Numer. Math. Theor. Meth. Appl.
doi: 10.4208/nmtma.2013.mssvm08

Vol. **6**, No. 1, pp. 138-168
February 2013

Numerical Schemes for Linear and Non-Linear Enhancement of DW-MRI

Eric Creusen[1,*], Remco Duits[1,2,*], Anna Vilanova[2] and Luc Florack[1,2]

[1] *Department of Mathematics and Computer Science, CASA Applied Analysis, Eindhoven University of Technology, P.O. Box 513, 5600 MB, Eindhoven, The Netherlands.*
[2] *Department of Biomedical Engineering, BMIA Biomedical Image Analysis, Eindhoven University of Technology, P.O. Box 513, 5600 MB, Eindhoven, The Netherlands.*

Received 6 December 2011; Accepted (in revised version) 11 October 2012

Available online 11 January 2013

Abstract. We consider the linear and non-linear enhancement of diffusion weighted magnetic resonance images (DW-MRI) to use contextual information in denoising and inferring fiber crossings. We describe the space of DW-MRI images in a moving frame of reference, attached to fiber fragments which allows for convection-diffusion along the fibers. Because of this approach, our method is naturally able to handle crossings in data. We will perform experiments showing the ability of the enhancement to infer information about crossing structures, even in diffusion tensor images (DTI) which are incapable of representing crossings themselves. We will present a novel non-linear enhancement technique which performs better than linear methods in areas around ventricles, thereby eliminating the need for additional preprocessing steps to segment out the ventricles. We pay special attention to the details of implementation of the various numeric schemes.

AMS subject classifications: 65M10, 78A48

Key words: DTI, DW-MRI, scale spaces, finite differences, convection-diffusion, adaptive diffusion, Perona-Malik diffusion, Lie groups.

1. Introduction

Diffusion-Weighted Magnetic Resonance Imaging (DW-MRI) is an MRI techniques for non-invasively measuring local water diffusion inside tissue. It has been stipulated that the water diffusion profiles of the imaged area allow inference of the underlying tissue structure. For instance in brain white matter, diffusion is less constrained parallel to nerve fibers

*Corresponding author. *Email addresses:* `e.j.creusen@tue.nl` (E. J. Creusen), `r.duits@tue.nl` (R. Duits) than perpendicular to them and so the water diffusion gives information about the fiber structures present. This allows for the extraction of anatomical information concerning biological fiber structures from DW-MRI scans.

The diffusion of water molecules in tissue over some time interval t can be described by a diffusion propagator which is the probability density function $\mathbf{x} \mapsto p_t(X_t = \mathbf{x}+\mathbf{r} \mid X_0 = \mathbf{x})$ of finding a particular water molecule at time $t \geq 0$ displaced by $\mathbf{r} \in \mathbb{R}^3$ from its initial position $\mathbf{x} \in \mathbb{R}^3$ at $t = 0$. Here the family of random variables $(X_t)_{t\geq 0}$ describes the distribution of water molecules over time. The function p_t can be related to MRI signal attenuation of diffusion weighted image sequences through the Fourier transform and so can be estimated given enough measurements [28]. The exact methods to do this are described by e.g., Alexander [2].

The most common DW-MRI scans is Diffusion Tensor Imaging (DTI), which measures for each position a 3×3, symmetric positive definite tensor called the diffusion tensor [3]. DTI makes the assumption that locally the diffusion propagator is given by an anisotropic Gaussian function, characterized by this diffusion tensor. Other, more recent techniques collectively called High Angular Resolution Diffusion Imaging (HARDI), allows more general shapes for the diffusion propagator [8, 32].

Often, p_t is not reconstructed completely from measurements, since this would involve long scanning times. Instead, the Orientation Distribution Function(ODF) can be obtained using less measurements [1]. The ODF gives the probability density that a water particle diffuses in a certain direction $\mathbf{n}$, regardless of distance traveled. It is defined by:

$$\mathrm{ODF}(\mathbf{x},\mathbf{n}) = \int_0^\infty p_t(X_t = \mathbf{x} + \alpha\mathbf{n} \mid X_0 = \mathbf{x})\alpha^2 d\alpha. \tag{1.1}$$

This ODF is an example of a function of position and orientation. A general function $U : \mathbb{R}^3 \times S^2 \mapsto \mathbb{R}^+$ of positions and orientations can be visualized by a field of surfaces

$$S_\mu(U)(\mathbf{x}) = \{\mathbf{x} + \mu U(\mathbf{x},\mathbf{n})\mathbf{n} \mid \mathbf{n} \in S^2\} \subset \mathbb{R}^3, \tag{1.2}$$

which are called glyphs. A figure is generated by visualizing all these surfaces for different, sampled $\mathbf{x}$ and with a suitable value for $\mu > 0$ that determines the size of the glyphs. An example of such a visualization can be found in the top left of Fig. 1. Note that for DTI data a different visualization based on ellipsoids is commonly used. Such an ellipsoid visualization can not handle our enhanced DW-MRI images, which can represent crossings, which explains our choice for visualization. To reduce noise and to infer information about fiber crossings, contextual information can be used [25, 26]. This enhancement is useful both for visualization purposes and as a preprocessing step for other algorithms, such as fiber tracking algorithms, which may have difficulty in noisy or incoherent regions. Recent studies indicate the increasing relevance for enhancement techniques in clinical applications [5, 19, 21, 30, 31].

We perform the enhancements on modeled DWI images. We use two models: diffusion tensors and orientation distribution functions. We want to stress that the algorithms can be applied to any other model, as long as it can be converted to a function of positions and orientations. We will use the term DWI data in this paper to refer to both these models.

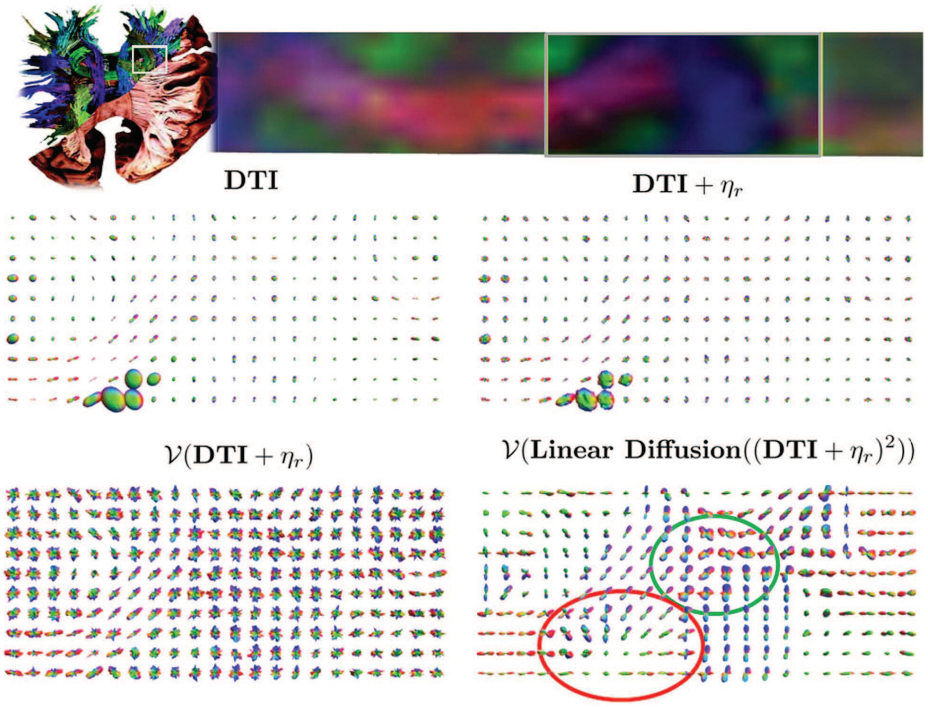

DTI
DTI + η_r
$\mathcal{V}(\mathbf{DTI} + \eta_r)$
$\mathcal{V}(\mathbf{Linear\ Diffusion}((\mathbf{DTI} + \eta_r)^2))$

Figure 1: Top row: DTI data of the corpus callosum and corona radiata visualized by Eq. (1.2) without (left) and with Rician noise (right). Bottom row: the same data with a normalization defined in Section 7.2 (left) and the result of linear enhancement (right). The red circle shows areas in which the ventricles induce anatomically incorrect crossing structures in the surrounding fibers. The green circle shows anatomically expected crossings between the corpus callosum and corona radiata.

These enhancements, done in this paper with left-invariant linear and nonlinear adaptive convection-diffusion processes, is the main focus of this paper. An example of the denoising properties of a linear enhancement process can be found in Fig. 1, in which most of the artificially added noise has been removed. The benefit of this framework is that it performs diffusion both on positions and orientations at the same time. This allows it to handle crossing structures in a natural way, since crossing fibers are separated from each other in the domain. Other methods often use spatial and angular diffusion separately [5, 15, 16], thereby having more difficulties handling complex fiber structures such as crossing fibers. Special attention is given to implementation of these algorithms through the use of finite difference schemes.

Linear convection-diffusion processes have some disadvantages. One is that linear diffusion can occur across regions where the gradient is very large. In particular, the neural tracts of the brain are sometimes located near the ventricles of the brain. These ventricles are structures that contain cerebrospinal fluid which shows up in DW-MRI as unrestricted, isotropic diffusion profiles much larger in magnitude than the restricted, anisotropic diffusion profiles of the neural tracts. It is undesirable that these large isotropic diffusion profiles start to interfere with the oriented structures of the neural tracts when we apply a diffusion scheme, because they are likely to destroy the fiber structures. This effect can be seen in the bottom right of Fig. 1.

By means of the introduced nonlinear left-invariant diffusion scheme, we prevent diffusion from the ventricles to the fibers. This eliminates the need for additional preprocessing steps to reduce the effect of isotropic areas (such as the ventricles or the grey matter) on the rest of the data, as was done in e.g., our previous work [22]. This preprocessing typically involves segmenting out these areas by hand or with hard thresholds (on diffusivity or FA) or by registration with anatomical (T1 or T2 weighted) images.

The structure of this paper is as follows: in Section 2 we will briefly discuss the basics of the theory and of notation. Subsequently in Section 3 we will discuss linear convection-diffusion processes, paying special attention to discretization and finite difference schemes. In Sections 4 and 5 we will elaborate on two special cases of linear convection-diffusion processes. After that, in Section 6, we introduce a novel nonlinear diffusion scheme. We conclude our paper with experiments on real DTI data from healthy volunteers in Section 7.

2. Preliminaries

2.1. The Euclidean motion group $SE(3)$

For the enhancement of data, it is crucial to determine which fiber fragments are well aligned with each other. To enhance (neuronal) fibers, it is vital that fiber fragments that are well aligned exchange information. To find out which fiber fragments are well aligned, you must consider the spatial and orientational information together, as they interact with each other. Fig. 2 shows how $(\mathbf{x}_0, \mathbf{n}_0)$ and $(\mathbf{x}_1, \mathbf{n}_1)$ are better aligned (by forming a more natural connecting curve) than $(\mathbf{x}_0, \mathbf{n}_0)$ and $(\mathbf{x}_2, \mathbf{n}_1)$ are, even though both pairs have the same spatial distance from each other and the same difference in angles. This shows us that the space of positions and orientations is coupled and it is therefore conceptually wrong to consider the space of positions and orientations as a flat Euclidean space.

To align fiber fragments one needs rigid body motions. The non-commutative group product of rigid body motions induces the required coupling between positions and orientations. Unfortunately, $\mathbb{R}^3 \times S^2$ does not form a group[†] and thus we need to embed it into a larger space with group structure. For this we use the *Euclidean motion group*, i.e., the group of rigid body motions $SE(3) = \mathbb{R}^3 \rtimes SO(3)$, where $SO(3)$ represents the (noncommutative) group of 3D rotations defined as a matrix group by

$$SO(3) = \{\mathbf{R} \mid \mathbf{R} \in \mathbb{R}^{3\times 3}, \mathbf{R}^T = \mathbf{R}^{-1}, \det(\mathbf{R}) = 1\}.$$

Expressed in Euler angles, this becomes

$$\mathbf{R}_{(\alpha,\beta,\gamma)} = \mathbf{R}_{\gamma}^{\mathbf{e}_x} \mathbf{R}_{\beta}^{\mathbf{e}_y} \mathbf{R}_{\alpha}^{\mathbf{e}_z}, \tag{2.1}$$

where $\mathbf{e}_1 = \mathbf{e}_x$, $\mathbf{e}_2 = \mathbf{e}_y$ and $\mathbf{e}_3 = \mathbf{e}_z$ are the unit vectors in the Cartesian coordinate frame and $\mathbf{R}_{\alpha}^{\mathbf{e}_i}$ denotes a counterclockwise rotation of α around vector $\mathbf{e}_i$. Here, an Euler

[†]S^2 is not a proper Lie group, since if it were its left invariant vector fields would violate Brouwer's theorem/Hairy Ball theorem [12]

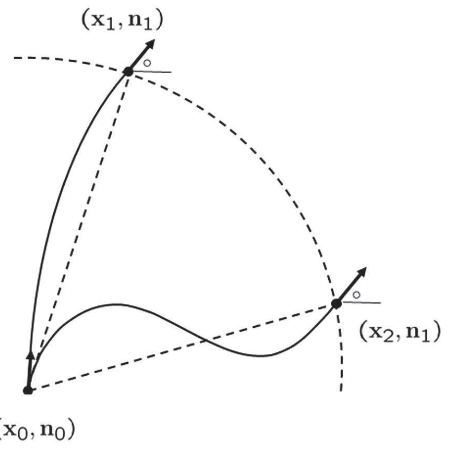

$(\mathbf{x}_1, \mathbf{n}_1)$
$(\mathbf{x}_2, \mathbf{n}_1)$
$(\mathbf{x}_0, \mathbf{n}_0)$

Figure 2: A figure demonstrating the coupling of position and orientation. $(\mathbf{x}_0, \mathbf{n}_0)$ is better aligned with $(\mathbf{x}_1, \mathbf{n}_1)$ than with $(\mathbf{x}_2, \mathbf{n}_1)$ despite having equal spatial distance and the same angle between vectors.

angle parametrization is used that has a discontinuity at $\mathbf{n} = (\pm 1, 0, 0)$, so that the tangent space of $SE(3)$ is well parameterized at the unity element $(\mathbf{0}, \mathbf{I})$ corresponding to element $(\mathbf{0}, \mathbf{e}_z) \in \mathbb{R}^3 \rtimes S^2$. Standard Euler angle parametrization, such as used in [10] do not lead to well defined tangent space in the identity element, which explains our choice for this Euler Angle parametrization.

For $g = (\mathbf{x}, \mathbf{R}) \in SE(3)$ and $g' = (\mathbf{x}', \mathbf{R}') \in SE(3)$ the group product and inverse element are given by

$$\begin{aligned} gg' &= (\mathbf{x} + \mathbf{R} \cdot \mathbf{x}', \mathbf{R}\mathbf{R}'), \\ g^{-1} &= (-\mathbf{R}^{-1}\mathbf{x}, \mathbf{R}^{-1}). \end{aligned}$$

In order to embed S^2 into $SO(3)$ we introduce equivalence classes on $SO(3)$. Two group elements $g, h \in SO(3)$ are equivalent if $g^{-1}h = \mathbf{R}_\alpha^{\mathbf{e}_z}$ for some angle $\alpha \in [0, 2\pi)$. This equivalence relation induces sections of equivalent group members, called the *left cosets* of $SO(3)$. If we associate $SO(2)$ with rotations around the z-axis, then formally we can use this equivalence to write $S^2 \equiv SO(3)/SO(2)$ to denote these left cosets.

If we extend this equivalence relation to $SE(3)$, i.e., $g, h \in SE(3)$, g is equivalent to h if $g^{-1}h = (\mathbf{0}, \mathbf{R}_\alpha^{\mathbf{e}_z})$, we obtain the left coset of $SE(3)$ which equals the space of positions and orientations. To stress that this space has been embedded in $SE(3)$ and to stress the induced (quotient) group structure we write the space of positions and orientations as

$$\mathbb{R}^3 \rtimes S^2 := (\mathbb{R}^3 \rtimes SO(3))/(\{\mathbf{0}\} \times SO(2)). \tag{2.2}$$

Now, we can express any function of position and orientation $U : \mathbb{R}^3 \rtimes S^2 \to \mathbb{R}$ with an equivalent function of $SE(3)$ $\tilde{U} : \mathbb{R}^3 \rtimes SO(3) \to \mathbb{R}$, where for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ we have

$$\tilde{U}(\mathbf{x}, \mathbf{R}) = U(\mathbf{x}, \mathbf{R}\mathbf{e}_z), \tag{2.3a}$$
$$U(\mathbf{y}, \mathbf{n}) = \tilde{U}(\mathbf{y}, \mathbf{R}_\mathbf{n}), \tag{2.3b}$$

where $\mathbf{R_n}$ is *any* rotation matrix that maps $\mathbf{e}_z$ to $\mathbf{n}$. Note that $\tilde{U}$ has the following symmetry restriction

$$\tilde{U}(\mathbf{x},\mathbf{R}) = \tilde{U}(\mathbf{x},\mathbf{R}\mathbf{R}_\alpha^{\mathbf{e}_z}) \quad \text{for all } \alpha \in [0,2\pi), \tag{2.4}$$

which means that $\tilde{U}$ should be constant within equivalence classes. We note that the ambiguity in the choice of $\mathbf{R_n}$ is harmless provided that we make sure Eq. (2.4) is satisfied.

Every group element from $SE(3)$ can be associated with a representation [6], i.e., an action of a group element on a function. Representations of $SE(3)$ can be used to rotate and translate functions and will be used later to define a rotating frame of reference to describe oriented water particles. We define the action of the group $SE(3)$ on $\mathbb{R}^3 \rtimes S^2$ by:

$$(\mathbf{x},\mathbf{R}) \cdot (\mathbf{y},\mathbf{n}) = (\mathbf{R}\mathbf{y}+\mathbf{x},\mathbf{R}\mathbf{n}). \tag{2.5}$$

We use this action to associate to every $g = (\mathbf{x},\mathbf{R}) \in SE(3)$ the corresponding rotated and translated DW-MRI dataset by

$$(\mathfrak{L}_g U)(\mathbf{y},\mathbf{n}) = U\big(g^{-1}\cdot(\mathbf{y},\mathbf{n})\big) = U(\mathbf{R}^{-1}(\mathbf{y}-\mathbf{x}),\mathbf{R}^{-1}\mathbf{n}). \tag{2.6}$$

We also need group actions of $SE(3)$ on the functions $\tilde{U}$ defined on the group itself, which we define next. For each $g_0 = (\mathbf{x}_0,\mathbf{R}_0) \in SE(3)$ and $h = (\mathbf{x},\mathbf{R_n}) \in SE(3)$, $\tilde{U} \in \mathbb{L}_2(SE(3))$, the left- and right-regular representations of $SE(3)$ on $\mathbb{L}_2(SE(3))$ are given by

$$(\mathcal{L}_{g_0} \circ \tilde{U})(h) = \tilde{U}(g_0^{-1}h) = \tilde{U}(\mathbf{R}_0^{-1}(\mathbf{x}-\mathbf{x}_0),\mathbf{R}_0^{-1}\mathbf{R_n}), \tag{2.7a}$$

$$(\mathcal{R}_{g_0} \circ \tilde{U})(h) = \tilde{U}(hg_0) = \tilde{U}(\mathbf{x}+\mathbf{R_n}\mathbf{x}_0,\mathbf{R_n}\mathbf{R}_0). \tag{2.7b}$$

Note that using the right-regular action we can rewrite the restriction of Eq. (2.4) as

$$\mathcal{R}_{(\mathbf{0},\mathbf{R}_\alpha^{\mathbf{e}_z})}\tilde{U} = \tilde{U}. \tag{2.8}$$

Since we use *left*-cosets as equivalence classes and since we have $\mathcal{L}_g\mathcal{R}_h = \mathcal{R}_h\mathcal{L}_g$, we have with our equivalence classes that

$$\widetilde{\mathfrak{L}_g U} = \mathcal{L}_g\tilde{U}, \tag{2.9}$$

where the tilde has been used to convert $\mathfrak{L}_g U$ to its equivalent function of $SE(3)$. The right action does not admit such a well-defined extension on $\mathbb{R}^3 \rtimes S^2$, so one should take care when applying right actions and make sure that when converting functions from $SE(3)$ to $\mathbb{R}^3 \rtimes S^2$ that the symmetry relation of Eq. (2.4) still holds.

Duits and Franken [10, Theorem 1] demonstrated that every reasonable linear operation on functions of $SE(3)$ must be left-invariant by showing that the orientation marginal $\int_{S^2} U(\mathbf{y},\mathbf{n})d\sigma(\mathbf{n})$ commutes with rotations and translations under such operations. This and the ill-posedness of the right-regular action on the space $\mathbb{R}^3 \rtimes S^2$ explains our choice for left-invariant processes in this paper. Formally an operator $\Phi : \mathbb{L}_2(SE(3)) \to \mathbb{L}_2(SE(3))$ is left invariant iff

$$\forall g \in SE(3),\ \ \tilde{U} \in \mathbb{L}_2(SE(3)):\ \ (\mathcal{L}_g \circ \Phi \circ \tilde{U}) = (\Phi \circ \mathcal{L}_g \circ \tilde{U}). \tag{2.10}$$

It should be noted that because of the non commutative structure of $SE(3)$ the left-regular representation $\mathcal{L}_g$ is *not* left-invariant. The right-regular representation $\mathcal{R}_g$ *is* left-invariant and can thus be used to generate left-invariant derivatives, as is shown in the next section.

2.2. Left-invariant derivatives

By viewing functions of $\mathbb{R}^3 \rtimes S^2$ as probability density functions of oriented particles, it becomes a natural idea to describe these particles in a moving coordinate system. This is done by attaching a coordinate system to each point $(\mathbf{x}, \mathbf{R}) \in SE(3)$ such that one of the spatial axes points in the direction of $\mathbf{n} = \mathbf{R_n e}_z$. In this section we will introduce diffusion equations for these oriented particles and for these processes this coordinate system is the natural choice to easily differentiate between motion forward, sideways and rotations. We can obtain such a coordinate system by starting at the identity element $(\mathbf{0}, \mathbf{I})$ of $SE(3)$, which corresponds to $(\mathbf{0}, \mathbf{e}_z)$ and attaching a suitable coordinate system using Euler angles. We express a basis of tangent vectors at the unity element by

$$A_1 = \partial_x, \;\; A_2 = \partial_y, \;\; A_3 = \partial_z, \;\; A_4 = \partial_\gamma, \;\; A_5 = \partial_\beta, \;\; A_6 = \partial_\alpha, \tag{2.11}$$

where we use the coordinate system in the parametrization of $SE(3)$: $(\mathbf{x}, R) = (x, y, z, R_{(\alpha,\beta,\gamma)})$ (see Eq. (2.1)). Note that $A_i \in T_{(\mathbf{0},\mathbf{I})}(SE(3))$ can be viewed both as tangent vectors and as differential operators on locally defined smooth functions.

We construct a moving frame of reference attached to fibers in the space $\mathbb{R}^3 \rtimes S^2$ by using the derivative of the right-regular representation $\mathcal{R}$:

$$\mathcal{A}_i|_g \tilde{U} = (d\mathcal{R}(A_i)\tilde{U})(g) = \lim_{t \downarrow 0} \frac{\tilde{U}(g e^{tA_i}) - \tilde{U}(g)}{t}, \quad i = 1,2,3,4,5,6, \tag{2.12}$$

where $\mathcal{R}$ is defined by Eq. (2.7) and e^{tA_i} is the exponential map in $SE(3)$ [10], which can be seen as the group element obtained by traveling distance t in the A_i direction from the identity element. We note that $\mathcal{A}_6 \tilde{U} = 0$, because $\tilde{U}(\mathbf{x}, \mathbf{R})$ is constant within equivalence classes and that $\mathcal{A}_1$,$\mathcal{A}_2$, $\mathcal{A}_4$ and $\mathcal{A}_5$ are well-defined on $SE(3)$ and not on $\mathbb{R}^3 \rtimes S^2$. We therefore use combinations of these operators that *are* well-defined on $\mathbb{R}^3 \rtimes S^2$ in the diffusion generator (see Section 2.3). For instance, $\mathcal{A}_4$ and $\mathcal{A}_5$ are not well defined on $\mathbb{R}^3 \rtimes S^2$ since they depend on a particular value of α, however $(\mathcal{A}_4)^2 + (\mathcal{A}_5)^2$ is well defined on $\mathbb{R}^3 \rtimes S^2$. A full classification of such well-defined second order differential operators can be found in [10].

We use Eq. (2.12) to define the left invariant derivatives on $\mathbb{R}^3 \times S^2$ by

$$\mathcal{A}_1 U(\mathbf{y}, \mathbf{n}) = \lim_{h \to 0} \frac{U(\mathbf{y} + h R_{\mathbf{n}} \mathbf{e}_x, \mathbf{n}) - U(\mathbf{y}, \mathbf{n})}{h}, \tag{2.13a}$$

$$\mathcal{A}_2 U(\mathbf{y}, \mathbf{n}) = \lim_{h \to 0} \frac{U(\mathbf{y} + h R_{\mathbf{n}} \mathbf{e}_y, \mathbf{n}) - U(\mathbf{y}, \mathbf{n})}{h}, \tag{2.13b}$$

$$\mathcal{A}_3 U(\mathbf{y}, \mathbf{n}) = \lim_{h \to 0} \frac{U(\mathbf{y} + h R_{\mathbf{n}} \mathbf{e}_z, \mathbf{n}) - U(\mathbf{y}, \mathbf{n})}{h}, \tag{2.13c}$$

$$\mathcal{A}_4 U(\mathbf{y}, \mathbf{n}) = \lim_{h_a \to 0} \frac{U(\mathbf{y}, R_{\mathbf{n}} R^{\mathbf{e}_x}_{h_a} \mathbf{e}_z) - U(\mathbf{y}, \mathbf{n})}{h_a}, \tag{2.13d}$$

$$\mathcal{A}_5 U(\mathbf{y}, \mathbf{n}) = \lim_{h_a \to 0} \frac{U(\mathbf{y}, R_{\mathbf{n}} R^{\mathbf{e}_y}_{h_a} \mathbf{e}_z) - U(\mathbf{y}, \mathbf{n})}{h_a}, \tag{2.13e}$$

where $\mathbf{R_n}$ is a *fixed* (α-dependent) rotation mapping $\mathbf{e}_z$ to $\mathbf{n}$, where we (arbitrarily) choose $\alpha = 0$. We would like to stress that here we use $\mathcal{A}_i$ both as a coordinate frame and as differential operators on smooth functions. In further sections of this paper, we will use them as differential operators exclusively.

Analytical formulas for these left-invariant derivatives, expressed in charts of Euler angles can be found in [10] where they are used to analytically approximate Green's functions of convection diffusion processes. Here, we focus on the numerical aspects and instead give only the left-invariant finite difference schemes (Section 3.2) that do not suffer from the discontinuities of the Euler angle parametrization.

2.3. Linear convection-diffusion processes on $\mathbb{R}^3 \rtimes S^2$

The left-invariant derivatives given in the previous section can be used to write the equations for diffusion processes on $SE(3)$ [10], which are processes which can remove noise from the data while preserving complex structures such as crossings and junctions [22].

The general convection-diffusion equation with diffusion matrix $\mathbf{D}$ and convection parameters $\mathbf{a}$ is given by:

$$\begin{cases} \partial_t W(\mathbf{y},\mathbf{n},t) = Q^{\mathbf{D},\mathbf{a}}(\mathcal{A}_1,\mathcal{A}_2,\cdots,\mathcal{A}_5)W(\mathbf{y},\mathbf{n},t), \\ W(\mathbf{y},\mathbf{n},0) = U(\mathbf{y},\mathbf{n}), \end{cases} \tag{2.14}$$

where the convection-diffusion generator $Q^{\mathbf{D},\mathbf{a}}$ is given by

$$Q^{\mathbf{D},\mathbf{a}}(\mathcal{A}_1,\mathcal{A}_2,\cdots,\mathcal{A}_5) = \sum_{i=1}^{5}\left(-a_i\mathcal{A}_i + \sum_{j=1}^{5}\mathcal{A}_i D_{ij}\mathcal{A}_j \right) \tag{2.15}$$

and a_i are convection parameters and D_{ij} diffusion coefficients. There are conditions on $Q^{\mathbf{D},\mathbf{a}}$ in order to be well defined on $\mathbb{R}^3 \rtimes S^2$, see [10]. We make sure these conditions have been satisfied by setting $D_{11} = D_{22} = 0$, $D_{44} = D_{55} =$ constant, $a_1 = a_2 = a_4 = a_5 = 0$ and all other $D_{ij} = 0$ for $i \neq j$. Note that there is no restriction on a_3 and D_{33}, since $\mathcal{A}_3$ is always well defined on $\mathbb{R}^3 \rtimes S^2$. Also note that although we set $D_{11} = D_{22} = 0$, there is implicit smoothing in the $\{\mathcal{A}_1,\mathcal{A}_2\}$-plane due to the commutator relations of the left-invariant vector fields, cf. [10].

In the linear case, a_i and D_{ij} are chosen constant and the solution to these evolution equations can be obtained by an $SE(3)$-convolution of the initial data with the appropriate Green's function or by using finite difference methods to directly simulate the PDEs. For a general comparison of finite difference schemes and kernel implementations, see [10,23]. We focus in this paper on the finite difference methods, as they are easier to extend to non-linear, adaptive convection-diffusion processes.

The advantage of this approach is that the convection-diffusion generator is expressed in a moving frame of reference attached to fiber fragments (see Fig. 3). This allows us to diffuse along the fibers (even across crossings) and gives a geometric interpretation to the convection-diffusion process. One could express these equations in a fixed coordinate

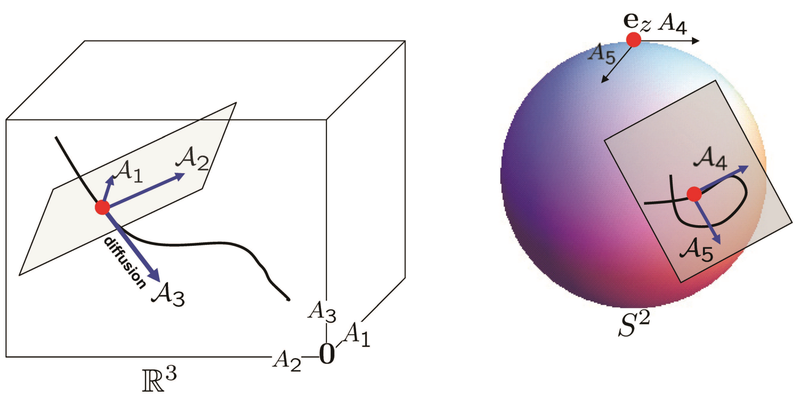


Figure 3: A figure showing the fixed coordinate system (A_i) and the one rotated along a fiber fragment $(\mathcal{A}_i)$ for a certain choice of parameter α. The left image shows the spatial part and the right image the angular part of the same curve.

system (using two charts for S^2) by substituting the formula's for the left-invariant vector fields [10, Eqs. (24) and (25)]. However, this leads to cumbersome bookkeeping and lengthy formulas unsuitable for finite difference implementations.

We will discuss two special cases of linear convection-diffusion equations. The first is contour enhancement, which is a diffusion process based on Brownian motion of oriented particles [10], which is useful for denoising and inferring crossings. The second is 3D contour completion [10], which is an extension of 2D contour completion [6, 11, 18]. Contour completion also incorporates a convection term and can be used to fill in missing data. First, however, we will focus on discretizing the space of $\mathbb{R}^3 \rtimes S^2$ and the finite difference operators of Eq. (2.14), so that we can simulate them numerically.

3. Discretization of $\mathbb{R}^3 \rtimes S^2$ and finite difference schemes

3.1. Discretization of $\mathbb{R}^3 \rtimes S^2$

So far, functions $U : \mathbb{R}^3 \rtimes S^2 \to \mathbb{R}^+$ were assumed to be continuously differentiable. In practice we have discretized functions $U[i,j,k,l]$, where $i \in \{1,\cdots,N_1\}$, $j \in \{1,\cdots,N_2\}$ and $k \in \{1,\cdots,N_3\}$ enumerate the discrete spatial grid and $l \in \{1,\cdots,N_o\}$ refers to an orientation $\mathbf{n}_l$ from a sampling of the sphere. These sampling points then correspond to the spatial point $\mathbf{y}_{ijk} = (y_i^1, y_j^2, y_k^3) \in \mathbb{R}^3$ with $y_i^1 = (i-1)\Delta x$, $y_j^2 = (j-1)\Delta y$, $y_k^3 = (k-1)\Delta z$, where $\Delta x, \Delta y$ and Δz are the sampling interval in x, y and z direction respectively. The spherical tessellation used in this paper is obtained by taking an icosahedron and regularly subdividing each face into 16 triangles before projecting the vertices back to the sphere. Every vertex of this shape becomes a sampling orientation and thus in our case $N_o = 162$.

For some proofs and to be able to write linear operations as matrix-column multiplications, it is often convenient to represent $U[i,j,k,l]$ as a column vector $\mathbf{u} = (u^p)_{p\in\{1,\cdots,N_1N_2N_3N_o\}}$. In this paper, we use two different ways to convert 4 indices to one, related by a simple transformation matrix. To do this, we first combine the spatial coordinates i, j, k to a linear index $q \in \{1, \cdots, N_1N_2N_3\}$ by first iterating over the x direction, then the y-direction and finally the z direction. This is described by function $\Psi[i, j, k]$ and its inverse $\Psi^{-1}[q]$ defined by:

$$q = \Psi[i, j, k] = (i-1) + N_1(j-1) + N_1N_2(k-1) + 1, \tag{3.1a}$$

$$i = (\Psi^{-1}[q])_1 = (q-1) \bmod (N_1) + 1, \tag{3.1b}$$

$$j = (\Psi^{-1}[q])_2 = \lfloor \frac{q-1}{N_1} \rfloor \bmod (N_2) + 1, \tag{3.1c}$$

$$k = (\Psi^{-1}[q])_3 = \lfloor \frac{q-1}{N_1N_2} \rfloor + 1. \tag{3.1d}$$

We now combine the two indices l, q into a single index $p \in \{1, \cdots, N_1N_2N_3N_o\}$ by first iterating over the orientations l and then the linear index q, so we have

$$u^p = U[\mathbf{y}_{q=\lfloor \frac{p-1}{N_o} \rfloor + 1}, \mathbf{n}_{l=(p-1) \bmod (N_o)+1}], \tag{3.2}$$

where we denote $\mathbf{y}_q := \mathbf{y}_{\Psi^{-1}[q]}$. If we instead iterate over the linear index q first and the orientations l last we obtain vector $\hat{\mathbf{u}}$ indexed by $\hat{p} \in \{1, \cdots, N_1N_2N_3N_o\}$ defined by

$$\hat{u}^{\hat{p}} = U\left[\mathbf{y}_{q=(\hat{p}-1) \bmod (N_1N_2N_3)+1}, \mathbf{n}_{l=\lfloor \frac{\hat{p}-1}{N_1N_2N_3} \rfloor + 1}\right]. \tag{3.3}$$

Using the relations

$$p = l + (q-1)N_o, \quad \hat{p} = (l-1)N_1N_2N_3 + q, \tag{3.4a}$$

$$l = (p-1) \bmod (N_o) + 1 = \lfloor \frac{\hat{p}-1}{N_1N_2N_3} \rfloor + 1, \tag{3.4b}$$

$$q = \lfloor \frac{p-1}{N_o} \rfloor + 1 = (\hat{p}-1) \bmod (N_1N_2N_3) + 1, \tag{3.4c}$$

we can relate p and $\hat{p}$ by

$$\hat{p} = (p-1) \bmod (N_o) \cdot N_1N_2N_3 + \lfloor \frac{p-1}{N_o} \rfloor + 1 \tag{3.5}$$

and therefore we can write $\hat{\mathbf{u}} = \mathbf{P}\mathbf{u}$, where

$$P_{\hat{p}p} = \delta_{\hat{p}, (p-1) \bmod (N_o) N_1N_2N_3 + \lfloor \frac{p-1}{N_o} \rfloor + 1} \quad \text{with} \quad \mathbf{P} = [P_{\hat{p}p}] \in \mathbb{R}^{N_1N_2N_3N_o \times N_1N_2N_3N_o}, \tag{3.6}$$

and $\mathbf{P}^{-1} = \mathbf{P}^T$ to convert between the different indexing methods.

A visual summary of the different vectors and transformations between them is given in Fig. 4.

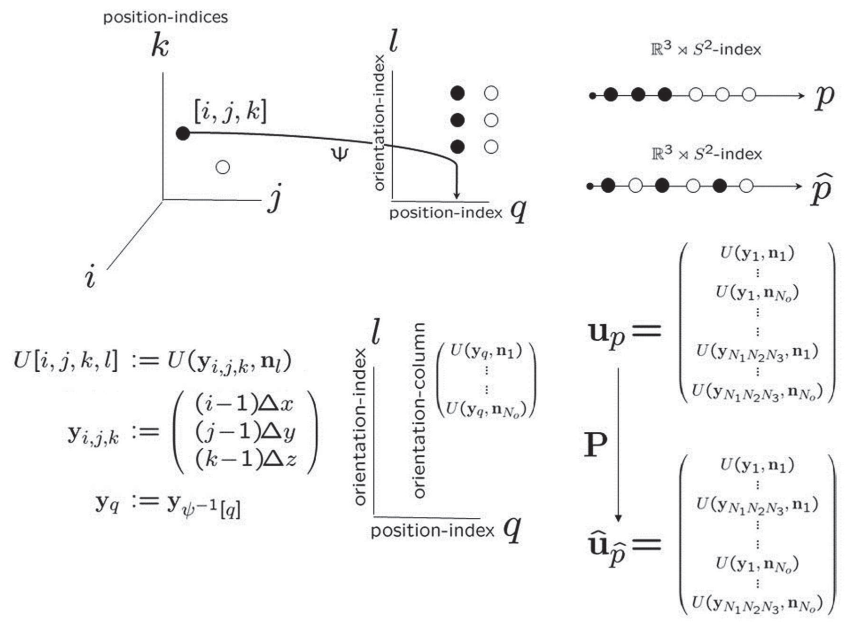

position-indices
k
$[i, j, k]$
j
i
Ψ
orientation-index
l
position-index q
$\mathbb{R}^3 \rtimes S^2$-index
p
$\mathbb{R}^3 \rtimes S^2$-index
$\hat{p}$
$U[i,j,k,l] := U(\mathbf{y}_{i,j,k}, \mathbf{n}_l)$
$\mathbf{y}_{i,j,k} := \begin{pmatrix} (i-1)\Delta x \\ (j-1)\Delta y \\ (k-1)\Delta z \end{pmatrix}$
$\mathbf{y}_q := \mathbf{y}_{\psi^{-1}[q]}$
orientation-index
orientation-column
$U(\mathbf{y}_q, \mathbf{n}_1)$
$U(\mathbf{y}_q, \mathbf{n}_{N_o})$
$\mathbf{u}_p =$
$U(\mathbf{y}_1, \mathbf{n}_1)$
$U(\mathbf{y}_1, \mathbf{n}_{N_o})$
$U(\mathbf{y}_{N_1N_2N_3}, \mathbf{n}_1)$
$U(\mathbf{y}_{N_1N_2N_3}, \mathbf{n}_{N_o})$
$\mathbf{P}$
$\hat{\mathbf{u}}_{\hat{p}} =$

Figure 4: Graphical overview of the discretization process. The top row visualizes the different indices used to discretize the domain of function U on position and orientation. The bottom row focusses on the range of such functions.

3.2. Finite difference schemes for left-invariant derivatives

To approximate the left-invariant derivatives of the previous section, we use finite difference approximations [10] of Eq. (2.12). These derivatives are approximated in the usual way, with the (conceptually) small difference that the steps are taken in the $\mathcal{A}_i$ direction on $\mathbb{R}^3 \times S^2$. The forward finite difference approximation of the left-invariant derivatives are given by

$$\mathcal{A}_1^f U(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y}+hR_{\mathbf{n}}\mathbf{e}_x,\mathbf{n}) - U(\mathbf{y},\mathbf{n})}{h}, \qquad \mathcal{A}_2^f U(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y}+hR_{\mathbf{n}}\mathbf{e}_y,\mathbf{n}) - U(\mathbf{y},\mathbf{n})}{h}, \tag{3.7a}$$

$$\mathcal{A}_3^f U(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y}+hR_{\mathbf{n}}\mathbf{e}_z,\mathbf{n}) - U(\mathbf{y},\mathbf{n})}{h}, \qquad \mathcal{A}_4^f U(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y},R_{\mathbf{n}}R_{h_a}^{\mathbf{e}_x}\mathbf{e}_z) - U(\mathbf{y},\mathbf{n})}{h_a}, \tag{3.7b}$$

$$\mathcal{A}_5^f U(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y},R_{\mathbf{n}}R_{h_a}^{\mathbf{e}_y}\mathbf{e}_z) - U(\mathbf{y},\mathbf{n})}{h_a}, \tag{3.7c}$$

where h is the spatial step size and h_a the angular step size in radians.

Analogously, the backward and central finite difference approximations can be obtained

$$\mathcal{A}_1^b U(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y},\mathbf{n}) - U(\mathbf{y}-hR_{\mathbf{n}}\mathbf{e}_x,\mathbf{n})}{h}, \qquad \mathcal{A}_2^b U(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y},\mathbf{n}) - U(\mathbf{y}-hR_{\mathbf{n}}\mathbf{e}_y,\mathbf{n})}{h}, \tag{3.8a}$$

$$\mathcal{A}_3^b U(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y},\mathbf{n}) - U(\mathbf{y} - hR_{\mathbf{n}}\mathbf{e}_z,\mathbf{n})}{h}, \qquad \mathcal{A}_4^b U(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y},\mathbf{n}) - U(\mathbf{y}, R_{\mathbf{n}} R^{\mathbf{e}_x}_{-h_a}\mathbf{e}_z)}{h_a}, \tag{3.8b}$$

$$\mathcal{A}_5^b U(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y},\mathbf{n}) - U(\mathbf{y}, R_{\mathbf{n}} R^{\mathbf{e}_y}_{-h_a}\mathbf{e}_z)}{h_a}, \tag{3.8c}$$

and

$$\mathcal{A}_1^c U(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y} + hR_{\mathbf{n}}\mathbf{e}_x,\mathbf{n}) - U(\mathbf{y} - hR_{\mathbf{n}}\mathbf{e}_x,\mathbf{n})}{2h}, \tag{3.9a}$$

$$\mathcal{A}_2^c U(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y} + hR_{\mathbf{n}}\mathbf{e}_y,\mathbf{n}) - U(\mathbf{y} - hR_{\mathbf{n}}\mathbf{e}_y,\mathbf{n})}{2h}, \tag{3.9b}$$

$$\mathcal{A}_3^c U(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y} + hR_{\mathbf{n}}\mathbf{e}_z,\mathbf{n}) - U(\mathbf{y} - hR_{\mathbf{n}}\mathbf{e}_z,\mathbf{n})}{2h}, \tag{3.9c}$$

$$\mathcal{A}_4^c U(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y}, R_{\mathbf{n}} R^{\mathbf{e}_x}_{h_a}\mathbf{e}_z) - U(\mathbf{y}, R_{\mathbf{n}} R^{\mathbf{e}_x}_{-h_a}\mathbf{e}_z)}{2h_a}, \tag{3.9d}$$

$$\mathcal{A}_5^c U(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y}, R_{\mathbf{n}} R^{\mathbf{e}_y}_{h_a}\mathbf{e}_z) - U(\mathbf{y}, R_{\mathbf{n}} R^{\mathbf{e}_y}_{-h_a}\mathbf{e}_z)}{2h_a}. \tag{3.9e}$$

We take second order centered finite differences by applying the discrete operators in the righthand side of Eq. (3.9) twice (where we replaced $2h \mapsto h$), e.g., we have for $\rho = 1,2,3$:

$$((\mathcal{A}_\rho^c)^2 U)(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y} + hR_{\mathbf{n}}\mathbf{e}_\rho,\mathbf{n}) - 2U(\mathbf{y},\mathbf{n}) + U(\mathbf{y} - hR_{\mathbf{n}}\mathbf{e}_\rho,\mathbf{n})}{h^2}, \tag{3.10a}$$

$$((\mathcal{A}_{\rho+3}^c)^2 U)(\mathbf{y},\mathbf{n}) = \frac{U(\mathbf{y}, R_{\mathbf{n}} R^{\mathbf{e}_\rho}_{h_a}\mathbf{e}_z) - 2U(\mathbf{y},\mathbf{n}) + U(\mathbf{y}, R_{\mathbf{n}} R^{\mathbf{e}_\rho}_{-h_a}\mathbf{e}_z)}{{h_a}^2}. \tag{3.10b}$$

3.3. Efficient computation of left-invariant finite differences

The finite difference approximations of the previous section requires sampling off-grid, which requires interpolation. Spatially, any regular 3D interpolation scheme such as linear interpolation or spline interpolation can be used. Since the approximations in Eq. (3.7) are only first order accurate, we use linear interpolation.

The three spatial derivatives only require neighboring samples with the same $\mathbf{n}$. They can therefore be efficiently computed through a regular $\mathbb{R}^3$ correlation (or convolution) for each orientation separately. We approximate the spatial derivatives by

$$(\mathcal{A}_\rho^f U)[i,j,k,l] \approx -\frac{1}{h} U[i,j,k,l] + \frac{1}{h} \sum_{i',j',k'} K_l^{\rho,h}[i-i',j-j',k-k'] U[i',j',k',l],$$

with $\rho \in \{1,2,3\}$ and l-indexed discrete spatial kernel $K_l^{\rho,h}$ given by

$$K_l^{\rho,h}[i^1,i^2,i^3] = \prod_{m=1}^{3} v_{(\mathbf{y}_{\rho,l})^m}[i^m], \tag{3.11}$$

where $(\mathbf{y}_{\rho,l})^m$ is the m-th component of the vector $\mathbf{y}_{\rho,l} := hR_{\mathbf{n}_l}\mathbf{e}_\rho$, $\rho \in \{1,2,3\}$. Linear interpolation kernel $v_a : \mathbb{Z} \to [0,1]$ is given by

$$v_a[b] = \begin{cases} 1-|a|, & \text{if } b=0, \\ H(ab)|a|, & \text{if } b \in \{-1,1\}, \\ 0, & \text{else,} \end{cases} \tag{3.12}$$

with heaviside function $u \mapsto H(u)$ while assuming $|a| < 1$.

For angular interpolation, either linear interpolation or spherical harmonics can be used. Spherical harmonics were used in previous work by Franken et al. [13], but we now use linear interpolation. In terms of stability in the diffusion process, both perform equally well (see Section 4), but Franken had to add an angular diffusion term t_{reg} which in practice was rather sensitive: when set too large the data becomes too isotropic (destroying fiber structures) and when set too small the algorithm becomes unstable. As we will show next, linear interpolation is also computationally cheaper.

The angular derivatives only require samples of neighbors with the same $\mathbf{y}$ and can therefore be computed by a matrix multiplication for each point $\mathbf{y}$:

$$\mathcal{A}^f_{\rho+3}U[i,j,k,l] \approx \frac{1}{h_a}\left(-U[i,j,k,l] + \sum_{l'=1}^{N_o} M^{f,h_a,\rho+3}_{ll'} U[i,j,k,l'] \right), \tag{3.13}$$

with $\rho \in \{1,2,3\}$ and where $\mathbf{M}^{f,h_a,\rho+3} = [M^{f,h_a,\rho+3}_{ll'}]$ is the interpolation matrix to interpolate $\mathbf{n}_{\rho,l} = R_{\mathbf{n}_l} R_{\mathbf{e}_\rho,h_a}\mathbf{e}_z$ and is given by

$$M^{f,h_a,\rho+3}_{ll'} = \begin{cases} 1 - \sum\limits_{\mathbf{n}_j \in A_{\rho,l}} (\mathbf{n}_{\rho,l} - \mathbf{n}_{l'}) \cdot (\mathbf{n}_j - \mathbf{n}_{l'}), & \text{if } \mathbf{n}_{l'} \in A_{\rho,l}, \\ 0, & \text{otherwise,} \end{cases} \tag{3.14}$$

where $A_{\rho,l}$ is the triangle within the spherical sampling that contains point $\mathbf{n}_{\rho,l}$. The matrix $\mathbf{M}^{f,h_a,\rho+3}$ is sparse due to the linear interpolation which enables Eq. (3.13) to be computed in $\mathcal{O}(N_1N_2N_3N_o)$. If $M^{f,h_a,\rho+3}$ is created using spherical harmonics then it becomes a full matrix and thus Eq. (3.13) becomes computationally more expensive and takes $\mathcal{O}(N_1N_2N_3N_o^2)$.

3.4. Matrix representation of left-invariant derivatives

We will derive the $N_1N_2N_3N_o \times N_1N_2N_3N_o$ matrix form of the *forward* approximation Eq. (3.7) of the left-invariant vector fields. Here we assume the data has been discretized and put in vector form as is described in Section 3.1. Note that the matrix-form of the backward and central differences can be derived analogously.

If we store $U[i,j,k,l]$ in one long column vector $\mathbf{u}$ as described in Section 3.1, we can represent the (forward) left-invariant vector fields by the matrix:

$$\mathcal{A}^f_\rho U[i,j,k,l] = \mathbf{A}^f_\rho \mathbf{u}, \tag{3.15}$$

with $\rho \in \{1,2,3,4,5\}$.

The (forward) *angular* left-invariant vector fields are then given by

$$\begin{aligned}\mathbf{A}^f_{\rho+3} :=& \frac{1}{h_a}\left(\bigoplus_{q=1}^{N_1N_2N_3}(\mathbf{M}^{f,h_a,\rho+3}-\mathbf{I}_{N_o})\right)\\ =& \frac{1}{h_a}\left(\mathbf{I}_{N_1N_2N_3}\otimes\mathbf{M}^{f,h_a,\rho+3}-\mathbf{I}_{N_1N_2N_3N_o}\right)\end{aligned} \tag{3.16}$$

with $\rho \in \{1,2,3\}$, $\oplus$ the direct sum of matrices, $\otimes$ the Kronecker product and with $\mathbf{M}^{f,h_a,\rho+3} \in \mathbb{R}^{N_o\times N_o}$ the matrix with entries given by Eq. (3.14) and where $\mathbf{I}_N \in \mathbb{R}^{N\times N}$ denotes the identity matrix.

The (forward) *spatial* left-invariant vector fields can be represented by the matrix

$$\mathbf{A}^f_\rho := \frac{1}{h}(\mathbf{M}^{f,\rho,h}-\mathbf{I}_{N_1N_2N_3N_o}) \tag{3.17}$$

with $\rho \in \{1,2,3\}$ and $\mathbf{M}^{f,\rho,h} = [M^{f,\rho,h}_{p'p}] \in \mathbb{R}^{N_1N_2N_3N_o\times N_1N_2N_3N_o}$ defined by

$$M^{f,\rho,h}_{p'p} = \begin{cases} K^{\rho,h}_{l_p}[\Psi^{-1}[q_p]-\Psi^{-1}[q'_{p'}]], & \text{if } l_p = l'_{p'},\\ 0, & \text{otherwise,}\end{cases} \tag{3.18}$$

where q_p and l_p are related to p by $q_p = \lfloor\frac{p-1}{N_o}\rfloor+1$ and $l_p = (p-1)\,\mathrm{mod}\,(N_o)+1$ and identical relations for $q'_{p'}$ and $l'_{p'}$ (see Eq. (3.4)) and Ψ and its inverse are given in Eq. (3.1). If however, we use $\hat{\mathbf{u}}$ for discretization instead (see Eq. (3.4)) then $\mathbf{M}^{f,\rho,h,l}$ becomes a block matrix[‡]. Using transformation matrix $\mathbf{P}^{-1}$ we can relate this block matrix to the original index p, by means of

$$\mathbf{M}^{f,\rho,h} = \mathbf{P}^{-1}\bigoplus_{l=1}^{N_o}K^{\rho,h}_l[\Psi^{-1}[q]-\Psi^{-1}[q']], \tag{3.19}$$

with $q,q' \in \{1,\cdots,N_1N_2N_3\}$. This matrix cannot be further simplified using the Kronecker product, since matrix $K^{\rho,h}_l$ is dependent on orientation l. This dependency on orientation is the direct consequence of the non-commutative structure of the Euclidean motion group. In a similar way, one defines matrices for central and backward differences, denoted by $\mathbf{A}^c_\rho$ and $\mathbf{A}^b_\rho$.

4. Numerical schemes for contour enhancement

The *contour enhancement* process on $\mathbb{R}^3 \rtimes S^2$ can be obtained from Eq. (2.14) by setting $a_i = 0$ (no convection), $D_{33} \geq 0$, $D_{44} = D_{55} \geq 0$ and other diffusion coefficients D_{ij} are set

[‡]Note that only spatial neighbors with the same $\mathbf{n}_l$ are required for the spatial derivatives. Points with the same $\mathbf{n}_l$ are stored adjacent to each other in $\hat{u}$, explaining the block structure of matrix $\mathbf{M}^{f,\rho,h}$.

to zero. These settings yield the following evolution equation

$$\begin{cases} \partial_t W(\mathbf{y},\mathbf{n},t) = \big(D_{33}(\mathcal{A}_3)^2 + D_{44}((\mathcal{A}_4)^2 + (\mathcal{A}_5)^2)\big) W(\mathbf{y},\mathbf{n},t), \\ W(\mathbf{y},\mathbf{n},0) = U(\mathbf{y},\mathbf{n}). \end{cases} \tag{4.1}$$

This process can intuitively be understood as a description of the Brownian motion of oriented particles both in space (diffusion along direction **n**) and orientation (changing direction) [10]. We first elaborate on explicit methods and derive stability bounds for them and then implement an implicit integration schemes and compare the different methods.

4.1. Explicit scheme for linear contour enhancement

The simulation of the contour enhancement PDE given by Eq. (4.1) is done by taking standard centered second order finite differences according to Eq. (3.10) and using a forward Euler scheme for the time discretization:

$$\begin{cases} W(\mathbf{y},\mathbf{n},t+\Delta t) = W(\mathbf{y},\mathbf{n},t) + \Delta t\big(D_{33}(\mathcal{A}_3^c)^2 + D_{44}((\mathcal{A}_4^c)^2 + (\mathcal{A}_5^c)^2)\big) W(\mathbf{y},\mathbf{n},t), \\ W(\mathbf{y},\mathbf{n},0) = U(\mathbf{y},\mathbf{n}). \end{cases}$$

We will now derive the equivalent equation on discretized functions. First we represent the generator of the (hypo)-elliptic diffusion, Eq. (4.1) in matrix form by using Section 3.4, so we obtain

$$\mathbf{J}^{\mathbb{R}^3} := D_{33}\mathbf{A}_3^f\mathbf{A}_3^b + D_{11}\big(\mathbf{A}_1^f\mathbf{A}_1^b + \mathbf{A}_2^f\mathbf{A}_2^b\big), \tag{4.2a}$$

$$\mathbf{J}^{S^2} := D_{44}\big(\mathbf{A}_4^f\mathbf{A}_4^b + \mathbf{A}_5^f\mathbf{A}_5^b\big), \tag{4.2b}$$

with $\mathbf{J}^{\mathbb{R}^3} + \mathbf{J}^{S^2} \in \mathbb{R}^{N_1N_2N_3N_o \times N_1N_2N_3N_o}$.

Then, using the discretization from Section 3.1, we define the solution of the contour enhancement process at $t = (s-1)\Delta t$, $s \in \{1,2,\cdots\}$ by $\mathbf{w}^s$ with $\mathbf{w}^1 = \mathbf{u}$, the initial state. With full discretization of $\mathbb{R}^3 \rtimes S^2$ and of the operators, the equations then become

$$\begin{cases} \mathbf{w}^{s+1} = (I + \Delta t(\mathbf{J}^{\mathbb{R}^3} + \mathbf{J}^{S^2}))\mathbf{w}^s, \\ \mathbf{w}^1 = \mathbf{u}. \end{cases} \tag{4.3}$$

Of these parameters, D_{44} and simulation time t are most important. D_{33} may be set to 1, as changing D_{33} is equivalent to scaling D_{44} and t, while Δt needs only be sufficiently small for the algorithm to remain stable (see Section 4.2) and accurate.

4.2. Stability criterion for contour enhancement using explicit euler

The stability limit of the finite difference schemes for the linear contour enhancement of Eq. (4.1) using linear interpolation can be found in a similar way as was done in previous work [10]. The difference is that here we use linear interpolation for the angular derivatives and therefore arrive at a different stability bound.

Using the vector form, discretization and notation from Section 3.1, we define the solution of the contour enhancement process at $t = (s-1)\Delta t$, $s \in \{1, 2, \cdots\}$ by $\mathbf{w}^s$, with $\mathbf{w}^1 = \mathbf{u}$, the initial state. Then

$$\mathbf{w}^{s+1} = (I + \Delta t(\mathbf{J}^{\mathbb{R}^3} + \mathbf{J}^{S^2}))\mathbf{w}^s, \tag{4.4}$$

where $\mathbf{J}^{\mathbb{R}^3}$ is the spatial increment matrix and $\mathbf{J}^{S^2}$ is the angular increment matrix defined in Eq. (4.2) that specify how much $\mathbf{w}^{s+1}$ grows relative to $\mathbf{w}^s$ due to spatial or angular effects. Such a system is stable if

$$||\mathbf{I} + \Delta t(\mathbf{J}^{\mathbb{R}^3} + \mathbf{J}^{S^2})|| = \sup_{||\mathbf{w}||=1} ||(I + \Delta t(\mathbf{J}^{\mathbb{R}^3} + \mathbf{J}^{S^2}))\mathbf{w}|| < 1. \tag{4.5}$$

Since $\mathbf{I} + \Delta t(\mathbf{J}^{\mathbb{R}^3} + \mathbf{J}^{S^2}) = (\nu\mathbf{I} + \Delta t\mathbf{J}^{\mathbb{R}^3}) + ((1-\nu)\mathbf{I} + \Delta t\mathbf{J}^{S^2})$ with $\nu \in (0,1)$, such a system will be stable if

$$\left\|\mathbf{I} + \frac{\Delta t}{\nu}\mathbf{J}^{\mathbb{R}^3}\right\| < 1 \quad \text{and} \quad \left\|\mathbf{I} + \frac{\Delta t}{1-\nu}\mathbf{J}^{S^2}\right\| < 1. \tag{4.6}$$

To obtain stability bounds for these two equations, we use Gerschgorin Circle Theorem [14], which states that for any Hermitian matrix $\mathbf{M}$, the eigenvalues are contained in the interval with center M_{ii} and radius $r_i = \sum_{j\neq i} |M_{ij}|$, the sum of the absolute values of the off diagonal values. This means that $\mathbf{M}$ is stable if

$$M_{ii} - r_i \geq -1 \quad \text{and} \quad M_{ii} + r_i \leq 1, \quad \forall i. \tag{4.7}$$

The first inequality of Eq. (4.6) can be solved by realizing that $\mathbf{J}^{\mathbb{R}^3}$ is the matrix representation of the spatial second order differential operators

$$D_{11}\left((\mathcal{A}_1)^2 + (\mathcal{A}_2)^2\right) + D_{33}(\mathcal{A}_3)^2,$$

where each of the $(\mathcal{A}_\rho)^2$ has been discretized by a $\{1, -2, 1\}$-stencil for differentiation (see Eq. (3.9)). This means that

$$J_{ii}^{\mathbb{R}^3} = \frac{-4D_{11} - 2D_{33}}{h^2} \quad \text{and} \quad \sum_{j\neq i} |J_{ij}^{\mathbb{R}^3}| = \frac{4D_{11} + 2D_{33}}{h^2}$$

for all $i \in \{1, \cdots, N_1 N_2 N_3 N_o\}$ (assuming cyclic boundary conditions). This means that the rightmost inequality of Eq. (4.7) is always satisfied. The left inequality is satisfied if

$$1 - 2\frac{\Delta t}{\nu}\frac{4D_{11} + 2D_{33}}{h^2} > -1,$$

which leads to the following bound

$$\Delta t \leq \frac{\nu h^2}{4D_{11} + 2D_{33}}. \tag{4.8}$$

To solve the second inequality of Eq. (4.6), we use the same ideas and realize that $\mathbf{J}^{S^2}$ is the matrix representation of $D_{44}((\mathcal{A}_4)^2 + (\mathcal{A}_5)^2)$, which is again obtained with a $\{1, -2, 1\}$-stencil. This leads to the stability criterion for the angular increments matrix

$$\Delta t \leq \frac{(1-\nu)h_a^2}{4D_{44}}. \tag{4.9}$$

Eqs. (4.8) and (4.9) now give us two bounds for the time step. Naturally, the smallest one of the two is the stability bound for the whole process. Since both limits are a function of ν, we can optimize to find the maximum value for Δt

$$\Delta t \leq \max_{\nu} \min \left(\frac{\nu h^2}{4D_{11} + 2D_{33}}, \frac{(1-\nu)h_a^2}{4D_{44}} \right) = \left(\frac{(4D_{11} + 2D_{33})}{h^2} + \frac{4D_{44}}{h_a^2} \right)^{-1}. \tag{4.10}$$

This result is very similar to the result obtained in [10, Eq.(89)], where the authors derived the following stability bound:

$$\Delta t \leq \left(\frac{4D_{11} + 2D_{33}}{h^2} + D_{44} \frac{L(L+1)}{2e^{t_{\mathrm{reg}} L(L+1)}} \right)^{-1}, \qquad \text{if } t_{\mathrm{reg}} L(L+1) \leq 1, \tag{4.11a}$$

$$\Delta t \leq \left(\frac{4D_{11} + 2D_{33}}{h^2} + D_{44} \frac{1}{2e t_{\mathrm{reg}}} \right)^{-1}, \qquad \text{if } t_{\mathrm{reg}} L(L+1) > 1, \tag{4.11b}$$

where they computed the angular derivatives in the spherical harmonics domain with spherical harmonics up to order L and had to add a regularization parameter t_{reg} that was necessary to obtain stable algorithms, but which essentially induces additional spherical diffusion.

Both stability criteria have parameters that may be tuned to optimize the stability. Both use h, the spatial step size of the spatial derivatives. Eq. (4.11) uses t_{reg}, a regularization parameter and L, the maximum number of spherical harmonics, while Eq. (4.10) uses h_a, the step size of the angular derivatives. Because both have regularizing parameters, both methods can be made equally stable and thus there is no real difference between them in terms of stability. We give preference to linear interpolation over the spherical harmonics implementation, as it only requires choosing one parameter (h_a) instead of two (L and t_{reg}). Linear interpolation is furthermore computationally cheaper, see Section 3.3.

4.3. Implicit scheme for linear contour enhancement

The implicit scheme for contour enhancement is obtained in the usual way from Eq. (4.1),

$$\begin{cases} W(\mathbf{y}, \mathbf{n}, t + \Delta t) = W(\mathbf{y}, \mathbf{n}, t) + \Delta t \big(D_{33} (\mathcal{A}_3^c)^2 + D_{44}((\mathcal{A}_4^c)^2 + (\mathcal{A}_5^c)^2) \big) W(\mathbf{y}, \mathbf{n}, t + \Delta t), \\ W(\mathbf{y}, \mathbf{n}, 0) = U(\mathbf{y}, \mathbf{n}), \end{cases}$$

which after discretization yields

$$\begin{cases} \mathbf{w}^{s+1} = \mathbf{w}^s + \Delta t (\mathbf{J}^{\mathbb{R}^3} + \mathbf{J}^{S^2}) \mathbf{w}^{s+1} = \mathbf{w}^s + \mathbf{Q} \mathbf{w}^{s+1}, \\ \mathbf{w}^1 = \mathbf{u}, \end{cases} \tag{4.12}$$

where $\mathbf{Q} = \Delta t(\mathbf{J}^{\mathbb{R}^3} + \mathbf{J}^{S^2})$ defined in Eq. (4.2). To solve this equation, the linear system $(\mathbf{I} - \mathbf{Q})\mathbf{w}^{s+1} = \mathbf{w}^s$ must be solved for each time step. To do this, we can employ any of the plethora of methods known for solving linear systems. In our case, we use the conjugate gradient method (see for example [17, Section 11.5.5]) to iteratively solve this system. Because it only requires evaluations of **Qv** and not matrix **Q** explicitly, this method allows the use of the Finite differences techniques of Section 3.3 to evaluate **Qv** without explicitly having to form matrix **Q**.

This algorithm is unconditionally stable, regardless of the value of Δt and is thus useful for solving Eq. (4.12) with large time steps.

The number of iterations $k_{\max}$ needs to be set for the conjugate gradient method (or a stopping criterion should be specified). To get a sense of how many iterations are necessary, we look at the relation between error and computation time for implicit methods with different amount of iterations and for explicit methods. To do this, we generate a synthetic $5 \times 5 \times 5$ dataset, consisting of a single, isotropic ball in the center and perform contour enhancement on it with $t = 1$, $D_{33} = 1$, $D_{44} = 0.1$. Different combinations of relative error and computation time are generated by using varying values of Δt, which varies from 0.001 to 1. For a measure of the error, we use the relative ∞-norm, defined by

$$||\mathbf{w}||_{\infty}^{rel} = \frac{||\mathbf{w} - \mathbf{w}_{gt}||_{\infty}}{||\mathbf{w}_{gt}||_{\infty}}, \tag{4.13}$$

where $\mathbf{w}$ is the result of numerical simulations and $\mathbf{w}_{gt}$ is the ground truth, which is generated by using an explicit scheme with $\Delta t = 10^{-5}$. The results can be found in Fig. 5.

Fig. 5 shows that the conjugate gradient algorithm converges very fast and typically 2 iterations are sufficient, since more iterations only take up time without improving the accuracy. The left side of the graph corresponds to very large Δt, higher than the stability

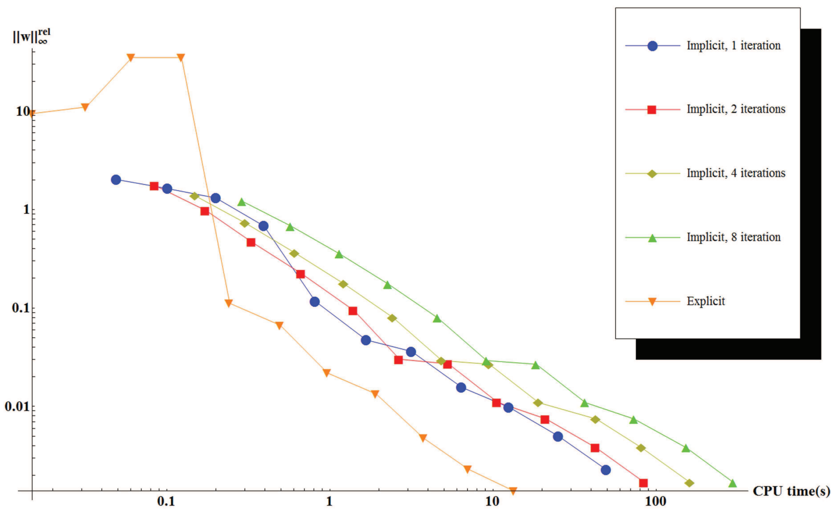


Figure 5: A graph showing the total computation time vs the relative error in ∞-norm.

bound given in Section 4.2, which explains the very high relative errors for the explicit method. We can conclude that the explicit method is better than the implicit method, provided that the explicit method is stable.

5. Numerical schemes for contour completion

By setting $a_3 = 1$, $D_{44} = D_{55} \geq 0$ in the convection-diffusion generator of Eq. (2.15) and the other constants equal to zero, we obtain a convection-diffusion process given by:

$$\begin{cases} \partial_t W(\mathbf{y},\mathbf{n},t) = \big(-\mathcal{A}_3 + D_{44}(\mathcal{A}_4)^2 + (\mathcal{A}_5)^2\big)W(\mathbf{y},\mathbf{n},t), \\ W(\mathbf{y},\mathbf{n},0) = U(\mathbf{y},\mathbf{n}), \end{cases} \tag{5.1}$$

which can be interpreted in much the same way as the contour enhancement equation, only with the diffusion part in the spatial domain replaced with a convection term. This means that the oriented particles are now propelled forwards, rather than being allowed to diffuse naturally.

For the numerics, we will be more brief and simply state that we use backwards finite difference approximation for $\mathcal{A}_3$ (see Eq. (3.8)) according to the upwind principle, centered finite differences for $(\mathcal{A}_4)^2 + (\mathcal{A}_5)^2$ (see 3.10) and explicit Euler time integration (see Section 4.1).

Special care should be given to choosing the time step Δt for simulating this equation, as convection terms and diffusion terms typically put different constraints on the size of Δt. For this reason, we apply the convection and diffusion steps separately and use different Δt for both processes. For details, see Section 5.1.

We obtain the resolvent of this convection-diffusion process by integrating this process over time and weighing each time with a probability density function specifying the likelihood of having that traveling time. This process is called the *contour completion* process, which is given by:

$$R(\mathbf{y},\mathbf{n}) = \int_0^\infty W(\mathbf{y},\mathbf{n},t) \cdot p(t)dt, \tag{5.2}$$

where $p(t)$ is a chosen probability function of the traveling time and $W(\mathbf{y},\mathbf{n},t)$ is described by the convection-diffusion process of Eq. (5.1). Here, we call the solution to Eq. (5.1) the convection-diffusion process and Eq. (5.2) the contour completion process. A natural choice for $p(t)$ is the exponential distribution (which was Duits and Franken's choice [10]) because it is the *only* probability distribution with the memoryless property and is given by:

$$p(t) = \begin{cases} \lambda e^{-\lambda t}, & t \geq 0, \\ 0, & t < 0, \end{cases} \tag{5.3}$$

with rate parameter λ which specifies how quickly the particles stop moving.

Contour completion is a process that can be used if parts of the data are missing by reconstructing these missing parts from contextual information. Fig. 6 shows the effect of contour completion on a synthetic test dataset with a hole in the center of a fiber.

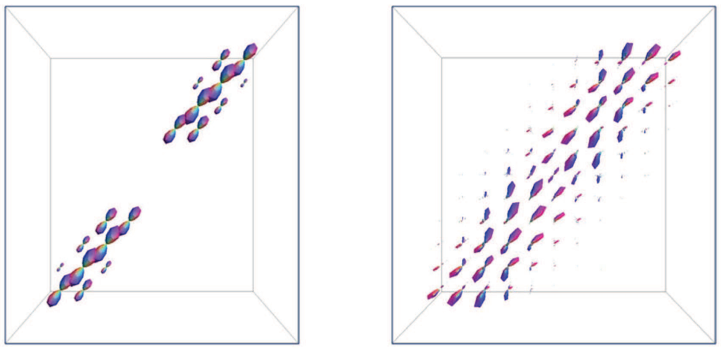


Figure 6: Initial, synthetic data showing a fiber with missing data (left) as well as the result of contour completion (right). $a_3 = 1, D_{44} = 0.01$, $\lambda = 0.25$.

5.1. Splitting scheme for solving convection-diffusion equations

To solve the general convection diffusion equations on $\mathbb{R}^3 \rtimes S^2$, we need a scheme that can handle both convection and diffusion. Unfortunately, convection and diffusion impose different constraints on the required time step for the finite difference scheme. Numerical convection requires time steps equal to the spatial grid size (in 1D $\Delta t = \Delta x$) to prevent blurring due to interpolation, while the diffusion steps require Δt small for stability and accuracy. This is problematic when simulating both processes at the same time. To cope with this problem, we use a scheme in which diffusion steps and convection steps are applied alternately, which essentially means that we ignore the fact that convection and diffusion influence each other and is therefore only first order accurate. This scheme was first proposed by Strang [29] and later applied to convection diffusion equations by Yanenko [34]. It is also used more recently in related problems [24].

Consider the following evolution equation:

$$\begin{cases} \partial_t W(\mathbf{x}, \mathbf{n}, t) = (A+B)W(\mathbf{x}, \mathbf{n}, t), \\ W(\mathbf{x}, \mathbf{n}, 0) = U(\mathbf{x}, \mathbf{n}), \end{cases} \tag{5.4}$$

where $A = (\mathcal{A}_4)^2 + (\mathcal{A}_5)^2 = \Delta_{\text{LB}}$ the spherical diffusion operator and $B = -\mathcal{A}_3$ the convection operator (although the proof holds true for any two linear operators). The solution to this equation at time Δt can symbolically be denoted by $S_{A+B}^{\Delta t} U(\mathbf{x}, \mathbf{n})$, where $S_{A+B}^{\Delta t} = e^{\Delta t(A+B)}$ (although the idea holds true for any two linear operators). It should be stressed that A and B have a nonzero commutator, so $[A, B] = AB - BA \neq \mathbf{0}$. Since the exact solution is difficult to calculate, we instead approximate this by splitting the two operators, so instead we solve $S_{A+B}^{\Delta t} \approx S_A^{\Delta t} S_B^{\Delta t}$. The error of this approximation becomes apparent when considering the Taylor series of the exponential function:

$$e^{\Delta t(A+B)} = I + \Delta t(A+B) + \Delta t^2 (A+B)^2 + \mathcal{O}(\Delta t^3), \tag{5.5a}$$

$$e^{\Delta t A} = I + \Delta t A + \Delta t^2 A^2 + \mathcal{O}(\Delta t^3), \tag{5.5b}$$

from which it follows that

$$
\begin{aligned}
&e^{\Delta t(A+B)} - e^{\Delta t A} e^{\Delta t B} \\
=&\frac{\Delta t^2}{2}(BA - AB) + \mathcal{O}(\Delta t^3) = \frac{\Delta t^2}{2}[B, A] + \mathcal{O}(\Delta t^3),
\end{aligned} \tag{5.6}
$$

and hence this scheme is only first order accurate.

There is a simple trick to easily get such a scheme one order more accurate [4,29] by splitting the diffusion step in two halves: $S_{A+B}^{\Delta t} \approx S_A^{\Delta t/2} S_B^{\Delta t} S_A^{\Delta t/2}$. The error of this method can be evaluated straightforwardly in the same way:

$$
e^{\Delta t(A+B)} - e^{\frac{\Delta t}{2}A} e^{\Delta t B} e^{\frac{\Delta t}{2}A} = \mathcal{O}(\Delta t^3), \tag{5.7}
$$

where the $\mathcal{O}(\Delta t^2)$-term from Eq. (5.6) has been removed due to the splitting of A into two parts.

In practice this means that the convection step uses a time step close to the grid size Δx, while the diffusion step can be split up into multiple smaller steps for stability, where half of these diffusion steps are performed before the convection step and half after. This means that there is no additional computational complexity for performing this splitting scheme.

In our case, we set $a_3 = 1$ without loss of generality (using other values for a_3 is equivalent to rescaling the total simulation time t and D_{44}), The integral in Eq. (5.2) can then easily be solved numerically by converting it to a sum over the convection time steps

$$
R(\mathbf{y}, \mathbf{n}) = \int_0^\infty W(\mathbf{y}, \mathbf{n}, t) \cdot p(t) dt \approx \sum_{t=0}^{t_{\max}} W(\mathbf{y}, \mathbf{n}, t) \cdot p(t), \tag{5.8}
$$

where $t_{\max}$ is a value such that $p(t_{\max})$ is sufficiently small. In our experiments we have used $t_{\max} = 10$.

6. Non-linear diffusion processes on $\mathbb{R}^3 \rtimes S^2$

Recall from the introduction that we aim to restrict diffusion from the ventricles, which show up as large isotropic diffusion profiles to the neuronal fibers, which show up as smaller, anisotropic diffusion profiles which are sometimes located very close to each other.

A Perona-Malik [20] type scheme for diffusion can separate these two regions and apply the diffusion within the neural tracts and within the ventricles, but prevents transport from one to the other by hindering diffusion in areas of large gradient.

Other reasonable extensions (not considered here) to nonlinear data adaptive diffusion processes on $\mathbb{R}^3 \rtimes S^2$ are coherence enhancing diffusion (CED) [5,33] and Laplace-Beltrami (LB) diffusion [27]. In the case of CED one replaces $D = [D_{ij}]$, $D > 0$, $D^T = D$ by a data dependent $D(W(\cdot, \cdot, t)) = [D_{ij}(W(\cdot, \cdot, t))]$ in order to adapt the conductivity to the data.

In the case of Laplace-Beltrami-framework one adapts the metric tensor $\mathbf{g}$ to the data. In this framework, the equivalent generator becomes

$$\frac{1}{\sqrt{\det(\mathbf{g})}}\sum_{i,j}\mathcal{A}_i(\sqrt{\det(\mathbf{g})}g^{ij}\mathcal{A}_j)=\sum_{i,j}\frac{1}{2}\frac{\mathcal{A}_i(\det(\mathbf{g}))}{\sqrt{\det(\mathbf{g})}}g^{ij}\mathcal{A}_j+g^{ij}\mathcal{A}_i\mathcal{A}_j, \tag{6.1}$$

where g^{ij} are components of the inverse metric tensor $\mathbf{g}^{-1}=\sum_{i,j}g^{ij}\mathcal{A}_i\otimes\mathcal{A}_j$ and are data dependent: so $g^{ij}=g^{ij}(W(\cdot,\cdot,t)\in\mathbb{R}$. In short, LB diffusion produces an extra term in the generator compared to the convection-diffusion generator of Eq. (2.15) if one sets $[D_{ij}]=[g^{ij}]$. The implementation and application of CED and LB are beyond the scope of this paper, but are worthwhile avenues for future research.

6.1. Perona-Malik diffusion on $\mathbb{R}^3 \rtimes S^2$

Our approach is similar to recent work by Burgeth et al. [5] who used adaptive, edge preserving diffusion on the DTI tensor components separately. The difference is that here the diffusion considers both positions and orientations in the domain and therefore separates two crossing fibers in the domain so that it is better equipped to handle crossing structures.

We test the algorithm on a synthetic test image consisting of two crossing fibers consisting of oriented glyphs surrounded by isotropic spheres, (see Fig. 7) in which linear diffusion destroys the fiber structure, whereas nonlinear adaptive diffusion both preserves the fiber structures and denoises the entire dataset.

Mutual influence of the anisotropic regions (fibers) and isotropic regions (ventricles) is avoided by replacing the constant diffusivity D_{33} in Eq. (2.15) by

$$\mathcal{A}_3D_{33}\mathcal{A}_3\mapsto\mathcal{A}_3\circ D_{33}e^{-\frac{(\mathcal{A}_3W(\cdot,t))^2}{K^2}}\circ\mathcal{A}_3, \tag{6.2}$$

where for $K\to\infty$, linear contour enhancement is obtained. The idea is to set a soft threshold (determined by K) on the amount of diffusion in $\mathcal{A}_3$ direction. Within homogeneous regions one expects $|\mathcal{A}_3W(\mathbf{y},\mathbf{n},t)|$ to be small, whereas in the transition areas between ventricles and white matter where one needs to block the diffusion process, one expects a large $|\mathcal{A}_3W(\mathbf{y},\mathbf{n},t)|$.

Remark 6.1. Since the Perona-Malik diffusion is nonlinear, the constant K is dependent on any normalization which may have been applied to U. If $\Phi_K(U)$ represents nonlinear diffusion with parameter K on data U then $\Phi_K(U)=\Phi_{cK}(cU)/c$, $\forall c\neq0$. Parameter K is typically chosen based on a histogram of $|\mathcal{A}_3U|$ in order to get the proper scaling invariance, as is illustrated in Fig. 8.

To implement this, we propose the following discretization scheme

$$\begin{aligned}\mathcal{A}_3(\tilde{D}_{33}\mathcal{A}_3)W(\mathbf{y},\mathbf{n},t)\approx{}&\frac{\tilde{D}_{33}(\mathbf{y}+\frac{1}{2}\mathbf{h},\mathbf{n})\mathcal{A}_3W(\mathbf{y}+\frac{1}{2}\mathbf{h},\mathbf{n},t)}{h}\\&-\frac{\tilde{D}_{33}(\mathbf{y}-\frac{1}{2}\mathbf{h},\mathbf{n})\mathcal{A}_3W(\mathbf{y}-\frac{1}{2}\mathbf{h},\mathbf{n},t)}{h},\end{aligned} \tag{6.3a}$$

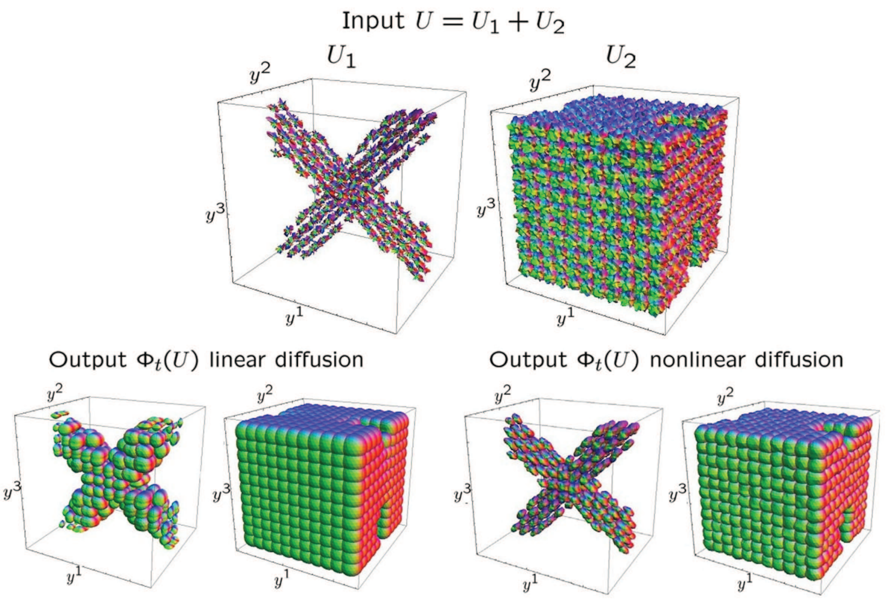

Input $U = U_1 + U_2$
U_1
U_2
y^1
y^2
y^3
Output $\Phi_t(U)$ linear diffusion
Output $\Phi_t(U)$ nonlinear diffusion

Figure 7: Adaptive Perona-Malik diffusion. Top row: Artificial $15 \times 15 \times 15 \times 162$ input data that is the sum of a noisy fiber part and a noisy isotropic part. For the sake of visualization, we depict these parts separately. Bottom row left: Output of linear diffusion with $t = 1$, $D_{33} = 1$,$D_{44} = 0.04$ and $\Delta t = 0.01$. Bottom right: Output of Perona-Malik adaptive diffusion with $D_{33} = 1$, $D_{44} = 0.015$, $K = 0.05$, $\Delta t = 0.01$, $t = 1$.

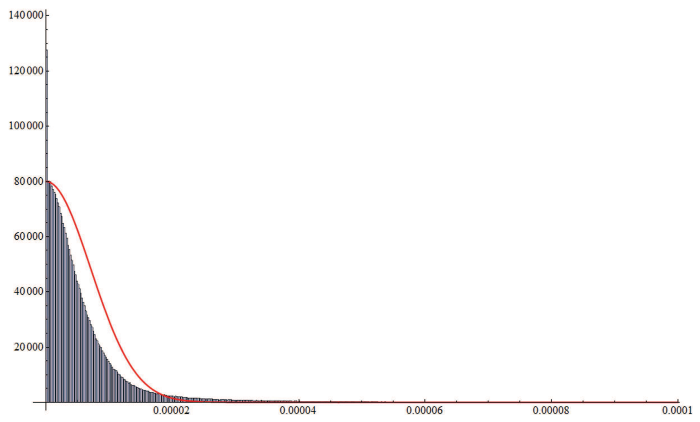


Figure 8: A histogram of $|\mathcal{A}_3 U|$ of a real dataset of dimensions $63 \times 27 \times 10$ and 162 sampled directions. The graph of Gaussian function $\exp\{-(\mathcal{A}_3 U)^2/K^2\}$ with $K = 10^{-5}$ is included in red. Perona-Malik diffusion with this parameter works well with this parameter on this data. A good rule of thumb is that the parameter K should be $1.5 \sim 2$ times higher than the standard deviation of the histogram of $|\mathcal{A}_3 U|$.

$$\mathcal{A}_3 W(\mathbf{y} + \frac{1}{2}\mathbf{h}, \mathbf{n}, t) \approx \frac{W(\mathbf{y} + \mathbf{h}, \mathbf{n}, t) - W(\mathbf{y}, \mathbf{n}, t)}{h} = \mathcal{A}_3^f W(\mathbf{y}, \mathbf{n}, t), \tag{6.3b}$$

$$\mathcal{A}_3 W(\mathbf{y} - \frac{1}{2}\mathbf{h}, \mathbf{n}, t) \approx \frac{W(\mathbf{y}, \mathbf{n}, t) - W(\mathbf{y} - \mathbf{h}, \mathbf{n}, t)}{h} = \mathcal{A}_3^b W(\mathbf{y}, \mathbf{n}, t). \tag{6.3c}$$

Combining these three equations leads to

$$\mathcal{A}_3(\tilde{D}_{33}\mathcal{A}_3)W(\mathbf{y},\mathbf{n},t) \approx \frac{\tilde{D}_{33}(\mathbf{y}+\frac{1}{2}\mathbf{h},\mathbf{n})\mathcal{A}_3^f W(\mathbf{y},\mathbf{n},t) - \tilde{D}_{33}(\mathbf{y}-\frac{1}{2}\mathbf{h},\mathbf{n})\mathcal{A}_3^b W(\mathbf{y},\mathbf{n},t)}{h}, \quad (6.4)$$

where for notational convenience $\mathbf{h} = hR_{\mathbf{n}}\mathbf{e}_z = h\mathbf{n}$ and

$$\tilde{D}_{33} = D_{33}\exp\left\{-\frac{(\max(|\mathcal{A}_3^f W|, |\mathcal{A}_3^b W|))^2}{K^2}\right\}.$$

The $\tilde{D}_{33}$ terms can easily be calculated with linear interpolation. Combined with the finite difference operators of Section 3.2, this give the full discretization scheme.

The discretization scheme for $\tilde{D}_{33}$ uses $\max(|\mathcal{A}_3^f W|, |\mathcal{A}_3^b W|)$ because forward and backwards finite difference schemes individually induce shifts near discontinuities, which are exactly the regions we want to dampen diffusion. Central finite difference schemes sometimes allow diffusion across region boundaries. This happens when fiber voxels have more than one isotropic neighbor, then $\mathcal{A}_3^c W$ may be close to zero because the stencil does not depend on the center point. Because of the spatial discretization and because every direction $\mathbf{n}$ is considered, this is very likely to occur in almost all geometries.

7. Enhancement of DTI of the human brain

In this section, we will look at the enhancement of DTI scans obtained from healthy volunteers. To do this, we first describe the process by which DTI tensor fields are converted to function of $\mathbb{R}^3 \rtimes S^2$, before showing the results of linear contour enhancement and non-linear enhancements.

7.1. Representing DTI data as functions of position and orientation

Diffusion Tensor Imaging(DTI), introduced by Basser et al. [3] assumes that the diffusion propagator p_t can be described for each voxel by an anisotropic Gaussian function, i.e.,

$$p_t(X_t = \mathbf{x}+\mathbf{r} \mid X_0 = \mathbf{x}) = \frac{1}{\sqrt{(4\pi t)^3 \det(D(\mathbf{x}))}} \exp\left(\frac{-\mathbf{r}^T D(\mathbf{x})^{-1}\mathbf{r}}{4t}\right), \quad (7.1)$$

which described the probability that water particles end up at displaced by $\mathbf{r}$ from their starting position $\mathbf{x}$ and where D is a tensor field of 3×3 positive definite symmetric tensors that describe the local Gaussian diffusion process. The tensors contain 6 parameters for each voxel, which means the tensor field requires at least 6 DW-MRI scans to compute.

The drawback of approximating p_t with an anisotropic Gaussian function is that it is only able to estimate one preferred direction per voxel. However, if more complex structures such as crossing, kissing or diverging fibers are present the Gaussian assumption fails, as was demonstrated by e.g., Alexander et al. [2]. In practice though, large areas of the brain can be approximated well with DTI tensors and in the regions where complex fiber structures are present the diffusion profile can often be inferred by taking contextual information into consideration [22].

Since DTI tensors cannot contain information regarding crossings, the DTI data needs to be represented in a form that does allow crossing fiber structures. A representation that suits these demands can be obtained by converting a DTI tensor field into an ODF. We obtain the ODF distribution associated with a DTI tensor field by inserting Eq. (7.1) into Eq. (1.1) and using $\mathbf{r} = \alpha \mathbf{n}$

$$\mathrm{ODF}(\mathbf{x}, \mathbf{n}) = \frac{1}{\sqrt{\det(D(\mathbf{x}))(4\pi t)^3}} \int_0^\infty \exp\left(-\frac{\mathbf{n}^T D^{-1}(\mathbf{x})\mathbf{n}\alpha^2}{4t}\right)\alpha^2 d\alpha. \tag{7.2}$$

This integral can be simplified (see Appendix for a derivation) to

$$\mathrm{ODF}(\mathbf{x}, \mathbf{n}) = \frac{1}{4\pi\sqrt{\det(D(\mathbf{x}))}} \left(\frac{1}{\mathbf{n}^T D^{-1}(\mathbf{x})\mathbf{n}}\right)^{\frac{3}{2}}. \tag{7.3}$$

It should be noted that this step can only be done provided that D is positive definite (otherwise the integral of Eq. (7.2) does not converge for all n). In experimental data, sometimes tensors with small negative eigenvalues are present because not all tensor fitting algorithms enforce positive definitiveness in the fitting process. This means that the ODF does not exist for these tensors and a better tensor fitting algorithm needs to be used for these positions in the data.

The ODF distribution is normalized per position:

$$\int_{S^2} \mathrm{ODF}(\mathbf{x}, \mathbf{n}) d\sigma(n) = 1, \quad \forall \mathbf{x} \in \mathbb{R}^3, \tag{7.4}$$

which means it is not dependent on the strength of diffusion. For example, multiplying a diffusion tensor with a constant will not affect the ODF of that tensor (this is also why the diffusion time t has canceled out in the equation). This means that in the conversion from DTI tensor to ODF some information has been lost.

To get a measure that *is* dependent on the total amount of diffusion, we can introduce an extra factor that is sensitive to the total amount of diffusion. A factor suitable for this is $\sqrt{\det(D(\mathbf{x}))}$, which is proportional to volume of the DTI ellipsoid associated with tensor $D(\mathbf{x})$. This ellipsoid is defined by $\mathbf{r}^T D^{-1}(\mathbf{x})\mathbf{r} = 1$ and has radii equal to the square root of the eigenvalues of $D(\mathbf{x})$ and volume equal to $4\pi\sqrt{\det(D(\mathbf{x}))}/3$. It is also a convenient factor, because it cancels one of the factors of Eq. (7.3). Another justification for multiplying by $\sqrt{\det(D(\mathbf{x}))}$ is because we are interested in viewing DW-MRI data as a probability density function of position and orientation simultaneously. In short, we want to know $p_t(X_t = \mathbf{x} + \mathbf{r} \text{ and } X_0 = \mathbf{x})$ instead of the diffusion propagator $p_t(X_t = \mathbf{x} + \mathbf{r} \mid X_0 = \mathbf{x})$. Of course, using elementary probability theory we can say that

$$p_t\big(X_t = \mathbf{x} + \mathbf{r} \text{ and } X_0 = \mathbf{x}\big) = p_t(X_t = \mathbf{x} + \mathbf{r} \mid X_0 = \mathbf{x}) \cdot p(X_0 = \mathbf{x}).$$

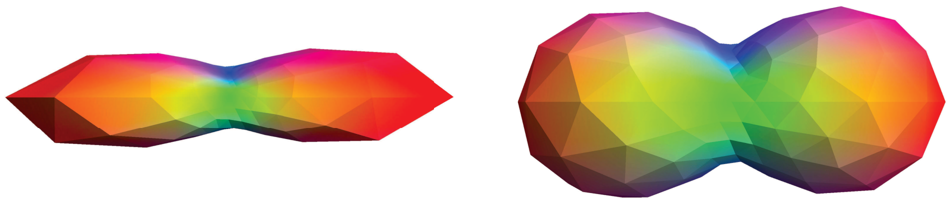


Figure 9: Comparison of the old and new methods on the same DTI tensor obtained from medical data. The left glyph shows the data the new method generated by Eq. (7.5) and the right glyph shows the old method described in Eq. (7.6).

We then assume that $p(X_0 = \mathbf{x}) \propto \sqrt{\det(D(\mathbf{x}))}$, i.e., the a priori probability that water particles start their diffusion at $\mathbf{x}$ is proportional to the total amount of diffusion at that position which seems a reasonable assumption.

If we perform this multiplication and collect all the constant factors in c, we obtain the following formula:

$$U(\mathbf{x},\mathbf{n}) = c\left(\mathbf{n}^T D^{-1}(\mathbf{x})\mathbf{n}\right)^{-\frac{3}{2}}, \tag{7.5}$$

where we choose $c = 1$ for convenience. Note that the choice of c is not arbitrary because it influences parameter K from the nonlinear diffusions, see Remark 6.1.

In our previous work [7, 10], an ad hoc method was used to convert DTI tensors to functions of position and orientation using a simple quadratic form:

$$U'(\mathbf{x},\mathbf{n}) = c_2\mathbf{n}^T D(\mathbf{x})\mathbf{n}, \tag{7.6}$$

which in general gives completely different results and in practice tends to make glyphs more isotropic. This extra anisotropy makes using oriented contextual information much more difficult, since extra sharpening steps become critical to obtain the orientation information in the data.

To conclude this section, we present Fig. 9 to visualize the difference between the old and new methods for one DTI tensor and Fig. 10 shows this difference on a coronal cross-section of the corpus callosum.

7.2. Experimental results

To test the algorithm on real data, a DTI brain scan was acquired from a healthy volunteer with 132 gradient directions and a b-value of 1000s/mm^2. Linear contour enhancement (Eq. (4.1)) as well as Perona-Malik adaptive diffusion (Eq. (6.2)) was performed on it, as can be seen in Fig. 11. Prčkovska et al. [22] showed that DTI combined with enhancement techniques can extrapolate crossing information from contextual information. It is interesting to see if such a method can be improved with a Perona-Malik type scheme, especially since the ventricles may make such methods unreliable in those areas.

Since visualization of larger datasets is difficult, only coronal slices through the center of the brain are depicted, where the ventricles are visible as large, isotropic spheres.

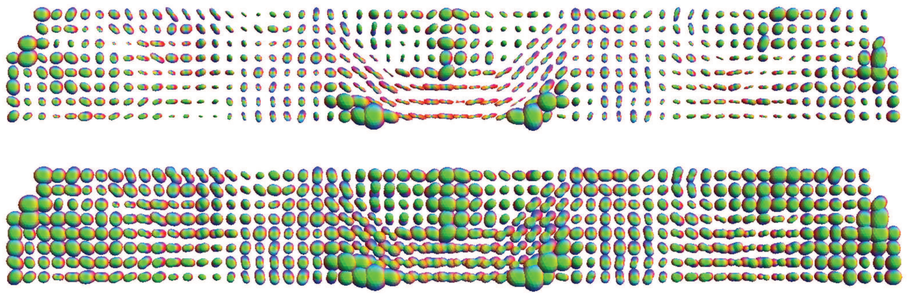
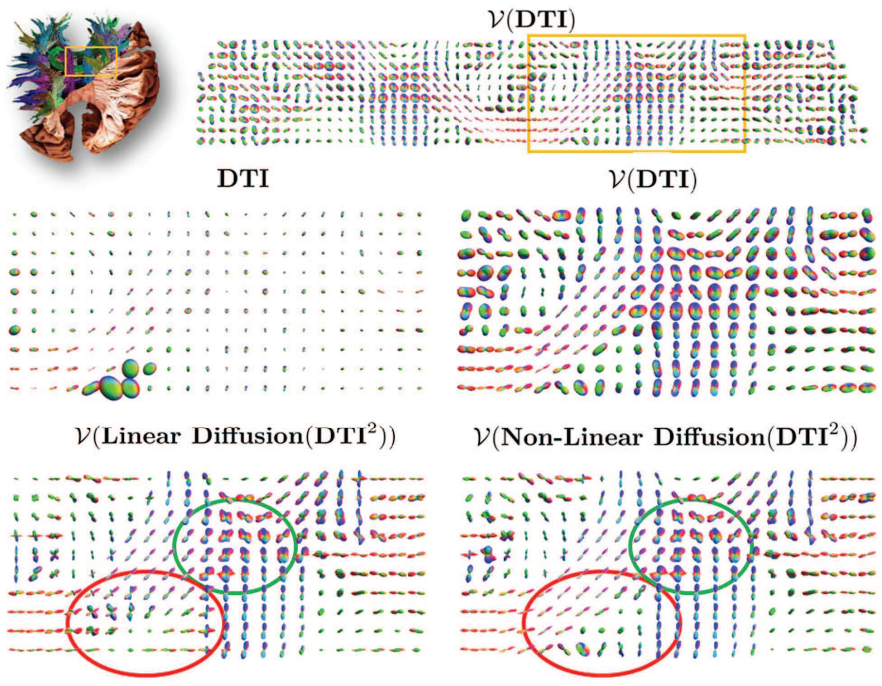


Figure 10: Comparison of the old and new methods on a field of glyphs. Shown is a coronal cross-section of the corpus callosum. Top row: new method according to Eq. (7.5). Bottom row: old method according to Eq. (7.6).


$\mathcal{V}$(DTI)
DTI
$\mathcal{V}$(DTI)
$\mathcal{V}$(Linear Diffusion(DTI^2))
$\mathcal{V}$(Non-Linear Diffusion(DTI^2))
Figure 11: DTI data of the corpus callosum and corona radiata fibers in a human brain with b-value $1000\mathrm{s/mm}^2$ and 132 gradient directions on voxels of $(2\mathrm{mm})^3$. Top row: A coronal slice of the original data with a region of interest in the yellow square. Middle row: The unprocessed region of interest by applying Eq. (7.5) (left) and same region with min-max normalization and sharpening according to Eq. (7.7) (right). Bottom row: The result of linear contour enhancement (left) and Perona-Malik diffusion (right). Parameters for both: $t = 1$, $\Delta t = 0.01$, $D_{33} = 1$, $D_{44} = 0.01$ and for Perona-Malik $K = 1 \times 10^{-9}$. Marked in red are areas in which the ventricles have induced crossing structures in the linear diffusion process. Marked in green are areas known to have crossing fibers.

Because inducing crossings requires sharp peeks, sharpening techniques have to be employed. Squaring the input data is the simplest way to do this (and is used here), but other techniques such as $\mathbb{R}^3 \rtimes S^2$-erosions are also an option [9]. For visualization, a min-max-normalization and another sharpening step are used, given by operator

$$\mathcal{V}(U)(\mathbf{y},\mathbf{n}) = \left(\frac{U(\mathbf{y},\mathbf{n}) - U_{\min}(\mathbf{y})}{U_{\max}(\mathbf{y}) - U_{\min}(\mathbf{y})}\right)^2, \quad \text{with} \quad U_{\substack{\min\\ \max}}(\mathbf{y}) = \substack{\min\\ \max}\{U(\mathbf{y},\mathbf{n}) \mid \mathbf{n} \in S^2\}. \tag{7.7}$$

From Fig. 11 it can be seen that the Perona-Malik method performs better than linear contour enhancement. The first effect is visible on the boundary of the data. Linear contour enhancement diffuses signal outside of the boundaries of the image (because of zero padding boundary conditions for the calculation of derivatives), which causes artifacts visible as horizontal structures near the top and bottom edges. The same zero padding ensures a large derivative at these places so Perona-Malik does not suffer from this problem.

The second effect is visible around the ventricles (marked by red in Fig. 11). Linear diffusion shows the influence of the ventricles in every direction, particularly creating anatomically incorrect fiber crossings in the corpus callosum (towards the top left from the ventricles) and corona radiata (towards the right of the ventricles).

Both methods are able to properly infer information about crossings, shown in the green circle, which is a location known to have crossing fibers. This shows that even though the diffusion from the ventricles has been suppressed, crossings in the data can still be inferred from the surrounding data.

8. Conclusions

We have numerically solved the diffusion and convection-diffusion equations on DW-MRI data. We have shown that these equations can be used to enhance the DW-MRI data, to reduce noise and to infer crossings from contextual information. We have derived the finite difference schemes for the convection-diffusion process on DW-MRI, where we compared explicit and implicit time integration schemes. We have developed an edge-preserving, adaptive Perona-Malik diffusion process using finite difference schemes. We have shown that this adaptive diffusion process performs better than linear diffusion in areas which contain large isotropic diffusion profiles such as the ventricles of the brain. Since the non-linear diffusion strongly reduces the interference of the isotropic glyphs of the ventricles on the anisotropic glyphs of the fibers, the need to segment these areas and the involved preprocessing beforehand is thereby eliminated.

Appendix: Derivation of Eq. (7.3)

Inserting Eq. (7.1) into Eq. (1.1) yields Eq. (7.2), repeated here for clarity:

$$\mathrm{ODF}(\mathbf{x},\mathbf{n}) = \frac{1}{\sqrt{\det(D(\mathbf{x}))(4\pi t)^3}} \int_0^\infty \exp\left(-\frac{\mathbf{n}^T D^{-1}(\mathbf{x})\mathbf{n}\alpha^2}{4t}\right)\alpha^2 d\alpha.$$

For ease of notation we collect all terms independent of α into single symbols

$$A = \frac{1}{\sqrt{\det(D(\mathbf{x}))(4\pi t)^3}} > 0, \quad B = \frac{\mathbf{n}^T D^{-1}(\mathbf{x})\mathbf{n}}{4t} > 0,$$

where formally $A = A(\mathbf{x}, t)$ and $B = B(\mathbf{x}, \mathbf{n}, t)$ which we will ignore here for notational brevity. A and B are positive because $t > 0$ and $D(\mathbf{x})$ and $D^{-1}(\mathbf{x}))$ are positive definite. The equation then simplifies to

$$\mathrm{ODF}(\mathbf{x}, \mathbf{n}) = A \int_0^\infty \exp(-B\alpha^2)\alpha^2 d\alpha.$$

By integration by parts we find

$$A \int_0^\infty \alpha \exp(-B\alpha^2)\alpha d\alpha = \frac{A}{2B} \int_0^\infty \exp(-B\alpha^2) d\alpha = \frac{A\sqrt{\pi}}{4B^{\frac{3}{2}}},$$

where the last integral is the Gaussian integral

$$\int_0^\infty \exp(-B\alpha^2) d\alpha = \frac{\sqrt{\pi}}{2\sqrt{B}}.$$

Finally we fill in A and B (formatting to emphasize the canceling terms) to arrive at Eq. (7.3)

$$\begin{aligned} \mathrm{ODF}(\mathbf{x}, \mathbf{n}) &= \frac{A\sqrt{\pi}}{4B^{\frac{3}{2}}} = \frac{\sqrt{\pi}}{\sqrt{\det(D(\mathbf{x}))}(4\pi t)^{\frac{3}{2}}} \cdot \frac{1}{4} \left(\frac{4t}{\mathbf{n}^T D^{-1}(\mathbf{x})\mathbf{n}} \right)^{\frac{3}{2}} \\ &= \frac{1}{4\pi \sqrt{\det(D(\mathbf{x}))}} \left(\frac{1}{\mathbf{n}^T D^{-1}(\mathbf{x})\mathbf{n}} \right)^{\frac{3}{2}}. \end{aligned}$$

References

[1] I. Aganj, C. Lenglet and G. Sapiro, *ODF reconstruction in q-ball imaging with solid angle consideration*, In Biomedical Imaging: From Nano to Macro, 2009. ISBI '09. IEEE International Symposium on, pages 1398–1401, 28 July 2009.

[2] D. C. Alexander, G. J. Barker and S. R. Arridge, *Detection and modeling of non-Gaussian apparent diffusion coefficient profiles in human brain data*, Magnetic Resonance in Medicine, 48 (2002), pp. 331–340.

[3] P. J. Basser, J. Mattiello and D. Lebihan, *MR diffusion tensor spectroscopy and imaging*, Biophys. J., 66 (1994), pp. 259–267.

[4] A. V. Bobylev and T. Ohwada, *The error of the splitting scheme for solving evolutionary equations*, Appl. Math. Lett., 14 (2001), pp. 45–48.

[5] B. Burgeth, L. Pizarro, S. Didas and J Weickert, *Coherence-enhancing diffusion for matrix fields*, Locally Adaptive Filtering in Signal and Image Proc., to appear.

[6] G. S. Chirikjian and A. B. Kyatkin, Engineering Applications of Noncommutitative Harmonic Analysis: with Emphasis on Rotation and Motion Groups, Boca Raton CRC Press, 2001.

[7] E. J. Creusen, R. Duits and T. Dela Haije, *Numerical schemes for linear and non-linear enhancement of DW-MRI*, In A. M. Bruckstein, editor, Proceedings of the 3rd International Conference on Scale Space and Variational Methods in Computer Vision, LNCS 6667, 2012.

[8] M. Descoteaux, High Angular Resolution Diffusion MRI: From Local Estimation to Segmentation and Tractography. PhD thesis, Universite de Nice, 2008.

[9] R. Duits, T. Dela Haije, A. Ghosh, E. J. Creusen, A. Vilanova and B. ter Haar Romeny, *Fiber enhancement in diffusion-weighted MRI,* In A. M. Bruckstein, editor, Proceedings of the 3rd International Conference on Scale Space and Variational Methods in Computer Vision, LNCS 6667, 2012.

[10] R. Duits and E. Franken, *Left-invariant diffusions on the space of positions and orientations and their application to crossing-preserving smoothing of HARDI images*, Int. J. Comput. Vision, 40 (2010).

[11] R. Duits and M. van Almsick, *The explicit solutions of linear left-invariant second order stochastic evolution equations on the 2D Euclidean motion group*, Quart. Appl. Math., 66 (2008), pp. 27–67.

[12] Murray Eisenberg and Robert Guy, *A proof of the hairy ball theorem*, The American Mathematical Monthly, 86(7) (1979), pp. 571–574.

[13] E. M. Franken, R. Duits and B. M. ter Haar Romeny, *Diffusion on the 3D euclidean motion group for enhancement of HARDI data*, In Scale Space and Variational Methods in Computer Vision (Lecture Notes in Computer Science), Volume 5567, pages 820–831, Heidelberg, 2009. Springer-Verlag.

[14] S. Gerschgorin, *Über die Abgrenzung der Eigenwerte einer Matrix*, Izv. Akad. Nauk. USSR Otd. Fix.-Mat. Nauk., 7 (1931), pp. 749–754.

[15] Y. Gur, O. Pasternak and N. Sochen, *SPD tensors regularization via Iwasawa decomposition*, In L. Florack, R. Duits, G. Jongbloed, M.-C. van Lieshout and L. Davies, editors, Mathematical Methods for Signal and Image Analysis and Representation, Springer, 2012.

[16] Y. Gur and N. Sochen, *Fast invariant Riemannian DT-MRI regularization*, In IEEE 11th International Conference on Computer Vision, 2007. ICCV 2007, pages 1–7, 2007.

[17] Michael T. Heath, Scientific Computing: An Introductory Survey, McGraw-Hill, International second edition, 2005.

[18] D. Mumford, *Elastica and computer vision*, Algebraic Geometry and Its Applications, Springer-Verlag, pages 491–506, 1994.

[19] O. Pasternak, Y. Assaf, N. Intrator and N. Sochen, *Variational multiple-tensor fitting of fiber-ambiguous diffusion-weighted magnetic resonance imaging voxels*, Magnetic Resonance Imaging, 26 (2008), pp. 1133–1144.

[20] P. Perona and J. Malik, *Scale-space and edge detection using anisotropic diffusion*, IEEE Trans. Pattern Anal. Mach. Intell., 12(7) (1990).

[21] J. Polzehl, Mathematical Methods for Signal and Image Analysis and Representation, pages 71–89, Springer-Verlag, 2011/2012. in press.

[22] V. Prčkovska, P. R. Rodrigues, R. Duits, B. M. ter Haar Romeny and A. Vilanova, *Extrapolating fiber crossings from DTI data. can we gain the same information as HARDI*? In Workshop on Computational Diffusion MRI, MICCAI; Beijing, China, 2010.

[23] P. Rodrigues, R. Duits, A. Vilanova and B. M. ter Haar Romeny, *Accelerated Diffusion Operators for Enhancing DW-MRI,* In Eurographics Workshop on Visual Computing for Biology and Medicine, pages 49–56, Leipzig, Germany, 2010.

[24] G. Rosman, L. Dascal, X. C. Tai and R. Kimmel, *On semi-implicit splitting schemes for the Beltrami color image filtering*, J. Math. Image. Vision, 40(2) (2011), pp. 199–213.

[25] P. Savadjiev, Perceptual Organisation in Diffusion MRI: Curves and Streamline Flows, PhD thesis, McGill University, 2010.

[26] P. Savadjiev, J. S. W. Campbell, G. B. Pike and K. Siddiqi, *3D curve inference for diffusion MRI regularization and fibre tractography*, Medical Image Anal., 10(5) (2006), pp. 799–813.

[27] A. Spira, R. Kimmel and N. Sochen, *A short-time Beltrami kernel for smoothing images and manifolds*, IEEE Trans. Image Process., 16(6) (2007), pp. 1628–1636.

[28] E. O. Stejskal and J. E. Tanner, *Spin diffusion measurements: spin echoes in the presence of a time-dependent field gradient*, The Journal of Chemical Physics, 42(1) (1965).

[29] G. Strang, *On the construction and comparison of difference schemes*, SIAM J. Numer. Anal., 5(3) (1968), pp. 506–517.

[30] K. Tabelow, S. S. Keller, S. Mohammadi, H. Kugel, J. S. Gerdes, J. Polzehl and M. Deppe, *Structural adaptive smoothing increases sensitivity of DTI to detect microstructure grey matter alterations*, Poster at the 17th Annual Meeting of the Organization for Human Brain Mapping, June 26-30, 2011, Quéec City, Canada, 2011.

[31] K. Tabelow, J. Polzehl, V. Spokoiny and H. U. Voss, *Diffusion tensor imaging: Structural adaptive smoothing*, NeuroImage, 39(4) (2008), pp. 1763–1773.

[32] D. S. Tuch, T. G. Reese, M. R. Wiegell, N. Makris, J. W. Belliveau and V. J. Wedeen, High Angular Resolution Diffusion Imaging Reveals Intravoxel White Matter Fiber Heterogeneity.

[33] J. Weickert, *Coherence-enhancing diffusion filtering*, Int. J. Comput. Vision, 31(2) (1999), pp. 111–127.

[34] N. N. Yanenko, The Method of Fractional Steps: The Solution of Problems of Mathematical Physics in Several Variables, Springer, 1971.