S_0, carefully noting down b and **g**), we can fill in everything but the *six unknown parameters* of the diffusion tensor model. As solving a single equation for six unknowns is quite an impossible task, we'll clearly need more of these equations, and by consequence more acquisitions. Mathematically, we know that at least six equations will be necessary to be able to determine the full apparent diffusion tensor. In practice, we'll be needing *at least* six DWIs for *different* gradient directions as well as a single B0 to normalize our measurements to. As stated before, it is essential that every DWI is tied to its respective gradient direction. Acquiring *more* DWIs (for different gradient directions) will lead to more than six equations. In such a setting, no exact solution for the six unknown tensor elements generally exists, because the full system of equations is *overdetermined*. This actually is a good thing! Even though we can find a single exact solution in case of six DWI measurements, this solution will also exactly represent all the *noise* in the data. If we perform the acquisition using a *larger* amount of different gradient directions, a solution will have to be found that *fits* the data as good as possible. We then hope that the part that *doesn't fit* the model (i.e., the *residuals*) is the *noise*, which we (optimally) don't want to model anyway. We'll focus on the issue of tensor fitting later. For the time being, let's take it for granted.

Understanding the Tensor Elements, in Practice

So, effectively applying such a tensor fitting method to our 45 gradient direction dataset at hand, we end up with six numbers in each voxel, i.e., the components which describe an apparent diffusion tensor. Now what can we actually do with it? For starters, let's take a look at some maps of these six tensor elements. There are two distinct categories amongst them: the *diagonal* elements (D_{xx}, D_{yy}, D_{zz}) and the *off-diagonal* elements (D_{xy}, D_{xz}, D_{yz}). The former are presented on the first row of (Fig. 4.9), while the latter are shown on the second row. The interpretation of the *diagonal* elements is straightforward: they represent ADC values along the respective directions of x, y, and z. Because we've shown maps of the original (unfitted) ADC values along these directions in (Fig. 4.8), we can compare them directly to the maps of D_{xx}, D_{yy}, and D_{zz} in (Fig. 4.9). Indeed, they look more or less alike. The more careful observer may note that the latter look less noisy. This is not surprising: they represent fitted values, i.e., all 45 measurements contributed to them. From what we already know, we can even figure out mathematically why e.g. D_{yy} corresponds to the value of the tensor model (i.e. the ADC) along y: just fill in $[0\ 1\ 0]^T$ (i.e., the direction of y) as direction **g** in Eq. (4.6) and evaluate using the right-hand side expression; the outcome trivially equals D_{yy}. We can thus conclude that the *diagonal* elements are in practice meaningful and quite easy to understand. The *off-diagonal* elements (second row of Fig. 4.9), on the other hand, offer a less intuitive source of information. They represent the covariance between each pair of axes (i.e., xy, xz, and yz). That's because the full diffusion tensor is actually a covariance matrix. Apart from being a great conversation starter at an engineering party, those last two sentences won't get you anywhere in daily practice: the *off-diagonal* elements just don't really have a *direct practical use* or meaning. The only reason we do discuss them here, is to emphasize what they *don't mean*: they do *not* represent the values of the ADC along some diagonal direction (so don't mistake them for that!). This should also be clear from the fact that they equally cover a range of positive as well as *negative* values (while ADC values should *not* be negative). One final property worth mentioning though: if all off-diagonal elements are zero, the tensor is perfectly *aligned* to the x-, y-, and z-axes. What that means will become more clear if we visualize the tensor in 3D.

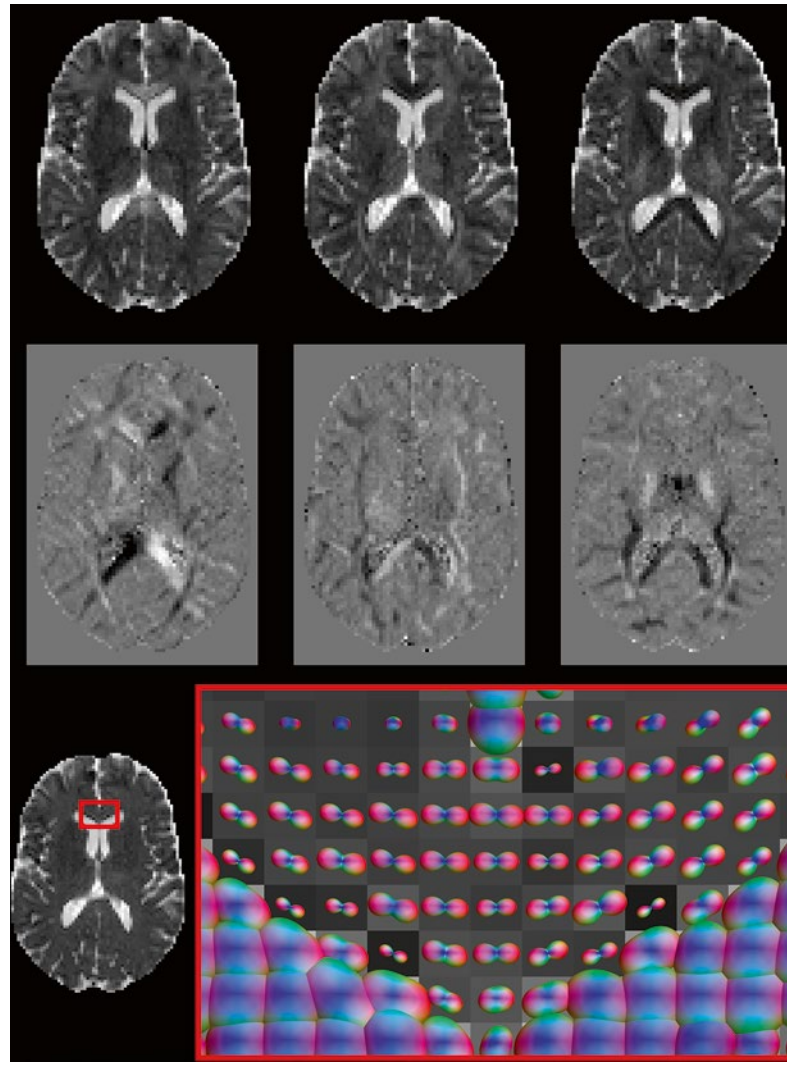


Fig. 4.9 *Top row*: Maps of the diagonal diffusion tensor elements (D_{xx}, D_{yy}, D_{zz}). *Middle row*: Maps of the off-diagonal diffusion tensor elements (D_{xy}, D_{xz}, D_{yz}). The *background grey* level equals zero; darker/brighter levels represent negative/positive values. *Bottom row*: Spherical polar plots of the ADC values provided by the diffusion tensor model in a region of the GCC, overlaid on a map of the average ADC value. Note the characteristic peanut shapes that appear in the GCC

Understanding the Tensor, in Practice: Peanuts!

Talking about visualization, let's take a look at a spherical polar plot of the ADC values actually represented by the fitted diffusion tensors. These are shown for our trustworthy GCC region on the bottom row of (Fig. 4.9), and can again be directly compared to the original (unfitted) values in (Fig. 4.8). From this comparison, it is obvious that the noisy appearance has been greatly reduced: while the original plots showed a unique and different pattern in each voxel (due to the varying noise), the directional profiles that represent the tensor fitted values are much more *consistent* within regions. In regions where a single bundle of axons is present, e.g. the GCC, these plots typically take on the shape of *peanuts*. The advantage of modeling is that some features of interest, such as the *main direction* of the tensor, are recovered more prominently. From the region shown in (Fig. 4.9), it is now also more evident that the nearby CSF in the ventricles exhibits an isotropic pattern of diffusion. And finally, for those who'd like to go just that extra mile in interpretation: consider again one of those curious maps of the *off-diagonal* elements, D_{xy}, and note that it indeed shows a value of *zero* for the voxels right in the middle of the GCC, where the main directions of the tensors are nicely *aligned* to the x-axis.

Conclusions

We have introduced the (apparent) *diffusion tensor* model [4], which is used in DTI to represent the *ADC values* of our measurements along different (gradient) directions. The diffusion tensor **D** is represented by a 3×3 *symmetric matrix*, containing *six unique tensor elements*, and can be evaluated along any direction **g** by the expression $\mathbf{g}^T\mathbf{D}\mathbf{g}$. Casting this expression in the role of the ADC value in the Stejskal-Tanner equation yields the one and only equation at **the *core* of DTI**: it directly relates the experimental parameters, the measurements and the parameters of the diffusion tensor model to each other. This equation can again be rewritten to show clearly that the *diffusion tensor* model is meant to fit the *ADC values*. As there are now *six unknowns* in this equation, the *minimum requirements* for obtaining the diffusion tensor are a B0 (non diffusion-weighted image), at least six DWIs and knowledge of the *b*-value *and* ***gradient directions*** that were used for performing the acquisition of the DWIs. We simply cannot stress enough that knowledge of the *b*-value *and* ***gradient directions***, i.e., the full gradient table as shown in (Fig. 4.6), is absolutely *essential* to fill in the equations and obtain the diffusion tensors! Fitting the tensor model to data with a larger (than six) number of DWIs *reduces the noisy appearance of the ADC values* when visualized as a spherical polar plot. In regions of white matter containing a single consistent bundle of axons, the plot has a characteristic *peanut* shape that clearly shows features such as the *main direction* of the tensor. The *diagonal* elements of the diffusion tensor represent *ADC values along the x-, y-, and z-axes*, while the *off-diagonal* elements represent the covariance between pairs of those axes. Maps of the former thus provide a *meaningful* interpretation, while maps of the latter are neither intuitive nor useful in daily practice. Just *don't mistake the off-diagonal elements* for ADC values along some oblique angle. A final overview of the most important steps taken up to this point is shown for a single voxel in the middle of the GCC in (Fig. 4.10): from raw DWI measurements, to calculated ADC values, and finally the fitted tensor!

Eigenvalues and Eigenvectors

The Tensor Elements: Not Very Practical

Looking back at the ADC peanuts that represent the diffusion tensor values in (Fig. 4.9) and (Fig. 4.10), we notice that the tensor model indeed did a good job in capturing the most important *features* of the angular ADC profile: we can clearly observe its main direction, the maximal ADC value (along this main direction), etc. and how these features relate to each other over larger regions (e.g., qualitatively observe the curving global path of the axon bundle in the GCC). Although these very practical features are in each voxel captured and described by those six unique tensor elements, it's not immediately clear *how*. We do have the diagonal tensor elements (D_{xx}, D_{yy}, D_{zz}) that come with a clear interpretation (i.e., the ADC values along *x*, *y*,

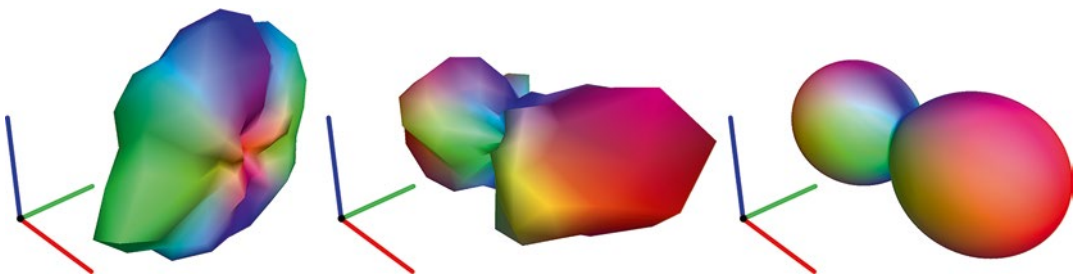


Fig. 4.10 From DWI data to the tensor, for a single voxel in the middle of the GCC. Spherical polar plots of the DWI values (*left*), the ADC values (*middle*) and the ADC values evaluated from the fitted tensor (*right*)

and z). From the ADC peanuts in (Fig. 4.9), we can tell for instance that the value of D_{xx} will coincide with the maximal ADC value of the peanuts right in the middle of the GCC, because those peanuts are nicely *aligned* along x. However, as we move away from that middle region, the orientation of the peanuts changes, causing D_{xx} to gradually take on lower values. The information on the maximal ADC value of the peanuts is now "spread out" somewhere between D_{xx} and D_{yy}. And to make matters even worse, the information on how this "spread" is balanced between those components, is in turn encoded somehow by D_{xy}, one of those elusive off-diagonal tensor elements. That's also why we didn't run into all that trouble right in the middle region: D_{xy} equals zero in that part of the GCC.

Reasoned Wishful Thinking of Alternatives

So, what is at the core of all this confusion and why do we need to be so tedious about trying to infer useful information from the tensor components? The answer is simple: our definition of axes (i.e., x, y, and z) is in fact quite artificial and—more importantly—*very rigid*. To formulate it in another, maybe more clear, way: the axon bundles simply couldn't care less about how we happened to define our *globally fixed axes*; they just happily twist and curve through the full 3D space. On those rare occasions where the tensor perfectly *aligns* to our predefined axes, we get lucky: the off-diagonal elements become zero and the three diagonal components describe the *shape and size* of the tensor in a more direct, intuitive manner. But how do we solve our problem in all those other voxels then? As we just stated, the axon bundles are not going to adjust themselves to our axes; and thus the only solution will be to *adjust our axes* to them in each voxel instead. So, what we are looking for is a new description of the diffusion tensor that provides *a set of axes aligned to the tensor* as well as *three* "*new diagonal tensor elements*" to describe the tensor within this new local set of axes (the "new off-diagonal tensor elements" become zero). The benefits are twofold. These "new diagonal tensor elements" should provide us with everything we need to know about the *shape and size* of the tensor, *independently of its orientation*. On top of that, the customized set of axes by itself also describes the full 3D *orientation* of the tensor. To conclude, such a representation thus effectively *splits up* information about the orientation and the shape/size of the tensor, while the classical six tensor elements *mix it all up*.

Maths to the Rescue: The Eigendecomposition

Now that we know what we want, the question remains how to obtain it. In this case, we are lucky: the mathemagician can help us out with something called *eigendecomposition*. Applied to the diffusion tensor, it basically boils down to rewriting the 3×3 symmetric tensor in the following format:

$$\mathbf{D} = \begin{matrix} \vdots & \vdots & \vdots \\ \epsilon_1 & \epsilon_2 & \epsilon_3 \\ \vdots & \vdots & \vdots \end{matrix} \cdot \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \cdot \begin{matrix} \cdots & \epsilon_1 & \cdots \\ \cdots & \epsilon_2 & \cdots \\ \cdots & \epsilon_3 & \cdots \end{matrix} \tag{4.10}$$

where $\lambda_1 \geq \lambda_2 \geq \lambda_3$, and $\boldsymbol{\epsilon_1}$, $\boldsymbol{\epsilon_2}$, and $\boldsymbol{\epsilon_3}$ are three-element unit vectors that are mutually perpendicular to each other. The right-hand side intuitively reads: start with a tensor (with diagonal elements λ_1, λ_2, and λ_3) aligned to the axes (x, y, and z), and then *reorient* it to a new set of axes ($\boldsymbol{\epsilon_1}$, $\boldsymbol{\epsilon_2}$, and $\boldsymbol{\epsilon_3}$). The process of eigendecomposition aims to *reverse* this set of actions: it starts with the diffusion tensor $\mathbf{D}$, and subsequently tries to figure out which axis aligned tensor could have been reoriented to which new set of axes in order to obtain $\mathbf{D}$. The result is referred to as the *eigenvalues* (λ_1, λ_2, and λ_3) and *eigenvectors* ($\boldsymbol{\epsilon_1}$, $\boldsymbol{\epsilon_2}$, and $\boldsymbol{\epsilon_3}$) of $\mathbf{D}$. They come in so-called *eigenpairs* (e.g., λ_2 is paired to $\boldsymbol{\epsilon_2}$): an eigenvalue (e.g., λ_2) represents the ADC value of the tensor along the direction of the corresponding eigenvector (e.g., $\boldsymbol{\epsilon_2}$). The eigenvector $\boldsymbol{\epsilon_1}$ that is associated

with the largest eigenvalue λ_1 is also referred to as the *principal eigenvector*. It plays quite an important role in DTI: due to its orientation along the peak direction of the ADC peanut, it's indicative of *the local direction of the axon bundle*. While the *largest* eigenvalue λ_1 equals the *maximal* value of the ADC peanut, the *smallest* eigenvalue λ_3 represents its *minimal* value.

Understanding the Eigenvalues, in Practice

Maps of the eigenvalues (λ_1, λ_2, and λ_3) are presented in the top row of (Fig. 4.11). A strict ordering ($\lambda_1 \geq \lambda_2 \geq \lambda_3$) is always enforced. As stated before, the combination of all three eigenvalues fully encodes the exact total *shape and size* of the tensors (and by consequence, the ADC peanuts) by providing the ADC value along three perpendicular axes aligned to the tensors (the eigenvectors). In some regions (e.g., the GCC, or the white matter in general) a larger mutual difference between the eigenvalues can be seen as compared to other regions (e.g., the CSF). This clearly relates to the differing amounts of anisotropy that we could also see in the ADC peanuts. Because all information on the shape and size of the tensors is stored in the eigenvalues, they will also be the basis for other tensor *measures* that are *independent* of the tensor's *orientation*; but we'll get to that later.

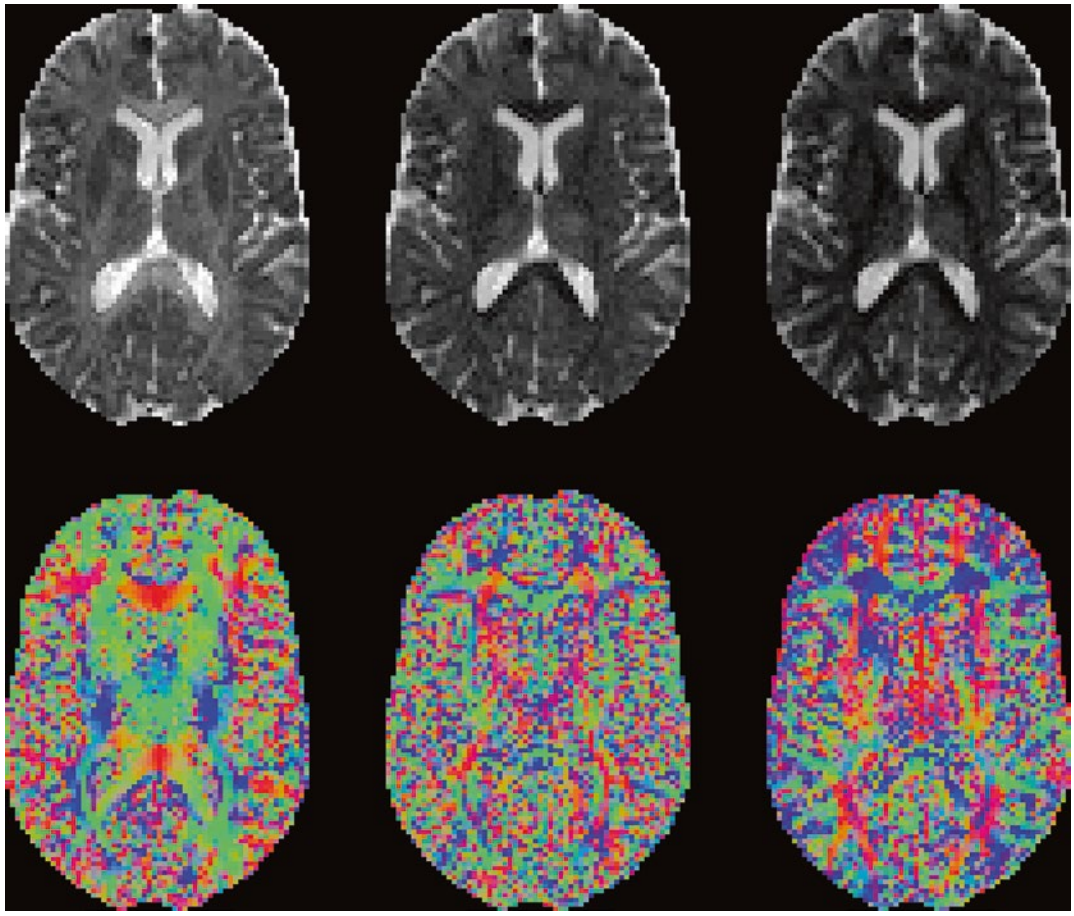


Fig. 4.11 *Top row*: Maps of the eigenvalues (λ_1, λ_2, λ_3). *Bottom row*: Directionally encoded color (DEC) maps of the eigenvectors ($\boldsymbol{\epsilon_1}$, $\boldsymbol{\epsilon_2}$, $\boldsymbol{\epsilon_3}$)

Understanding the Eigenvectors, in Practice

Directionally encoded color (*DEC*) *maps* [5] of the eigenvectors ($\boldsymbol{\epsilon_1}$, $\boldsymbol{\epsilon_2}$, and $\boldsymbol{\epsilon_3}$) are provided in the bottom row of (Fig. 4.11). Since each eigenvector has unit length, no magnitude information is represented in these maps; only *orientation* is encoded. This is achieved by assigning the three elements of an eigenvector to *the red*, *green and blue channels* of a color image. As the eigenvector itself is specified relative to the original (x, y and z) axes, the meaning of the colors is similar to the scheme we used before for displaying spherical polar plots: *red* is linked to x, *green* to y, and *blue* to z. As mentioned before, one of the most important outcomes of DTI is the orientation of the *principal eigenvector* $\boldsymbol{\epsilon_1}$. Within regions of the white matter (e.g., the GCC), the DEC map of $\boldsymbol{\epsilon_1}$ shows a consistent and smoothly evolving pattern that can intuitively be related to the *local orientation of the axon bundles*. In regions such as the CSF, the orientation of $\boldsymbol{\epsilon_1}$ proves to be more or less *random*, resulting in a *noisy* appearance of its DEC map in those particular regions. Associated with the isotropic pattern of diffusion in these regions, we should ideally observe a *spherical* ADC plot (instead of a peanut), satisfying $\lambda_1 = \lambda_2 = \lambda_3$. However, due to random noise in the data, there might be a slight *deviation* from this pattern. The orientation of the principal (and any other) eigenvector is entirely determined by the *random* noise in such a case. Whenever two (or all three) of the eigenvalues of a given tensor are (nearly) equal, we say that the corresponding eigenvectors become *ill defined*.

Conclusions

We introduced the *eigendecomposition* of the diffusion tensor in its *eigenvalues* and *eigenvectors*. The six diffusion tensor components *mix up* information on the shape, size and orientation of the tensor and it becomes hard to untangle the information we're typically interested in by purely intuitive reasoning on these components. Our new representation, however, nicely *separates* information on the *size*/*shape* of the tensor from information on its *orientation*. This is achieved by recovering a set of *axes* (the eigenvectors) that is locally aligned to the tensor as well as three *ADC values* (the eigenvalues) of the tensor along these new axes. The solution thus comes as a set of *eigenpairs*: a certain eigenvalue encodes the ADC along a specific eigenvector. Whereas the eigenvalues encode the size/shape *independently* of the orientation, the eigenvectors describe the orientation *independently* of the size/shape. A final schematic (2D) example, illustrating these properties and providing an overview of the relation between the most important tensor-related numbers we've come across up to this point, is shown in (Fig. 4.12). The eigenvector associated to the *largest* eigenvalue is also referred to as the *principal eigenvector*. Mapping eigenvectors is typically done by use of *directionally encoded color* (*DEC*) *maps*. One of the key practices in DTI consists of mapping the *principal eigenvector*, since it is indicative of the *local orientation of the axon bundles*. In regions of white matter such as the GCC, this map shows a *consistent* pattern. In regions of (nearly) isotropic diffusion such as the CSF, however, the principal eigenvector becomes *ill defined*, leading to a *noisy* appearance of the associated DEC map. Since there are no axon bundles hindering/restricting the diffusion in such a region, the principal eigenvector therein is pretty *meaningless* anyway.

Visualizations, Measures, and Maps

Aiming for Usability: Tensor Glyphs

While all maps and visualizations (using, e.g., spherical polar plots) presented up to this point have provided us with great insight into the underlying information that eventually leads to the diffusion tensors and describes their main features, we are yet to encounter the visualizations and maps that we're most likely to run into when processing DTI data *in practice*. Let's first have a look at the most common 3D visualization of the diffusion tensor, which is not the ADC peanut we've already become acquainted with

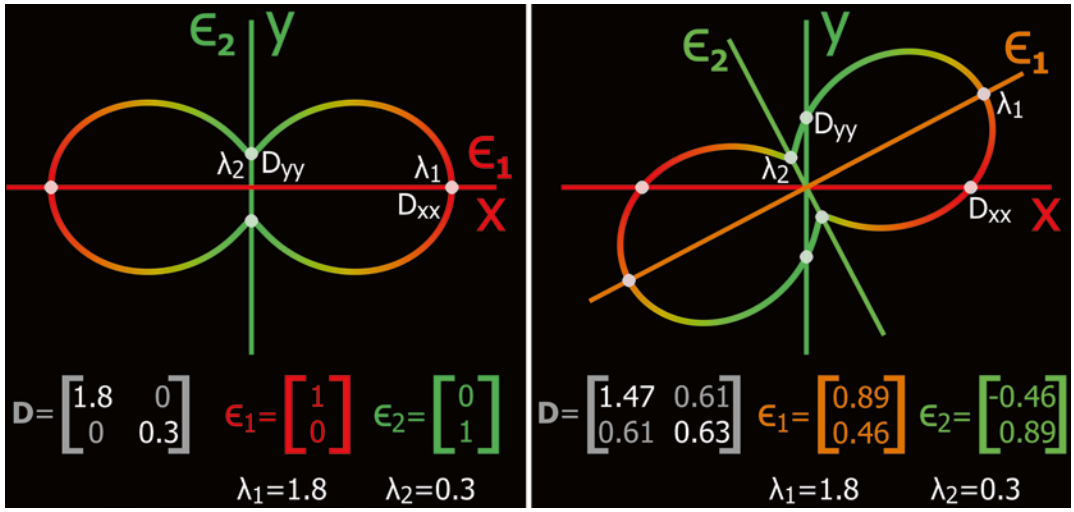


Fig. 4.12 Example of 2D tensors and the convenience of eigenvalue decomposition. *Left*: Perfectly axis aligned tensor. The diagonal tensor elements directly define the tensor shape. The eigenvectors coincide with the global axes and the eigenvalues are equal to the diagonal tensor elements. *Right*: General (not axis aligned) tensor. The diagonal tensor elements encode the value of the ADC peanut along the global axes, yet do not fully define the tensor shape. The eigenvectors provide a new set of axes along which the eigenvalues directly provide the information on the shape

(even though it most directly shows all the values that the tensor represents). Instead, we will only visualize the *most prominent features* that define its size/shape and orientation, as provided by its eigenvalues and eigenvectors: meet the *tensor glyphs*! Two variations are shown in (Fig. 4.13) for our familiar GCC region: *cuboids* and *ellipsoids*. In general, a mostly primitive 3D shape is chosen (e.g., a rectangular cuboid or a scalene ellipsoid) and its three main dimensions are *scaled by the eigenvalues* (or a transformation thereof) and *aligned along the eigenvectors*. Optionally, the glyph is *colored* according to the DEC map of the *principal eigenvector* (i.e. using the map of $\boldsymbol{\epsilon}_1$ in (Fig. 4.11)). Comparing our previous ADC peanuts in (Fig. 4.9) with the newly obtained glyphs in (Fig. 4.13), we can clearly tell the latter score higher on the *usability* scale: they show a more contrasting description of the features that really matter (and still fully define the tensor and thus its ADC peanut). The most commonly visualized glyph shape is the ellipsoid [6], but since a cuboid requires far fewer polygons to be drawn onscreen, it lends itself for faster interaction with larger tensor fields. Interaction with a field of glyphs is useful for a better characterization of their full *3D shapes*: e.g. in (Fig. 4.13), we freely rotated the slice of glyphs to a certain angle. Especially when the axon bundles are not running "in plane", the ability to freely rotate the tensor field is a helpful addition. A reason to prefer ellipsoids (instead of e.g. cuboids) might be that they do not overexaggerate some features of the tensor in cases where those features are not very meaningful or appropriate anyway. A good example is the isotropic diffusion in the CSF, as seen in (Fig. 4.13): the cuboids become cubes, but still clearly indicate the orientation of the eigenvectors, even though they are *ill defined* in this region. The ellipsoids, however, take on the shape of spheres, and thus any visual cues of the eigenvectors inherently *fade away* (apart from the coloration, which is of course also not very informative in this region). Another reason why *ellipsoids* are a meaningful choice is that they actually come with a true *meaning* when scaled using the *square roots of the eigenvalues* [6]: under the model of diffusion that DTI assumes, if we would investigate a single water molecule that starts at the center of the ellipsoid and is allowed to diffuse randomly during a fixed time interval, then there is an equal chance for it to

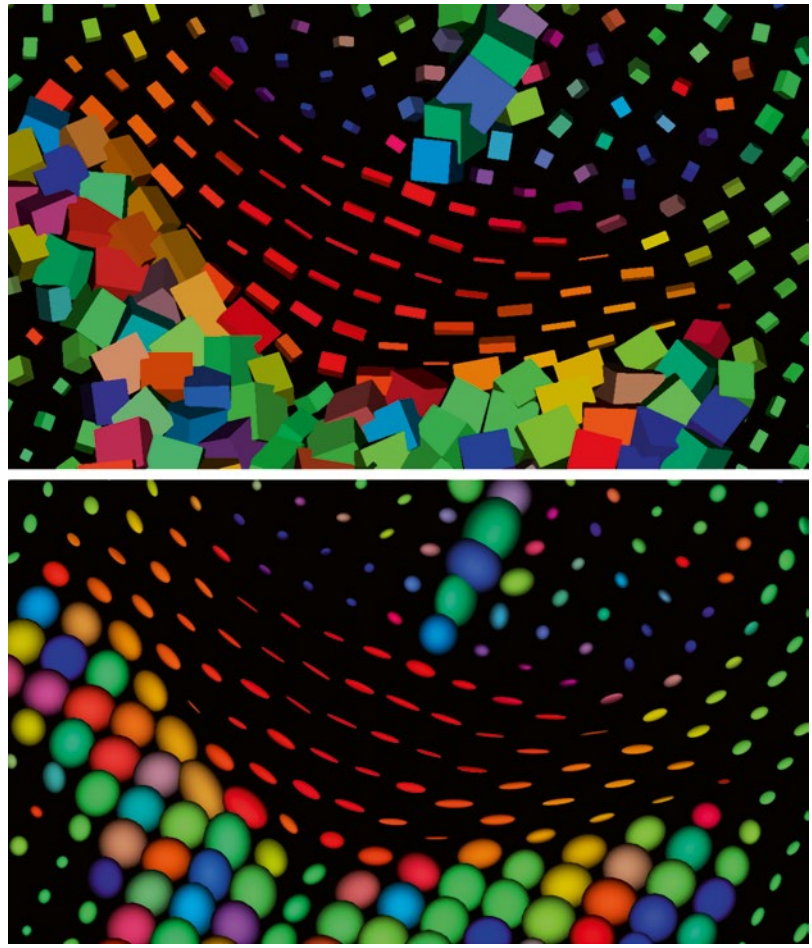


Fig. 4.13 Tensor glyphs (*top*: cuboids; *bottom*: ellipsoids) in a region of the GCC. The glyphs are colored according to the DEC map of the principal eigenvector $\boldsymbol{\epsilon}_1$

displace to any specific point on the surface of this ellipsoid. It might take a few reads of that sentence before one may grasp its meaning, and we won't even go into why it is true; the fact just is that there exists a pretty good reason to prefer these ellipsoids over any other specific glyph! *In practice*, however, *any glyph will do* for exploring the data (even though some software may offer many different options), as long as it's easy on the eyes and the processing power of the machine one is working on.

Mapping Size: Mean Diffusivity (MD) and Friends

Now let's consider some of the more common diffusion tensor measures. All the measures we're about to present are so-called *rotationally invariant* measures: they tell us something about the size or shape of the tensors, *independently* of their orientation. Therefore, they are typically defined in function of the *eigenvalues* of the tensors. Let's start with a straightforward one: the *mean diffusivity* (*MD*) [7]. It is defined as follows:

$$\mathrm{MD} = \frac{\lambda_1 + \lambda_2 + \lambda_3}{3} = \frac{D_{xx} + D_{yy} + D_{zz}}{3} \qquad (4.11)$$

As simply being the *average of the eigenvalues*, it describes the overall *size* of the tensor and as such represents a rotationally invariant ADC measure. A map of it is provided in (Fig. 4.14). The same contrast is sometimes also referred to as the *trace* [7], which equals the *sum* of the eigenvalues. Other related variants exist, such as the pair of *axial diffusivity* (equating to the first eigenvalue) and *radial diffusivity* (equating to the average of the second and third eigenvalues). As

seen in Eq. (4.11), the MD can (surprisingly) also be obtained by averaging the *diagonal tensor elements*: even though these individual elements are dependent on the orientation of the tensor, their average is not. An important warning at this point: this does *not* mean that we can simply acquire three DWIs using perpendicular gradient directions, and subsequently average the three ADCs in order to obtain the *same rotationally invariant MD* [7]! It only applies for three perpendicular ADC values as evaluated from a *tensor model*, so *six gradient directions* are still the bare mathematical minimum in order to account for the anisotropy in the measurements!

Mapping Fractional Anisotropy (FA) and Orientation

Next up is the *fractional anisotropy* (*FA*) [7], probably the most unique selling point of DTI. It is calculated by the following hefty formula:

$$\mathrm{FA} = \sqrt{\frac{3}{2}} \cdot \frac{\sqrt{(\lambda_1 - \bar{\lambda})^2 + (\lambda_2 - \bar{\lambda})^2 + (\lambda_3 - \bar{\lambda})^2}}{\sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}} \quad (4.12)$$

where $\bar{\lambda}$ is the average of the three eigenvalues. In words, this amounts to the *standard deviation* of the eigenvalues *divided by* their *root mean square*. Or, more simply: a measure for how much the eigenvalues *differ*, but *normalized*, so it becomes *independent* of their absolute magnitude. As such, it describes an aspect of the *shape* of the tensor, *independently* of its size (and of course, orientation). Because of the way the formula is carefully normalized, the FA takes on values in an interval *between zero and one*, the former representing perfect *isotropy* (i.e., all eigenvalues are equal) and the latter corresponding to perfect *anisotropy* (e.g., the extreme case where only λ_1 would have a nonzero value). An FA map is provided in (Fig. 4.14). From this, we learn that the white matter clearly has higher anisotropy than any other (healthy) tissue in the brain. As we already know, in regions of low anisotropy, the principal eigenvector's orientation becomes *ill defined*. Hence, the FA map is the perfect candidate to weight the DEC map of the principal eigenvector from (Fig. 4.11): doing so will *hide* the colors in regions where they are *ill defined* (i.e., where they are noisy, confusing, and meaningless). The result is known as the *DEC FA map* [5], and is also presented in (Fig. 4.14). This map isn't the most iconic DTI map for no reason: it's a very handy one and becoming acquainted with the color encoding is key to quickly interpreting a lot of the valuable and unique information in the dataset at once. Mentally processing DEC should become second nature; *red for x* (*left-right*), *green for y* (*back-front*), *and blue for z* (*bottom-top*). Take a look again at the original (grayscale) FA map. Notice how easily we could be tempted to believe that every bundle-like feature of this map represents an in-plane axonal bundle. Now shift your attention back to the DEC FA map, and realize that *blue* stands for an orientation *perpendicular* to the visualized slice. There you have it; that's some *indispensable DEC information* for you!

Exploring Shape Space and Reaching Beyond…

Finally, let's briefly touch upon a triplet of slightly more exotic measures: a *linear* measure (c_l), a *planar* measure (c_p), and a *spherical* measure (c_s) [8]. This is what their formulas look like:

$$c_l = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3} \qquad c_p = \frac{2 \cdot (\lambda_2 - \lambda_3)}{\lambda_1 + \lambda_2 + \lambda_3}$$
$$c_s = \frac{3 \cdot \lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \quad (4.13)$$

All of them are again automatically restricted to an interval of values *between zero and one*. The *sum* of these three measures exactly equals *one*. The full triplet of measures provides the coordinates of our tensor in some "*shape space*": a higher c_l means a more linear, *prolate*, cigar-shaped tensor ellipsoid; a higher c_p means a more planar, *oblate*, pancake-shaped tensor ellipsoid; a higher c_s means a more spherical, *isotropic*, ball-shaped tensor ellipsoid. Just like the FA, these

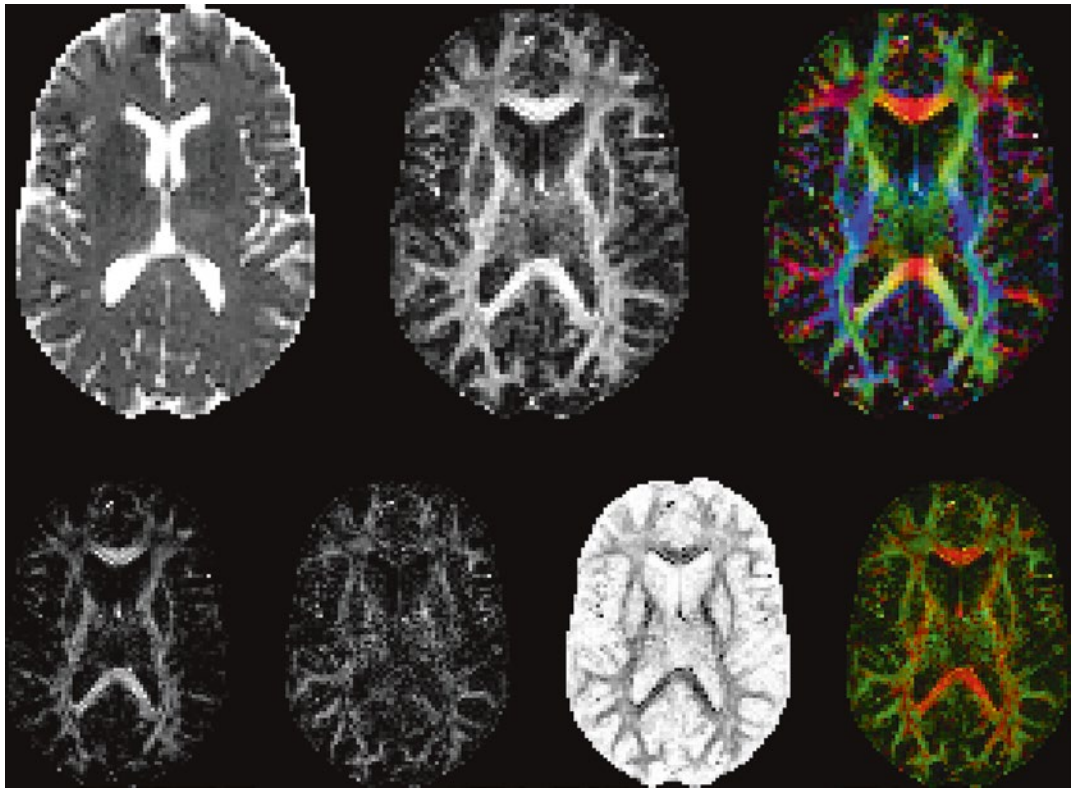


Fig. 4.14 *Top row*: Maps of the mean diffusivity (MD), fractional anisotropy (FA), DEC FA. *Bottom row*: Maps of a linear measure (c_l), planar measure (c_p), spherical measure (c_s), combination of c_l and c_p using *red* and *green* color channels

measures each describe an aspect of the *shape* of the tensor, *independently* of its size (and of course, orientation). One could even use them to come up with new *anisotropy* measures, such as $1-c_s=c_l+c_p$ [8]. Maps of the shape measures are provided in (Fig. 4.14). We also present a map where we employ the red and green color channels to encode c_l and c_p. This final map's absolute intensity thus equals the custom anisotropy measure we just mentioned, while the color somehow shows what "*kind* of anisotropy" is present: *linear or planar*. Interestingly, we notice (in the presented slice) that mostly the central part of the corpus callosum (including the GCC region we have been considering in all our examples) shows highly linear behavior, while *many other regions* of white matter contain a decent portion of *planar diffusion*. Reasoning about the axon bundles as a bunch of cylindrical tubes, as we did before, cannot simply cause such a pattern if all axons in the voxel are coherently running along the same direction: they must be curving or dispersing (within certain planes), or maybe more than one population of axons is present in such voxels. Whatever the underlying situation might be in those voxels, the planarity hints at certain *limitations* of the DTI model ….

Conclusions

We introduced some of the most *mainstream* visualizations and maps that one is bound to run across in daily DTI practice. These include the visualization of the tensors by *glyphs* (most notably the *diffusion tensor ellipsoid*) and maps of the *mean diffusivity* (*MD*), *fractional anisotropy* (*FA*), *DEC FA* as well as a slightly more exotic

triplet of *linear*, *planar and spherical measures* (c_l, c_p, c_s). Other measures exist, but these are typically variations of—or at least heavily inspired by—the ones we presented: axial diffusivity, radial diffusivity, the trace, several variants of anisotropy measures, etc. The most *standard* measures of them all, however, are the typical couple of *MD* and *FA*: the former representing the average *size* of the tensor (independent of shape and orientation) and the latter encoding its *anisotropy*, an aspect of the *shape* of the tensor (independent of size and orientation). Information on the *orientation* can also be included by *combining* the FA map and a DEC map of the principal eigenvector in order to obtain *the DEC FA map*. This map is not only iconic for DTI, but it's also a very handy tool to quickly gain insight into any DTI dataset.

Tensor Fitting Methods

Facing the Issue

Up to now, we've been taking an essential step in the whole process for granted: the actual *tensor estimation*. As a matter of fact, this is the *least trivial* step along the pipeline; everything else we've discussed up to this point simply consists of applying some *well-defined* and quite straightforward formulas in the right order. Even the eigendecomposition—or at least what we desire it to yield for an outcome—is exactly defined (i.e., Eq. 4.10; no more, no less), and we can rely on computer science to provide us with an algorithm that does the job. In those cases where the outcome proved to be ill defined (e.g., isotropic diffusion), the ill defined parts of the outcome (i.e., the eigenvectors) were not informative anyway. But the *tensor estimation* … that's an entirely different beast! Different, because this time even the definition of "*what we want*" is not all that clear. Or is it? We simply want the tensor to *fit* the data (or the ADC values …?) *as good as possible*, right? But what is "*as good as possible*"? It's *vague*, that's what it is; and hence, a plethora of definitions and associated fitting methods exist. While it easily provides enough material to write a decent book on the subject alone, consider the following as your average quick and dirty hitchhiker's guide to making the right choice. In the foreign restaurant of tensor fitting, it should enable you to more or less translate the menu, pick something that you're not allergic to, all the while giving you the confidence that your choice will leave you satisfied up to a certain level, but also allow you to leave the restaurant in time, so you can still catch your bus. How they actually arrange stuff in the kitchen though, is the least of our concerns. Have a seat; the daily menu consists of: *linear least squares* (*LLS*), *weighted linear least squares* (*WLLS*), and *nonlinear least squares* (*NLS*). On top of that, there's also today's special: *robust estimation of tensors by outlier rejection* (*RESTORE*). Now let's have a look at our advice!

LLS: Quick and Dirty

Linear least squares (*LLS*) is the most basic choice. It will solve your system of equations—of which each single one takes on the form of Eq. (4.9)—by *minimizing the sum of squared residuals* of those equations. In the case where we have more than six equations (as opposed to only six unknowns), it is typically impossible to perfectly satisfy all equations. The error or difference that still exists between the left- and right-hand sides of one such equation is referred to as its *residual*. In practice, minimizing the sum of the squared residuals of all the equations amounts to "*spreading out*" those unavoidable residuals as much as possible *over all of them*. This is equivalent to stating that each equation has an "*equal say*" in the process of the fit. A great advantage of LLS is that it can be implemented as a *single-step process*. That's right: it just takes a single specific operation on the whole system of equations to automatically obtain the solution that optimally minimizes that sum of squared residuals. Depending on your hardware and the size and number of DWIs in the dataset, you can have your tensors rolling out in mere *seconds*! In fact, the tensors we generated in this chapter—and thus all the resulting visualizations and maps

we've been looking at—are the result of a single quick application of LLS. For that specific purpose, LLS is certainly well suited: *qualitatively* speaking, our tensors and the subsequently calculated maps of tensor measures look *perfectly fine*. So, why don't we just stick with LLS for all intents and purposes then? The *problem* is subtle and quite well hidden: it actually concerns the fact that we allowed each equation to have *an equal say* in the fit. This assumes that each value that we are trying to fit, was provided to us with an error (i.e., the measurement noise) of a "similar magnitude". In more professional words and adapted to how we specifically formulated Eq. (4.9): LLS assumes that the noise *on the ADC values* (i.e., the right-hand side of each equation, which we are trying to fit with the left-hand side) results from a distribution that has the *exact same variance* for all our different ADC values. While this is the case for our DWI values (that originate directly from the scanner, where the noise is "officially" added to the measurements), it does *not* apply to the ADC values: the distribution of the noise on the original data is logarithmically transformed along with those data to obtain the ADC values! You might remember that we mentioned this before, additionally stating that it would happen to "bug us later on". So now, here it is: officially bugging us. The problem at hand is that each "measured" (i.e., calculated) ADC comes with noise of a *different* variance: i.e., we can trust some ADCs more (or less) than others. It seems sensible to *weigh* the amount of say of each equation in the fit with this information. That's where *weighted linear least squares* (*WLLS*) kicks in.

WLLS: Still Quick, Less Dirty

Weighted linear least squares (*WLLS*) assigns to each equation (still of the form of Eq. 4.9) a *weight* according to how much the original noise variation is affected by the logarithmic transform of the data. These weights directly *depend on the magnitudes of the original data* (i.e., the intensity of the different DWIs). In practice, once these weights are known, only a limited modification has to be made to LLS to take them into account and obtain a WLLS fit, which now truly provides us with the optimal correct fit! It still only takes a single (slightly bigger) operation on the whole system of equations to get this solution: it might take a few extra seconds, but it still just remains a matter of mere *seconds* to have your tensors again rolling out; yet much more *accurately*. If we would have generated all the maps in this chapter based on a WLLS fit of the tensors, you wouldn't have noticed the difference: it doesn't suddenly change the *visually* informative contrast of those maps. For *quantitative* purposes (such as group studies) though, it certainly matters, *a lot*: WLLS already *removes* a great deal of inherent *biases* on final measures (such as MD and FA) that are typically caused by careless use of LLS. So, why don't we just stick with WLLS for all intents and purposes then? Again, a sneaky problem manifests itself: to determine the weights, we need the magnitudes of the original data … *without the noise*. Of course, once we have obtained a fitted tensor, we could reason that we got rid of the noise (because, optimally, only the noise is left in the residuals). We could then evaluate that tensor for all gradient directions and calculate from the ADCs back to the DWIs, i.e. the *noiseless* magnitudes that we needed to determine the weights. So, if we could obtain a fitted tensor, then we would also have our weights; but in order to obtain a fitted tensor, we need those weights in the first place. Yes, that's a *chicken-and-egg problem* we're facing here. No perfect solution exists (and thus, unfortunately, also *WLLS can never be perfect*). A first approach could be to just use the magnitudes of the original noisy data to determine the weights. This may, however, result in a *worse* outcome as compared to using plain old LLS! A second trick is to start by performing *a LLS fit*, and get DWI magnitudes from this fit (where the noise should then already be accounted for up to a great extent) *in order to determine the weights for a subsequent WLLS fit*. One could then even repeat this process in the hope of getting gradually better fits and subsequent weights for the next fit (but this typically does not add much: most of the "magic" is in using that first LLS just for a robust set of

weights). In practice, *it's all still fast*: a LLS followed by a WLLS. The *danger* lies in the fact that some software packages might perform WLLS using the first approach, yielding worse results as compared to LLS. On the other hand, a responsible implementation of WLLS using the second approach should *definitely lead to better results*, without any significant increase in computation time as compared to LLS: it's typically still done in mere *seconds*! However, we can never truly know the correct weights due to the chicken-and-egg format of the problem: so the approach doesn't fix everything. The core of the *original* problem was that the noise got *logarithmically* transformed in the ADC values, and that we formulated Eq. (4.9) based on an ADC value in the left and right hand side of the equation. Knowing now that we actually want to compare the values of the original signals, can't we modify that equation so it compares stuff *that isn't logarithmically transformed*? Easy enough: just remove the logarithm by taking the *exponential* of both sides! Now we are facing the correct form of the equation, but sadly, it also lost its linearity in the unknowns: the linear sum of these unknowns on the left hand side now appears under that *exponential function*. Long story short: it's a nonlinear equation. That's where *nonlinear least squares* (*NLS*) kicks in.

NLS: A Long and Brave Quest in the Mountains

Nonlinear least squares (*NLS*) will try to solve an overdetermined system of *nonlinear* equations, again aiming to *minimize the sum of squared residuals* of those equations. Going into details about this one is nigh impossible: many methods exist. They all share a common thing, though: they take a much, much *longer time* to reach a solution as compared to LLS and WLLS. They are basically facing the fiendishly difficult problem of finding the lowest point in the lowest valley of a mountainous landscape in a six-dimensional world. Actually, LLS and WLLS also did, but due to the specific simple shape of the landscape when the equations are linear, they could come up with a nifty trick of finding that lowest point in *a single step*. NLS, on the other hand, is just dropped somewhere on the landscape and has to start a *walk* in the unprepared hitchhikers fashion: without a map (because the landscape is too big and complex) and just relying on its eyes and feeling to *gradually* move to lower regions. In theory, truly solving the NLS problem will yield the optimal result. However, the problem is not easy to solve. Due to the *limited range of sight* in the mountains, an NLS algorithm might get stuck in a *suboptimal* valley (not knowing there exists another lower valley somewhere). Some algorithms are more *robust* against this than others, but it's nearly impossible to come up with an algorithm that *never* makes these mistakes. In general, many NLS algorithms exist that will in most cases further *outperform WLLS*. In some specifically challenging voxels though, such an algorithm *might fail to converge* or get stuck in the previously mentioned *suboptimal* valleys. If and when the algorithm might detect this, it could for instance perform a WLLS fit instead (still better than nothing or something really wrong, right?). Because NLS algorithms are forced to take a walk in the mountains anyway, they may also come with extra bells and whistles allowing them to generate a solution that specifically satisfies some *constraints*. Due to noise in the data, LLS and even WLLS can sometimes come up with tensors that have one or more *negative eigenvalues*. Of course, this doesn't make sense: negative eigenvalues, and thus negative ADCs, have *no physically sensible meaning*. An NLS algorithm can be guided to not encounter such unwanted cases in the first place: barriers can be put up on the landscape in order to simply deny the NLS hitchhiker access to these forbidden areas. This all typically does come at an extra computational cost, and thus your valuable *time*. Depending on your hardware, the size of the dataset and the kind of NLS algorithm (and the bells and whistles it might come with), some of these strategies may take anywhere from a few *minutes* to several *hours* to finish calculating your tensors. Using a *brain mask* (so no unnecessary calculations are performed for voxels outside of the brain) is typically strongly advised to

reduce running time. And guess what? Even if the hitchhiker would be so extremely experienced that he would always find the lowest point in the landscape, his optimal NLS solution could still be unsatisfying. That's because the data aren't only messed up by noise, but possibly also by *outliers*! Motion, distortions, cardiac pulsation, signal dropout, ghosting … *artifacts* are abundant in MRI. Some can be avoided during acquisition, others can be partially dealt with by preprocessing, but in the end some *still leave their mark* on the data when we offer it to our favorite tensor fitting method. They cause *outliers*: data points that have lost all of their informative value by taking on truly silly values that *don't fit the picture*, at all. That's where *robust estimation of tensors by outlier rejection* (*RESTORE*) kicks in.

RESTORE and Beyond: Expecting the Unexpected

Robust estimation of tensors by outlier rejection (*RESTORE*) [9], as its name suggests, will handle outliers by *rejecting* them. To reject them, they first have to be *detected* though. To do this, it will start with a NLS fit. It will subsequently assign each measurement a *weight*, depending on how well it fits the picture. Another NLS is performed, where each equation is weighted according to how well its measurement fit the picture before. This process is *repeated until convergence*. The final weights should now be a reliable measure for how well each measurement does (not) fit in, i.e., for its "*outlier-ness*". Those measurements that meet a certain threshold are officially regarded as *outliers*, and simply kicked out of the game. The *final fit* is then performed by employing only the surviving "*non-outlier*" data. While this is an ingenious and very *robust* strategy, it does have a few implications. A first one is the fact that it might have to perform *several* subsequent NLS fits: that will surely have an impact on the total computation *time*. It could on average take more than three times as long as compared to a single NLS fit [9]. A second one is the fact that, after kicking out a possibly decent amount of outliers, *enough* measurements should of course still be left to reliably obtain the final fit. Those measurements are even needed to actually reliably classify the other ones as outliers in the first place. Hence, data *redundancy* is an important requirement. Even very recently, further improvements have still been made to relax that redundancy requirement up to a certain extent [10]. Given that the DTI model is about 20 years old now, this certainly proves that the fitting problem still remains *far from trivial*.

Conclusions

We took a bite out some of the most common tensor fitting methods: *linear least squares* (*LLS*), *weighted linear least squares* (*WLLS*), *nonlinear least squares* (*NLS*), and *robust estimation of tensors by outlier rejection* (*RESTORE*). It is typically said that this specific ordering is one of *increasing complexity*, implying *increasingly better results* at the cost of an even steeper *increase of computation time* (especially for the nonlinear methods). This is generally true; provided that each variant is implemented as good as possible (we rely on the responsibility of the software developers here). If your dataset has enough data *redundancy* (let's say, DWIs for more than 30–40 unique gradient directions [10]), we could easily always advise you to use *RESTORE*. However, depending on the specific implementation of RESTORE, the hardware, the size or even number of datasets you have to process, etc. it might take quite a while (possibly up to *several hours*) before you have access to your tensors for further processing. All the bells and whistles in these advanced nonlinear algorithms may not be necessary, if you're just concerned about having a quick *qualitative* look at the data. For *quantitative* purposes though, we certainly advice to go "*beyond LLS*." A very big gain is already achieved by WLLS (if implemented responsibly), at a minimal extra computational cost. Certainly be on the lookout for the method your favorite piece of DTI software is packing, or even what different choices it might be offering; as you now speak and understand some basic tensor fitting language!

Final Conclusions

In this chapter, we provided an overview that took us all the way from the raw DWI data to the diffusion tensor and even further to some of the more common visualizations and measures. This fact by itself makes for the most important conclusion: while the more "classical" imaging modalities (e.g., T1, T2) are obtained straight from the scanner, the maps that are typically employed in the practice of DTI (e.g., MD, FA, DEC FA) result from a *postprocessing* pipeline: i.e. *these maps are calculated*, *not directly acquired*. Most of this pipeline is clearly defined; but for the actual *tensor fitting*, there are *quite a few options*. The more advanced methods may also take a reasonable time to be computed. Some *scanner software* offers the option to directly show and export MD, FA, DEC FA, and even other maps; however, *don't let that fool you*: this software still has to go through all the steps we've come across in this chapter. Also, if the software almost instantly provides you with e.g. a DEC FA map, you should now be aware that it may probably not have performed much more than a simple LLS fit (which might of course be sufficient, if you're just qualitatively inspecting the data). The scanner software also has to rely on the same DWI dataset for this, and thus is *not any more or less reliable* in general than any other piece of software if it comes to providing you with accurate maps: if you do use its features, certainly also try to find out what (tensor fitting) algorithms it employs under the hood! If you want to take advantage of the plethora of different available (*freeware*) *software packages* that implement several advanced tensor fitting methods (and further postprocessing steps, such as fiber tractography), you'll need to *export* the raw DWI data from your scanner. We've also stressed at several occasions that these images are quite worthless if they don't come with the *accompanying gradient directions and b*-value*s*. More and more manufacturers are starting to take this into account and tuck that information safely away in, e.g., the DICOM headers, the headers of their own proprietary formats, or even in separate files (containing a *gradient table* in one way or another). However they do it, just try to somehow make sure that it is effectively packed with your data. Your next concern then is to get it *imported* correctly into your DTI software package. Unless that package supports a whole list of different (more and less) standards, you might be up for yet another daunting task. We're lucky up to some extent, however, as the "*diffusion community*" and the specific supporting communities revolving around some software packages are often very active and responsive: your specific question could be answered quickly after a simple e-mail to a support mailing list. Once you get your workflow up and running, the use of DTI in your daily practice should provide you with *new and exciting insights*!

References

1. Stejskal EO, Tanner JE. Spin diffusion measurements: spin echoes in the presence of a time-dependent field gradient. J Chem Phys. 1965;42:288–92.
2. Le Bihan D, Breton E. Imagerie de diffusion in vivo par résonance magnétique nucléaire. C R Acad Sci Paris. 1985;301(Série II):1109–12.
3. 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;161:401–7.
4. Basser PJ, Mattiello J, Le Bihan D. Estimation of the effective self-diffusion tensor from the NMR spin echo. J Magn Reson Ser B. 1994;103:247–54.
5. Pajevic S, Pierpaoli C. Color schemes to represent the orientation of anisotropic tissues from diffusion tensor data: application to white matter fiber tract mapping in the human brain. Magn Reson Med. 1999;42:526–40.
6. Basser PJ, Mattiello J, Le Bihan D. MR diffusion tensor spectroscopy and imaging. Biophys J. 1994;66:259–67.
7. Basser PJ. Inferring microstructural features and the physiological state of tissues from diffusion-weighted images. NMR Biomed. 1995;8:333–44.
8. Westin CF, Peled S, Gudbjartsson H, Kikinis R, Jolesz FA. Geometrical diffusion measures for MRI from tensor basis analysis. Proc Intl Soc Mag Reson Med. 1997;5:1742.
9. Chang LC, Jones DK, Pierpaoli C. RESTORE: robust estimation of tensors by outlier rejection. Magn Reson Med. 2005;53:1088–95.
10. Chang LC, Walker L, Pierpaoli C. Informed RESTORE: a method for robust estimation of diffusion tensor from low redundancy datasets in the presence of physiological noise artifacts. Magn Reson Med. 2012;68:1654–63.

Suggested Reading

Basser PJ, Özarslan E. Anisotropic diffusion: from the apparent diffusion coefficient to the apparent diffusion tensor. In: Jones DK, editor. Diffusion MRI: theory, methods, and applications. New York, NY: Oxford University Press; 2010.

Kingsley PB. Introduction to diffusion tensor imaging mathematics: Part I. Tensors, rotations, and eigenvectors. Concept Magn Reson A. 2006;28: 101–22.

Kingsley PB. Introduction to diffusion tensor imaging mathematics: Part II. Anisotropy, diffusion-weighting factors, and gradient encoding schemes. Concept Magn Reson A. 2006;28:123–54.

Kingsley PB. Introduction to diffusion tensor imaging mathematics: Part III. Tensor calculation, noise, simulations, and optimization. Concept Magn Reson A. 2006;28:155–79.

Jones DK, Basser PJ. "Squashing peanuts and smashing pumpkins": how noise distorts diffusion-weighted MR data. Magn Reson Med. 2004;52:979–93.