

# SpinDoctor User Guide

Updated October 30, 2019

The user is advised to download the latest version of the User Guide from <https://github.com/jingrebeccali/SpinDoctor>

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>4</b>
<b>3</b>	<b>Mathematical background</b>	<b>5</b>
3.1	Bloch-Torrey PDE . . . . .	5
3.2	Fitting the ADC from the dMRI signal . . . . .	6
3.3	HADC model . . . . .	6
3.4	Short diffusion time approximation of the ADC . . . . .	7
<b>4</b>	<b>Work flow</b>	<b>7</b>
4.1	Physical units . . . . .	8
4.2	User provided input files . . . . .	8
4.3	Important output quantities . . . . .	10
4.4	Important functions . . . . .	11
<b>5</b>	<b>Software components and algorithms</b>	<b>12</b>
5.1	Create cells (canonical configuration) . . . . .	12
5.2	Plot cells . . . . .	12
5.3	Create surface triangulation . . . . .	13
5.4	Plot surface triangulation . . . . .	15
5.5	Finite element mesh generation . . . . .	16
5.6	Plot FE mesh . . . . .	17
5.7	BTPDE . . . . .	18
5.8	HADC model . . . . .	19
5.9	Visualization of results . . . . .	20
<b>6</b>	<b>SpinDoctor examples</b>	<b>20</b>
6.1	Comparison of BTPDE and HADC with Short Time Approximation . . . . .	20
6.2	Permeable membranes . . . . .	21
6.3	Myelin layer . . . . .	22
6.4	Twisting and bending . . . . .	23
<b>7</b>	<b>Neuron Module</b>	<b>24</b>
7.1	Work flow . . . . .	25
7.2	User provided input files . . . . .	25
7.3	Neuron Module examples . . . . .	27

7.4	Finite elements meshes of neurons	29
7.5	Commented driver for typical simulations	33

# 1 Introduction

The MATLAB Toolbox SpinDoctor [1] is a simulation pipeline that

1. allows the user to define a geometrical configuration;
2. solves the Bloch-Torrey equation in that geometrical configuration;
3. fits the apparent diffusion coefficient from the simulated signal.

It includes two other methods for calculating the apparent diffusion coefficient:

1. The first is a homogenized apparent diffusion coefficient mathematical model, which was obtained recently using homogenization techniques on the Bloch-Torrey equation. In the homogenized model, the apparent diffusion coefficient of a geometrical configuration can be computed after solving a diffusion equation subject to a time-dependent Neumann boundary condition, under the assumption of negligible water exchange between compartments.
2. The second module computes the short time approximation formula for the apparent diffusion coefficient. The short time approximation implemented in SpinDoctor includes a recent generalization of this formula to account for finite pulse duration in the pulsed gradient spin echo.

Both of these two apparent diffusion coefficient calculations are sensitive to the diffusion-encoding gradient direction, unlike many previous works where the anisotropy is neglected in analytical model development.

In nutshell, SpinDoctor

1. solves the Bloch-Torrey equation in three dimensions to obtain the diffusion magnetic resonance imaging signal;
2. robustly fits the diffusion magnetic resonance imaging signal to obtain the apparent diffusion coefficient;
3. solves the homogenized apparent diffusion coefficient model in three dimensions to obtain the apparent diffusion coefficient;
4. computes the short-time approximation of the apparent diffusion coefficient;
5. computes useful geometrical quantities such as the compartment volumes and surface areas;
6. allows permeable membranes for the Bloch-Torrey equation (the homogenized apparent diffusion coefficient assumes negligible permeability);
7. displays the gradient-direction dependent signal or apparent diffusion coefficient in three dimensions.

SpinDoctor provides the following built-in functionalities:

1. the placement of non-overlapping spherical cells (with an optional nucleus) of different radii close to each other;
2. the placement of non-overlapping cylindrical cells (with an optional myelin layer) of different radii close to each other in a canonical configuration where they are parallel to the  $z$ -axis;
3. the inclusion of an extra-cellular space that is enclosed either
  - (a) in a tight wrapping around the cells; or
  - (b) in a rectangular box.
4. the deformation of the canonical configuration by bending and twisting.

Built-in diffusion-encoding pulse sequences include

1. the Pulsed Gradient Spin Echo;
2. the Oscillating Gradient Spin Echo (cos- and sin- type gradients).

SpinDoctor uses the following methods:

1. it generates a good quality surface triangulation of the user specified geometrical configuration by calling built-in MATLAB computational geometry functions;
2. it creates a good quality tetrahedra finite elements mesh from the above surface triangulation by calling Tetgen [2], an external package (executable files are included in the Toolbox package);
3. it constructs finite element matrices for linear finite elements on tetrahedra (P1 finite elements) using routines from [3];
4. it adds additional degrees of freedom on the compartment interfaces to allow permeability conditions for the Bloch-Torrey equation using the formalism in [4];
5. it solves the semi-discretized finite elements method equations by calling built-in MATLAB routines for solving ordinary differential equations.

## Abbreviations

MRI - magnetic resonance imaging

dMRI - diffusion magnetic resonance imaging

ADC - apparent diffusion coefficient

HADC - homogenized ADC

PGSE - pulsed gradient spin echo

OGSE - oscillating gradient

ECS - extra-cellular space

BTPDE - Bloch-Torrey partial differential equation

PDE - partial differential equation

ODE - ordinary differential equation

HARDI - high angular resolution diffusion imaging

STA - short time approximation

FE - finite elements

FEM - finite elements method

## 2 Installation

The SpinDoctor toolbox has been tested with MATLAB R2017b and requires no additional MATLAB toolboxes. The following software is publicly available:

1. The original SpinDoctor toolbox for simulations of brain white matter geometries is in the *master branch* of

<https://github.com/jingrebeccali/SpinDoctor>

2. The Neuron Module is in the *NeuronModule branch*:

<https://github.com/jingrebeccali/SpinDoctor/tree/NeuronModule>

The user is advised to download the latest version of the User Guide from <https://github.com/jingrebeccali/SpinDoctor>. Examples of drivers that run some typical simulations also can be found there.

### 3 Mathematical background

Suppose the user would like to simulate a geometrical configuration of cells with an optional myelin layer or a nucleus. If spins will be leaving the cells or if the user wants to simulate the extra-cellular space (ECS), then the ECS will enclose the geometrical shapes. Let  $\Omega^e$  be the ECS,  $\Omega_i^{in}$  the nucleus (or the axon) and  $\Omega_i^{out}$  the cytoplasm (or the myelin layer) of the  $i$ th cell. We denote the interface between  $\Omega_i^{in}$  and  $\Omega_i^{out}$  by  $\Gamma_i$  and the interface between  $\Omega_i^{out}$  and  $\Omega^e$  by  $\Sigma_i$ , finally the outside boundary of the ECS by  $\Psi$ .

#### 3.1 Bloch-Torrey PDE

In diffusion MRI, a time-varying magnetic field gradient is applied to the tissue to encode water diffusion. Denoting the effective time profile of the diffusion-encoding magnetic field gradient by  $f(t)$ , and letting the vector  $\mathbf{g}$  contain the amplitude and direction information of the magnetic field gradient, the complex transverse water proton magnetization in the rotating frame satisfies the Bloch-Torrey PDE:

$$\frac{\partial}{\partial t} M_i^{in}(\mathbf{x}, t) = -I\gamma f(t) \mathbf{g} \cdot \mathbf{x} M_i^{in}(\mathbf{x}, t) + \nabla \cdot (\sigma^{in} \nabla M_i^{in}(\mathbf{x}, t)), \quad \mathbf{x} \in \Omega_i^{in}, \quad (1)$$

$$\frac{\partial}{\partial t} M_i^{out}(\mathbf{x}, t) = -I\gamma f(t) \mathbf{g} \cdot \mathbf{x} M_i^{out}(\mathbf{x}, t) + \nabla \cdot (\sigma^{out} \nabla M_i^{out}(\mathbf{x}, t)), \quad \mathbf{x} \in \Omega_i^{out}, \quad (2)$$

$$\frac{\partial}{\partial t} M^e(\mathbf{x}, t) = -I\gamma f(t) \mathbf{g} \cdot \mathbf{x} M^e(\mathbf{x}, t) + \nabla \cdot (\sigma^e \nabla M^e(\mathbf{x}, t)), \quad \mathbf{x} \in \Omega^e, \quad (3)$$

where  $\gamma = 2.67513 \times 10^8 \text{ rad s}^{-1} \text{T}^{-1}$  is the gyromagnetic ratio of the water proton,  $I$  is the imaginary unit,  $\sigma^l$  is the intrinsic diffusion coefficient in the compartment  $\Omega_i^l$ . The magnetization is a function of position  $\mathbf{x}$  and time  $t$ , and depends on the diffusion gradient vector  $\mathbf{g}$  and the time profile  $f(t)$ . We denote the restriction of the magnetization in  $\Omega_i^{in}$  by  $M_i^{in}$ , and similarly for  $M_i^{out}$  and  $M^e$ .

Some commonly used time profiles (diffusion-encoding sequences) are:

1. The pulsed-gradient spin echo (PGSE) [5] sequence, with two rectangular pulses of duration  $\delta$ , separated by a time interval  $\Delta - \delta$ , for which the profile  $f(t)$  is

$$f(t) = \begin{cases} 1, & t_1 \leq t \leq t_1 + \delta, \\ -1, & t_1 + \Delta < t \leq t_1 + \Delta + \delta, \\ 0, & \text{otherwise,} \end{cases} \quad (4)$$

where  $t_1$  is the starting time of the first gradient pulse with  $t_1 + \Delta > T_E/2$ ,  $T_E$  is the echo time at which the signal is measured.

2. The oscillating gradient spin echo (OGSE) sequence [6, 7] was introduced to reach short diffusion times. An OGSE sequence usually consists of two oscillating pulses of duration  $T$ , each containing  $n$  periods, hence the frequency is  $\omega = n \frac{2\pi}{T}$ , separated by a time interval  $\tau - T$ . For a cosine OGSE, the profile  $f(t)$  is

$$f(t) = \begin{cases} \cos(n \frac{2\pi}{T} t), & t_1 < t \leq t_1 + T, \\ -\cos(n \frac{2\pi}{T} (t - \tau)), & \tau + t_1 < t \leq t_1 + \tau + T, \\ 0, & \text{otherwise,} \end{cases} \quad (5)$$

where  $\tau = T_E/2$ .

The Bloch-Torrey PDE needs to be supplemented by interface conditions. We recall the interface between  $\Omega_i^{in}$  and  $\Omega_i^{out}$  is  $\Gamma_i$ , the interface between  $\Omega_i^{out}$  and  $\Omega^e$  is  $\Sigma_i$ , and the outside boundary of the ECS is  $\Psi$ . The two interface conditions on  $\Gamma_i$  are the flux continuity and a condition that incorporates a permeability coefficient  $\kappa^{in,out}$  across  $\Gamma_i$ : :

$$\begin{aligned} \sigma^{in} \nabla M_i^{in}(\mathbf{x}, t) \cdot \mathbf{n}_i^{in} &= -\sigma^{out} \nabla M_i^{out}(\mathbf{x}, t) \cdot \mathbf{n}_i^{out}, & \mathbf{x} \in \Gamma_i, \\ \sigma^{in} \nabla M_i^{in}(\mathbf{x}, t) \cdot \mathbf{n}_i^{in} &= \kappa^{in,out} (M_i^{out}(\mathbf{x}, t) - M_i^{in}(\mathbf{x}, t)), & \mathbf{x} \in \Gamma_i, \end{aligned}$$

where  $\mathbf{n}$  is the unit outward pointing normal vector. Similarly, between  $\Omega_i^{out}$  and  $\Omega^e$  we have

$$\begin{aligned}\sigma^{out} \nabla M_i^{out}(\mathbf{x}, t) \cdot \mathbf{n}_i^{out} &= -\sigma^e \nabla M^e(\mathbf{x}, t) \cdot \mathbf{n}^e, & \mathbf{x} \in \Sigma_i, \\ \sigma^{out} \nabla M_i^{out}(\mathbf{x}, t) \cdot \mathbf{n}_i^{out} &= \kappa^{out,e} \left( M^e(\mathbf{x}, t) - M_i^{out}(\mathbf{x}, t) \right), & \mathbf{x} \in \Sigma_i.\end{aligned}$$

Finally, on the outer boundary of the ECS we have

$$0 = \sigma^e \nabla M^e(\mathbf{x}, t) \cdot \mathbf{n}^e, \quad \mathbf{x} \in \Psi.$$

The Bloch-Torrey PDE also needs initial conditions:

$$M_i^{in}(\mathbf{x}, 0) = \rho^{in}, \quad M_i^{out}(\mathbf{x}, 0) = \rho^{out}, \quad M^e(\mathbf{x}, 0) = \rho^e.$$

where  $\rho$  is the initial spin density.

The dMRI signal is measured at echo time  $t = T_E > \Delta + \delta$  for PGSE and  $T_E > 2\sigma$  for OGSE. This signal is the integral of  $M(\mathbf{x}, T_E)$ :

$$S := \int_{\mathbf{x} \in \cup\{\Omega_i^{in}, \Omega_i^{out}, \Omega^e\}} M(\mathbf{x}, T_E) \, d\mathbf{x}. \quad (6)$$

In a dMRI experiment, the pulse sequence (time profile  $f(t)$ ) is usually fixed, while  $\mathbf{g}$  is varied in amplitude (and possibly also in direction).  $S$  is usually plotted against a quantity called the *b*-value. The *b*-value depends on  $\mathbf{g}$  and  $f(t)$  and is defined as

$$b(\mathbf{g}) = \gamma^2 \|\mathbf{g}\|^2 \int_0^{T_E} du \left( \int_0^u f(s) ds \right)^2.$$

For PGSE, the b-value is [5]:

$$b(\mathbf{g}, \delta, \Delta) = \gamma^2 \|\mathbf{g}\|^2 \delta^2 (\Delta - \delta/3). \quad (7)$$

For the cosine OGSE with *integer* number of periods  $n$  in each of the two durations  $\sigma$ , the corresponding *b*-value is [8]:

$$b(\mathbf{g}, \sigma) = \gamma^2 \|\mathbf{g}\|^2 \frac{\sigma^3}{4n^2 \pi^2} = \gamma^2 \|\mathbf{g}\|^2 \frac{\sigma}{\omega^2}. \quad (8)$$

The reason for these definitions is that in a homogeneous medium, the signal attenuation is  $e^{-\sigma b}$ , where  $\sigma$  is the intrinsic diffusion coefficient.

### 3.2 Fitting the ADC from the dMRI signal

An important quantity that can be derived from the dMRI signal is the “Apparent Diffusion Coefficient” (ADC), which gives an indication of the root mean squared distance travelled by water molecules in the gradient direction  $\mathbf{g}/\|\mathbf{g}\|$ , averaged over all starting positions:

$$ADC := -\frac{\partial}{\partial b} \log \frac{S(b)}{S(0)} \Big|_{b=0}. \quad (9)$$

We numerically compute  $ADC$  by a polynomial fit of

$$\log S(b) = c_0 + c_1 b + \cdots + c_n b^n,$$

increasing  $n$  from 1 onwards until we get the value of  $c_1$  to be stable within a numerical tolerance.

### 3.3 HADC model

In a previous work [9], a PDE model for the time-dependent ADC was obtained starting from the Bloch-Torrey equation, using homogenization techniques. In the case of negligible water exchange between compartments (low permeability), there is no coupling between the compartments, at least to the quadratic order in  $\mathbf{g}$ , which is the ADC term. The ADC in compartment  $\Omega$  is given by

$$HADC = \sigma - \frac{1}{\int_0^{T_E} F(t)^2 dt} \int_0^{T_E} F(t) h(t) \, dt, \quad (10)$$

where  $F(t) = \int_0^t f(s) ds$ , and

$$h(t) = \frac{1}{|\Omega|} \int_{\partial\Omega} \omega(\mathbf{x}, t) (\mathbf{u}_g \cdot \mathbf{n}) ds \quad (11)$$

is a quantity related to the directional gradient of a function  $\omega$  that is the solution of the homogeneous diffusion equation with Neumann boundary condition and zero initial condition:

$$\begin{aligned} \frac{\partial}{\partial t} \omega(\mathbf{x}, t) - \nabla(\sigma \nabla \omega(\mathbf{x}, t)) &= 0, & \mathbf{x} \in \Omega, \\ \sigma \nabla \omega(\mathbf{x}, t) \cdot \mathbf{n} &= \sigma F(t) \mathbf{u}_g \cdot \mathbf{n}, & \mathbf{x} \in \partial\Omega, \\ \omega(\mathbf{x}, 0) &= 0, & \mathbf{x} \in \Omega, \end{aligned} \quad (12)$$

$\mathbf{n}$  being the outward normal and  $t \in [0, TE]$ ,  $\mathbf{u}_g$  is the unit gradient direction. The above set of equations, (10)-(12), comprise the homogenized model that we call the HADC model.

### 3.4 Short diffusion time approximation of the ADC

A well-known formula for the ADC in the short diffusion time regime is the following short time approximation (STA) [10, 11]:

$$STA = \sigma \left( 1 - \frac{4\sqrt{\sigma}}{3\sqrt{\pi}} \sqrt{\Delta} \frac{A}{\dim V} \right),$$

where  $\frac{A}{V}$  is the surface to volume ratio and  $\sigma$  is the intrinsic diffusivity coefficient. In the above formula the pulse duration  $\delta$  is assumed to be very small compared to  $\Delta$ . A recent correction to the above formula [9], taking into account the finite pulse duration  $\delta$  and the gradient direction  $\mathbf{u}_g$ , is the following:

$$STA = \sigma \left[ 1 - \frac{4\sqrt{\sigma}}{3\sqrt{\pi}} C_{\delta,\Delta} \frac{A_{\mathbf{u}_g}}{V} \right], \quad (13)$$

where

$$A_{\mathbf{u}_g} = \int_{\partial\Omega} (\mathbf{u}_g \cdot \mathbf{n})^2 ds,$$

and

$$C_{\delta,\Delta} = \frac{4}{35} \frac{(\Delta + \delta)^{7/2} + (\Delta - \delta)^{7/2} - 2(\delta^{7/2} + \Delta^{7/2})}{\delta^2(\Delta - \delta/3)} = \sqrt{\Delta} \left( 1 + \frac{1}{3} \frac{\delta}{\Delta} - \frac{8}{35} \left( \frac{\delta}{\Delta} \right)^{3/2} + \dots \right).$$

When  $\delta \ll \Delta$ , the value  $C_{\delta,\Delta}$  is approximately  $\sqrt{\Delta}$ .

## 4 Work flow

Figure 1 is a chart showing the work flow of SpinDoctor.

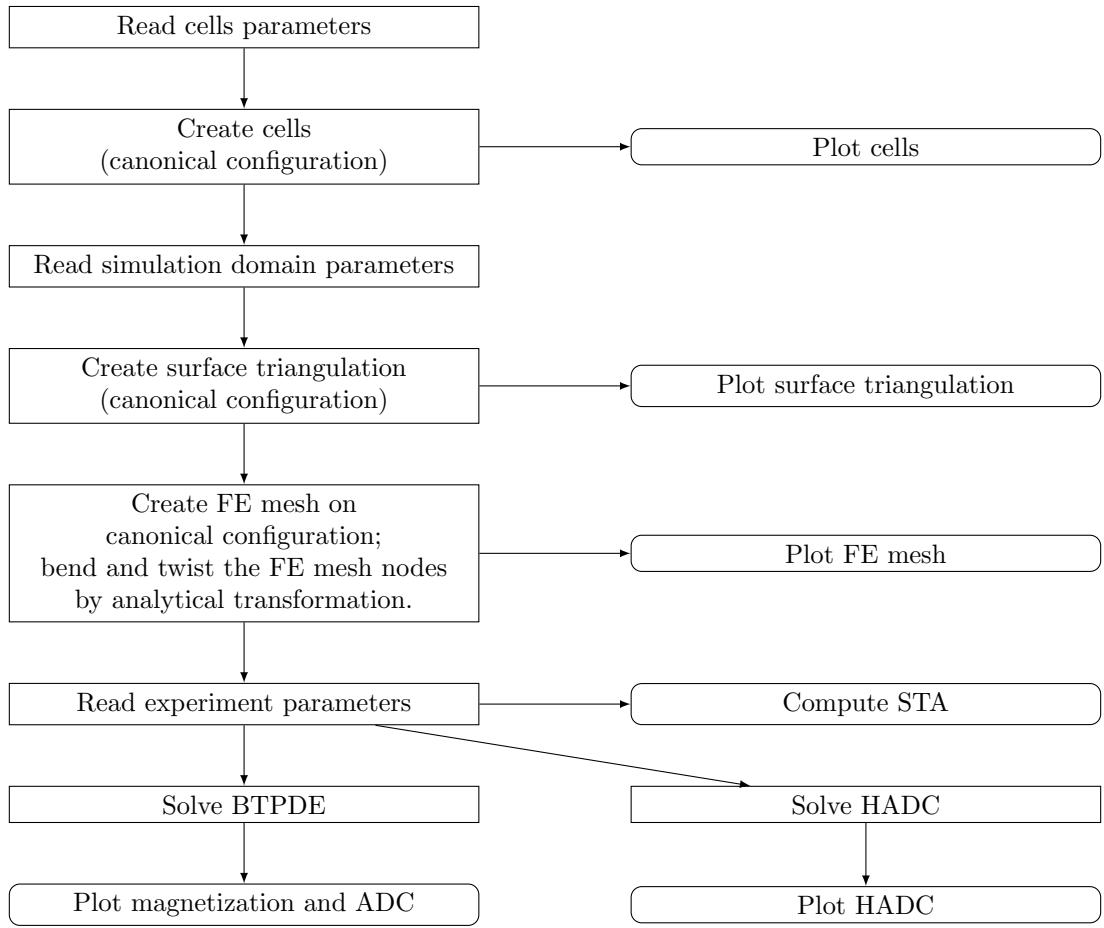


Figure 1: Flow chart describing the work flow of SpinDoctor

#### 4.1 Physical units

The physical units of the input and output quantities for SpinDoctor are shown in Table 1.

Parameter	Unit
length	$\mu\text{m}$
time	$\mu\text{s}$
diffusion coefficient	$\mu\text{m}^2/\mu\text{s} = \text{mm}^2/\text{s}$
permeability coefficient	$\mu\text{m}/\mu\text{s} = \text{m/s}$
b-value	$\mu\text{s}/\mu\text{m}^2 = \text{s/mm}^2$
q-value	$(\mu\text{s}\mu\text{m})^{-1}$

Table 1: Physical units of the input and output quantities for SpinDoctor.

#### 4.2 User provided input files

There are three input files SpinDoctor expects the user to provide:

1. Input file containing *cells parameters*. Format is given in Table 2;
2. Input file containing *simulation domain parameters*. Format is given in Table 3;
3. Input file containing *simulation experiment parameters*. Format is given in Table 4;

Line	Variable name	Example	Explanation
1	cell_shape	1	1 = spheres; 2 = cylinders;
2	fname_params_cells	'current_cells'	file name to store cells description
3	ncell	10	number of cells
4	Rmin	1.5	min Radius
5	Rmax	2.5	max Radius
6	dmin	1.5	min (%) distance between cells: $dmin \times \frac{(Rmin+Rmax)}{2}$
7	dmax	2.5	max (%) distance between cells $dmax \times \frac{(Rmin+Rmax)}{2}$
8	para_deform	0.05 0.05	$[\alpha \ \beta]$ ; $\alpha$ defines the amount of bend; $\beta$ defines the amount of twist
9	Hcyl	20	height of cylinders

Table 2: Input file containing cells parameters.

Line	Variable name	Example	Explanation
1	Rratio	0.0	if Rratio is outside [0,1], it is set to 0; else $Rratio = \frac{R_{in}}{R_{out}}$ ;
2	include_ECS	2	0 = no ECS; 1 = box ECS; 2 = tight wrap ECS;
3	ECS_gap	0.3	ECS thickness: a. if box: as percentage of domain length; b. if tight wrap: as percentage of mean radius
4	dcoeff_IN	0.002	diffusion coefficient in IN cmpt: a. nucleus; b. axon (if there is myelin);
5	dcoeff_OUT	0.002	diffusion coefficient in OUT cmpt: a. cytoplasm; b. axon (if there is no myelin);
6	dcoeff_ECS	0.002	diffusion coefficient in ECS cmpt;
7	ic_IN	1	initial spin density in In cmpt: a. nucleus; b. axon (if there is myelin)
8	ic_OUT	1	initial spin density in OUT cmpt: a. cytoplasm; b. axon (if there is no myelin);
9	ic_ECS	1	initial spin density in ECS cmpt:
10	kappa_IN_OUT	1e-5	permeability between IN and OUT cmpts: a. between nucleus and cytoplasm; b. between axon and myelin;
11	kappa_OUT_ECS	1e-5	permeability between OUT and ECS cmpts: a. if no nucleus: between cytoplasm and ECS; b. if no myelin: between axon and ECS;
12	Htetgen	-1	Requested tetgen mesh size; -1 = Use tetgen default;
13	tetgen_cmd	'SRC/TETGEN/ tetGen/win64/ tetgen'	path to tetgen_cmd

Table 3: Input file of simulation domain parameters.

Line	Variable name	Example	Explanation
1	ngdir	20	number of gradient direction; if $ngdir > 1$ , the gradient directions are distributed uniformly on a sphere; if $ngdir = 1$ , take the gradient direction from the line below;
2	gdir	1.0 0.0 0.0	gradient direction; No need to normalize;
3	nexperi	3	number of experiments;
4	sdeltavec	2500 10000 10000	small delta;
5	bdtlavec	2500 10000 10000	big delta;
6	seqvec	1 2 3	diffusion sequence of experiment; 1 = PGSE; 2 = OGSEsin; 3 = OGSEcos;
7	npervec	0 10 10	number of period of OGSE;
8	solve_hadc	1	0 = do not solve HADC; Otherwise solve HADC;
9	rtol_deff, atol_deff	1e-4 1e-4	[ $r_{tol}$ $a_{tol}$ ]; relative and absolute tolerance for HADC ODE solver;
10	solve_btpde	1	0 = do not solve BTPDE; Otherwise solve BTPDE;
11	rtol_bt, atol_bt	1e-5 1e-5	[ $r_{tol}$ $a_{tol}$ ]; relative and absolute tolerance for BTPDE ODE solver;
12	nb	2	number of b-values;
13	blimit	0	0 = specify bvec; 1 = specify [bmin,bmax]; 2 = specify [gmin,gmax];
14	const_q	0	0: use input bvalues for all experiments; 1: take input bvalues for the first experiment and use the same q for the remaining experiments
15	bvalues	0 50 100 200	bvalues or [bmin, bmax] or [gmin, gmax]; depending on line 13;

Table 4: Input file for simulation experiment parameters.

### 4.3 Important output quantities

In Table 5 we list some useful quantities that are the outputs of SpinDoctor. The braces in the "Size" column denote MATLAB cell data structure and the brackets denote MATLAB matrix data structure. We note that if the finite elements mesh is very big, the user should not output the magnetization after solving the BTPDE, rather, only the space integral of the magnetization which is the dMRI signal.

Variable name	Size	Explanation
SOL	{nexperi × nb × Ncmpt}[Nnodes]	Solution of the PDE (magnetization) . This variable should not be outputted from the solution of the BTPDE if the finite elements mesh is too large.
SIG_cmpts	[Ncmpt × nexperi × nb]	Signal (integral of magnetization) at $TE$ in each compartment.
SIG_allcmpts	[nexperi × nb]	Signal (integral of magnetization) at $TE$ summed over all compartments.
SIG_cmpts_hardi	[ngdir × Ncmpt × nexperi × nb]	Signal in each compartment in each direction.
SIG_allcmpts_hardi	[ngdir × nexperi × nb]	Signal accounting for all compartments in each direction.
ADC_cmpts	[Ncmpt × nexperi ]	ADC in each compartment.
ADC_allcmpts	[nexperi ]	ADC accounting for all compartments.
ADC_cmpts_hardi	[ngdir × Ncmpt × nexperi ]	ADC in each compartment in each direction.
ADC_allcmpts_hardi	[ngdir × nexperi ]	ADC accounting for all compartments in each direction.

Table 5: Some important SpinDoctor output quantities.

#### 4.4 Important functions

In Table 6 we list some important functions of SpinDoctor. For detailed information about them, including argument lists, please read the online documentation.

Function name	Purpose
create_geom	Reads the params_cells input file and creates the geometrical configuration.
create_femesh_fromcells	Creates a finite elements mesh.
read_params_simul_domain	Reads the params_simul_domain input file.
read_params_simul_experi	Reads the params_simul_experi input file.
read_tetgen	Reads the finite elements mesh.
PREPARE_PDE	Set up the PDE model in the geometrical compartments.
GET_VOL_SA	Gets the volume and the surface area quantities from the finite elements mesh.
BTPDE	Computes the BTPDE signal in one diffusion-encoding direction.
HADC	Computes the HADC in one diffusion-encoding direction.
STA	Computes the short time approximation in one diffusion-encoding direction.
ADCFREE	Computes the free diffusion ADC.
FIT_SIGNAL	Fits the $S_0$ and the ADC of the signal.
HARDI_PTS	Provides gradient directions uniformly distributed in unit 3D sphere.
SIG_BTPDE_HARDI	Computes the BTPDE signal in multiple diffusion-encoding directions.
HADC_HARDI	Computes the HADC in multiple diffusion-encoding directions.
PLOT_FEMESH	Displays the finite elements mesh.
PLOT_SIGNAL	Displays the simulated signal in one diffusion-encoding direction.
PLOT_ADC	Displays the simulated ADC in one diffusion-encoding direction.
PLOT_PDESOLUTION	Displays the PDE solution on the finite elements mesh. If the finite elements mesh is too large, the magnetization should not be outputted from the solution of the BTPDE, and this function cannot be used.
PLOT_HARDI_PTS	Displays simulation results in multiple diffusion-encoding directions.

Table 6: Some important functions of SpinDoctor.

## 5 Software components and algorithms

Below we discuss the various components of SpinDoctor in more detail.

### 5.1 Create cells (canonical configuration)

SpinDoctor supports the placement of a group of non-overlapping cells in close vicinity to each other. There are two proposed configurations, one composed of spheres, the other composed of cylinders. The algorithm is described in Algorithm 1.

**Algorithm 1:** Placing  $n_{cell}$  non-overlapping cells.

Generate a large number of possible cell centers.  
Compute the minimum distance,  $dist$ , between the current center and previously accepted cells.  
Find the intersection of  $[dist - dmax \times R_{mean}, dist - dmin \times R_{mean}]$  and  $[R_{min}, R_{max}]$ , where  $R_{mean} = \frac{R_{min} + R_{max}}{2}$ . If the intersection is not empty, then take the middle of the intersection as the new radius and accept the new center. Otherwise, reject the center.  
Loop through the possible centers until get  $n_{cell}$  accepted cells.

### 5.2 Plot cells

SpinDoctor provides a routine to plot the cells to see if the configuration is acceptable (see Fig. 2).

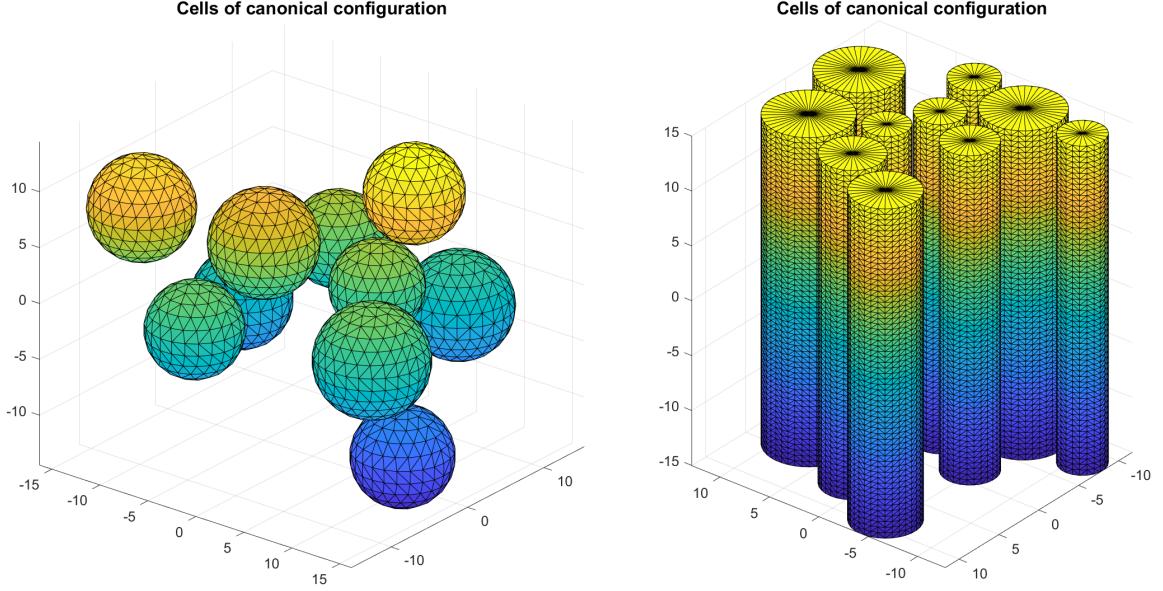


Figure 2: SpinDoctor plots cells in the canonical configuration.

### 5.3 Create surface triangulation

Finite element mesh generation software requires a good surface triangulation. This means the surface triangulation needs to be water-tight and does not self-intersect. How closely these requirements are met in floating point arithmetic has a direct impact on the quality of the finite element mesh generated.

It is often difficult to produce a good surface triangulation for arbitrary geometries. Thus, we restrict the allowed shapes to cylinders and spheres. Below in Algorithms 2 and 3 we describe how to obtain a surface triangulation for spherical cells with nucleus, cylindrical cells with myelin layer, and the ECS (box or tightly wrapped). We describe a canonical configuration where the cylinders are placed parallel to the  $z$ -axis. More general shapes are obtained from the canonical configuration by coordinate transformation in a later step.

**Algorithm 2:** Surface triangulation of spherical cells and ECS.

Suppose we have  $n_{cell}$  spherical cells with nucleus. Denote a sphere with center  $c$  and radius  $R$  by  $S(c, R)$ , we use the built-in functions (convex hull, delaunay triangulation) in MATLAB to get its surface triangulation,  $T(c, R)$ . Call the radii of the nucleus  $r_1, \dots, r_{ncell}$  and the radii of the cells  $R_1, \dots, R_{ncell}$ . Then the boundaries between the cytoplasm and the nucleus are

$$\{\Gamma_i = T(c_i, r_i)\}, i = 1, \dots, n_{cell};$$

and between the cytoplasm and the ECS

$$\{\Sigma_i = T(c_i, R_i)\}, i = 1, \dots, n_{cell};$$

For the box ECS, we find the coordinate limits of the set

$$\bigcup_i S(c_i, R_i) \in [x_0, x_f] \times [y_0, y_f] \times [z_0, z_f]$$

and add a gap  $k = \text{ECS\_gap} \times \max\{x_f - x_0, y_f - y_0, z_f - z_0\}$  to make a box

$$B = [x_0 - k, x_f + k] \times [y_0 - k, y_f + k] \times [z_0 - k, z_f + k].$$

We put 2 triangles on each face of  $B$  to make a surface triangulation  $\Psi$  with 12 triangles. For the tight-wrap ECS, we increase the cell radius by a gap size and take the union

$$W = \bigcup_i S(c_i, R_i + \text{ECS\_gap} \times R_{mean}),$$

where  $R_{mean} = \frac{R_{min}+R_{max}}{2}$ . We use the alphaShape function in MATLAB to find a surface triangulation  $\Psi$  that contains  $W$ .

**Algorithm 3:** Surface triangulation of cylindrical cells and ECS.

Suppose we have  $n_{cell}$  cylindrical cells with a myelin layer, all with height  $H$ . Denote a disk with center  $c$  and radius  $R$  by  $D(c, R)$ , and the circle with the same center and radius by  $C(c, R)$ .

Let the radii of the axons be  $r_1, \dots, r_{n_{cell}}$  and the radii of the cells be  $R_1, \dots, R_{n_{cell}}$ , meaning the thickness of the myelin layer is  $R_i - r_i$ .

The boundary between the axon and the myelin layer is:

$$C(c_i, r_i) \times [-H/2, H/2]$$

We discretize  $C(c_i, r_i)$  as a polygon  $P(c_i, r_i)$  and place one at  $z = -H/2$  and one at  $z = H/2$ . Then we connect the corresponding vertices of  $P(c_i, r_i) \times \{-H/2\}$  and  $P(c_i, r_i) \times \{H/2\}$  and add a diagonal on each panel to get a surface triangulation  $\Gamma_i$ .

Between the myelin layer and the ECS we discretize  $C(c_i, R_i)$  as a polygon and place one at  $z = -H/2$  and one at  $z = H/2$  to get a surface triangulation  $\Sigma_i$ .

For the box ECS, we find the coordinate limits of the union of  $D(c_i, R_i)$  and add a gap to make a rectangle in two dimensions. Then we place the rectangle at  $z = -H/2$  and at  $z = H/2$  to get a box. Finally, the box is given a surface triangulation with 12 triangles.

For tight-wrap ECS, we increase the cell radius by a gap size and take the union

$$W = \bigcup_i D(c_i, R_i + kR_{mean}).$$

We use the alphaShape function in MATLAB to find a two dimensional polygon  $Q$  that contains  $W$ . We place  $Q$  at  $z = -H/2$  and at  $z = H/2$  and connect correponding vertices, adding a diagonal on each panel. Suppose  $Q$  is a polygon with  $n$  vertices, then the surface triangulation of the side of the ECS will have  $2n$  triangles.

The above procedure produces a surface triangulation for the boundaries that are parallel to  $z$ -axis. We now must close the top and bottom. The top and bottom boundaries is just the interior of  $Q$ . However, the surface triangulation cannot be done on  $Q$  directly. We must cut out  $D(c_i, r_i)$ , the disk which touches the axon, and  $A_i = D(c_i, R_i) - D(c_i, r_i)$ , the annulus which touches the myelin. Then we triangulate  $Q - \bigcup_i D_i - \bigcup_i A_i$  using the MATLAB built-in function that triangulates a polygon with holes to get the boundary that touches the ECS. The surface triangulation for  $A_i$  and  $D(c_i, r_i)$  are straightforward.

## 5.4 Plot surface triangulation

SpinDoctor provides a routine to plot the surface triangulation (see Fig. 3).

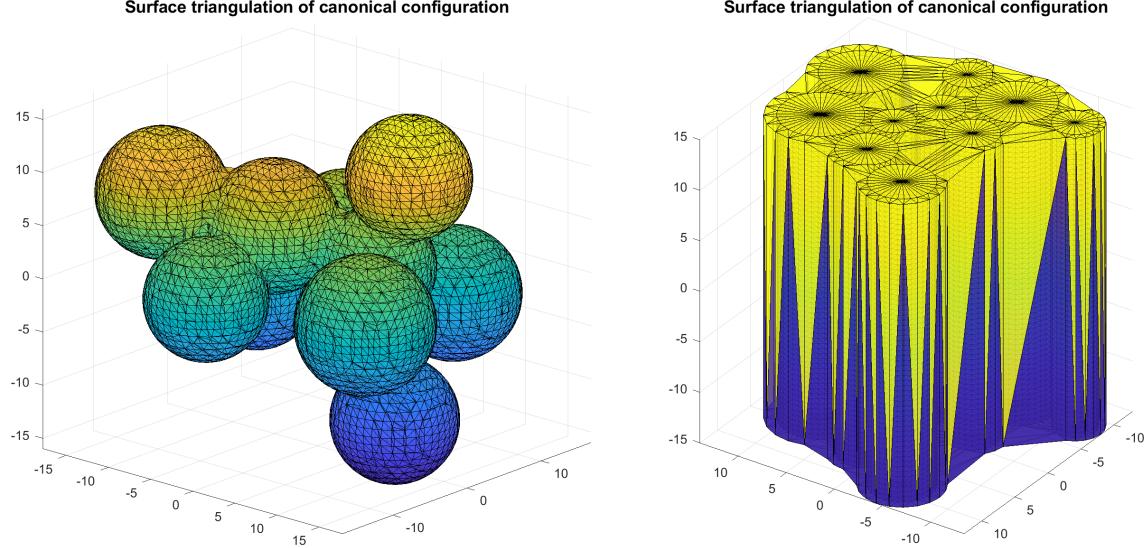


Figure 3: SpinDoctor plots the surface triangulation of the canonical configuration. Left: spherical cells with ECS; Right: cylindrical cells with ECS.

## 5.5 Finite element mesh generation

SpinDoctor calls Tetgen [2], an external package (executable files are included in the toolbox package), to create a tetrahedra finite elements mesh from the surface triangulation generated by Algorithms 2 and 3. The FE mesh is generated on the canonical configuration. The numbering of the compartments and boundaries used by SpinDoctor are given in Tables 7 and 8. The labels are related to the values of the intrinsic diffusion coefficient, the initial spin density, and the permeability requested by the user. Then the FE mesh nodes are deformed analytically by a coordinate transformation, described in Algorithm 4.

Spherical cells without nucleus			
Cmpt	Cytoplasm	Nucleus	ECS
Label	OUT		ECS
Number			
	$[1 : n_{cell}]$		$n_{cell} + 1$

Spherical cells with nucleus			
Cmpt	Cytoplasm	Nucleus	ECS
Label	OUT	IN	ECS
Number	$[1 : n_{cell}]$	$[n_{cell} + 1 : 2n_{cell}]$	$2n_{cell} + 1$

Cylindrical cells without myelin			
Cmpt	Axon	Myelin	ECS
Label	OUT		ECS
Number	$[1 : n_{cell}]$		$n_{cell} + 1$

Cylindrical cells with myelin			
Cmpt	Axon	Myelin	ECS
Label	IN	OUT	ECS
Number	$[1 : n_{cell}]$	$[n_{cell} + 1 : 2n_{cell}]$	$2n_{cell} + 1$

Table 7: The labels and numbers of compartments.

Spherical cells without nucleus			
Boundary	Sphere		Outer ECS boundary
Label	OUT_ECS		$\kappa = 0$
Number	$1 : n_{cell}$		$n_{cell} + 1$
Spherical cells with nucleus			
Boundary	Outer sphere	Inner sphere	Outer ECS boundary
Label	OUT_ECS	IN_OUT	$\kappa = 0$
Number	$1 : n_{cell}$	$n_{cell} + 1 : 2n_{cell}$	$2n_{cell} + 1$
Cylindrical cells without myelin			
Boundary	Cylinder side wall	Cylinder top and bottom	Outer ECS boundary minus cylinder top/bottom
Label	OUT_ECS	$\kappa = 0$	$\kappa = 0$
Number	$2[1 : n_{cell}] - 1$	$2[1 : n_{cell}]$	$2n_{cell} + 1$
Cylindrical cells with myelin			
Boundary	Inner cylinder side wall	Inner cylinder top and bottom	
Label	IN_OUT	$\kappa = 0$	
Number	$4[1 : n_{cell}] - 3$	$4[1 : n_{cell}] - 2$	
	Outer cylinder side wall	Outer cylinder top and bottom	Outer ECS boundary minus cylinder top/bottom
Label	OUT_ECS	$\kappa = 0$	$\kappa = 0$
Number	$4[1 : n_{cell}] - 1$	$4[1 : n_{cell}]$	$4n_{cell} + 1$

Table 8: The labels and numbers of boundaries.

**Algorithm 4:** Bending and twisting of the FE mesh of the canonical configuration.

The external package Tetgen [2] generates the finite element mesh that keeps track of the different compartments and the interfaces between them. The mesh is saved in several text files. The connectivity matrices of the finite elements and facets are not modified by the coordinates transformation described below. The nodes are transformed by bending and twisting as described next.

The set of FE mesh nodes  $\{x_i, y_i, z_i\}$  are transformed in the following ways:

Twisting around the  $z$ -axis with a user-chosen twisting parameter  $\alpha_{twist}$  is defined by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} \cos(\alpha_{twist}z) & -\sin(\alpha_{twist}z) & 0 \\ \sin(\alpha_{twist}z) & \cos(\alpha_{twist}z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}.$$

Bending on the  $x - z$  plane with a user-chosen bending parameter  $\alpha_{bend}$  is defined by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} x + \alpha_{bend}z^2 \\ y \\ z \end{bmatrix}.$$

Given  $[\alpha_{bend}, \alpha_{twist}]$ , bending is performed after twisting.

## 5.6 Plot FE mesh

SpinDoctor provides a routine to plot the FE mesh (see Fig. 4 for cylinders and ECS that have been bent and twisted).

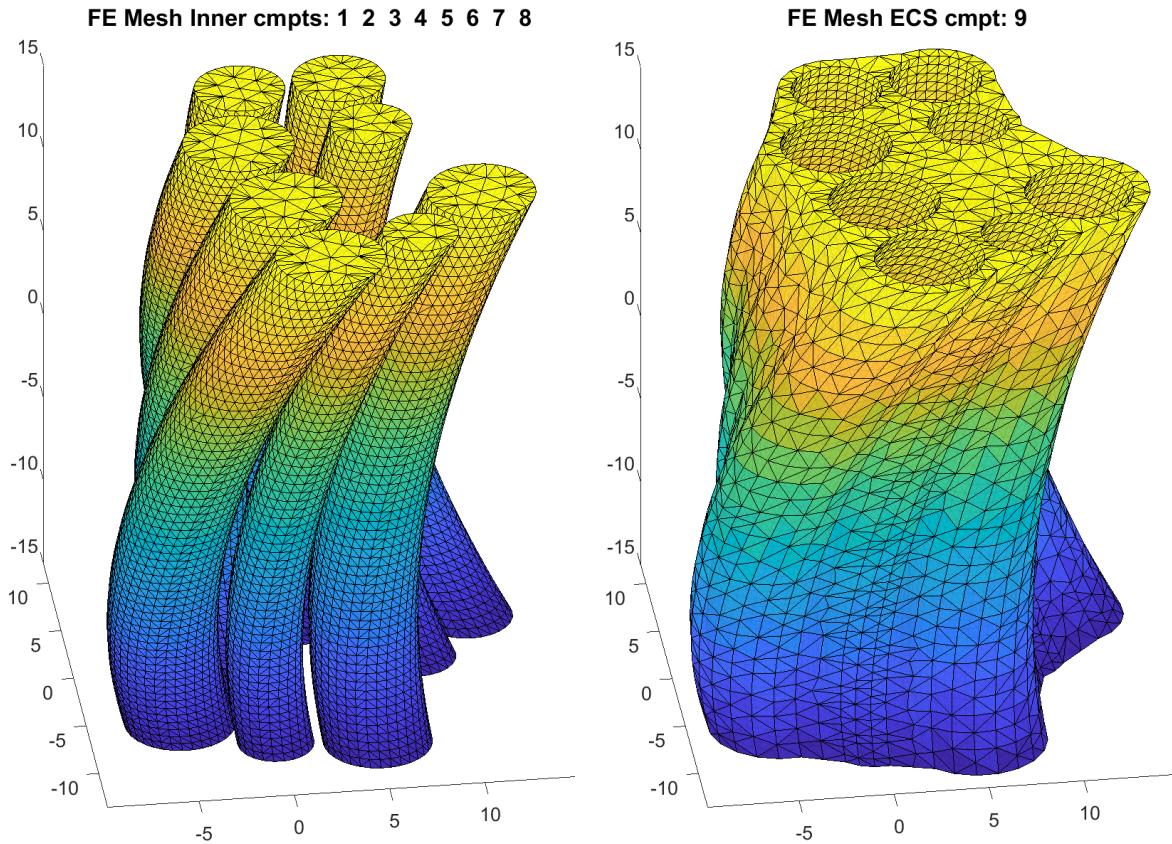


Figure 4: FE mesh of cylinders and ECS after bending and twisting. Compartment number is 1 to 8 for the cylinders and 9 for the ECS.

## 5.7 BTPDE

The spatial discretization of the Bloch-Torrey PDE is based on a finite element method where *interface (ghost) elements* [12] are used to impose the permeable interface conditions. The time stepping is done using the MATLAB built-in ODE routine `ode23t`. See Algorithm 5.

**Algorithm 5:** BTPDE.

FE matrices are generated for each compartment by the finite element method with continuous piecewise linear basis functions (known as  $P_1$ ). The basis functions are denoted as  $\varphi_k$  for  $k = 1, \dots, N_v$ , where  $N_v$  denotes the number of mesh nodes (vertices). All matrices are sparse matrices.  $\mathbf{M}$  and  $\mathbf{S}$  are known in the FEM literature as mass and stiffness matrices which are defined as follows:

$$\mathbf{M}_{ij} = \int_{\Omega} \varphi_i \varphi_j d\mathbf{x}, \quad \mathbf{S}_{ij} = \int_{\Omega} \sigma_i \nabla \varphi_i \cdot \nabla \varphi_j d\mathbf{x}.$$

$\mathbf{J}$  has a similar form as the mass matrix but it is scaled with the coefficient  $\mathbf{g} \cdot \mathbf{x}$ , we therefore call it the scaled-mass matrix

$$\mathbf{J}_{ij} = \int_{\Omega} \mathbf{g} \cdot \mathbf{x} \varphi_i \varphi_j d\mathbf{x}.$$

We construct the matrix based on the flux matrix  $\mathbf{Q}$

$$\mathbf{Q}_{ij} = \int_{\partial\Omega} w \varphi_i \varphi_j ds$$

where a scalar function  $w$  is used as an interface marker. The matrices are assembled from local element matrices and the assembly process is based on vectorized routines of [3], which replace expensive loops over elements by operations with 3-dimensional arrays. All local elements matrices in the assembly of  $\mathbf{S}, \mathbf{M}, \mathbf{J}$  are evaluated at once and stored in a full matrix of size  $4 \times 4 \times N_e$ , where  $N_e$  denotes the number of tetrahedral elements. The assembly of  $\mathbf{Q}$  is even simpler; all local matrices are stored in a full matrix of size  $3 \times 3 \times n_{be}$ , where  $n_{be}$  denotes the number of boundary triangles.

Double nodes are placed at the interfaces between compartments connected by permeable membrane.  $\bar{\mathbf{Q}}$  is used to impose the interface conditions and it is associated with the *interface (ghost) elements*. Specifically, assume that the double nodes are defined in a pair of indices  $\{i, \bar{i}\}$ ,  $\bar{\mathbf{Q}}$  is defined as the following

$$\bar{\mathbf{Q}}_{ij} = \begin{cases} \mathbf{Q}_{ij}, & \text{if vertex } i \text{ and } j \text{ belong to one interface} \\ -\mathbf{Q}_{\bar{i}\bar{j}}, & \text{if vertex } i \text{ and } j \text{ belong to two different interfaces} \end{cases}$$

The fully coupled linear system has the following form

$$\mathbf{M} \frac{\partial \xi}{\partial t} = - \left( I \gamma f(t) \mathbf{J} + \mathbf{S} + \bar{\mathbf{Q}} \right) \xi \quad (14)$$

where  $\xi$  is the approximation of the magnetization  $M$ . SpinDoctor calls MATLAB built-in ODE routine ode23t to solve the semi-discretized system of equations.

## 5.8 HADC model

Similarly, the diffusion equation of the HADC model is discretized by finite elements. See Algorithm 6.

**Algorithm 6:** HADC model.

Eq. (12) can be discretized similarly as described for the Bloch-Torrey PDE and has the matrix form

$$\mathbf{M} \frac{\partial \zeta}{\partial t} = -\mathbf{S} \zeta + \mathbf{Q} \bar{\zeta} \quad (15)$$

where  $\zeta$  is the approximation of  $w$  and  $\bar{\zeta}_i = \sigma_i F(t) \mathbf{u}_g \cdot \mathbf{n}(\mathbf{x}_i)$ . We note that the matrices here are assembled and solved separately for each compartment. SpinDoctor calls MATLAB built-in ODE routine ode23t to solve the semi-discretized equation.

## 5.9 Visualization of results

To display results in one diffusion-encoding direction, the user can choose the following options:

1. the signal is plotted against the b-values;
2. the ADC is plotted against the geometrical compartment number.

We note that if the ADC from the short time approximation is negative (meaning the short diffusion time assumption is not satisfied), we do not display the negative ADC in the plot.

For results in multiple diffusion-encoding directions uniformly distributed on the three dimensional unit sphere, the user can choose the following options:

1. the signals are plotted as a surface plot. The dots are the end points of the vectors in the diffusion-encoding direction multiplied the signal. The color gives the magnitude of the signal;
2. the ADCs are plotted as a surface plot. The dots are the end points of the vectors in the diffusion-encoding direction multiplied the ADC. The color gives the magnitude of the ADC.

The user can also choose to plot the magnetization solution at the surface nodes of the finite elements mesh.

## 6 SpinDoctor examples

In this section we show some prototypical examples using the available functionalities of SpinDoctor.

### 6.1 Comparison of BTPDE and HADC with Short Time Approximation

In Fig. 5 we show that both BTPDE and HADC solutions match the STA values at short diffusion times for cylindrical cells (compartments 1 to 5). We also show that for the ECS (compartment 6), the STA is too low, because it does not account for the fact that spins in the ECS can diffuse around several cylinders. This also shows that when the interfaces are impermeable, the BTPDE ADC and that from the HADC model are identical. The diffusion-encoding sequence here is cosine OGSE with 6 periods.

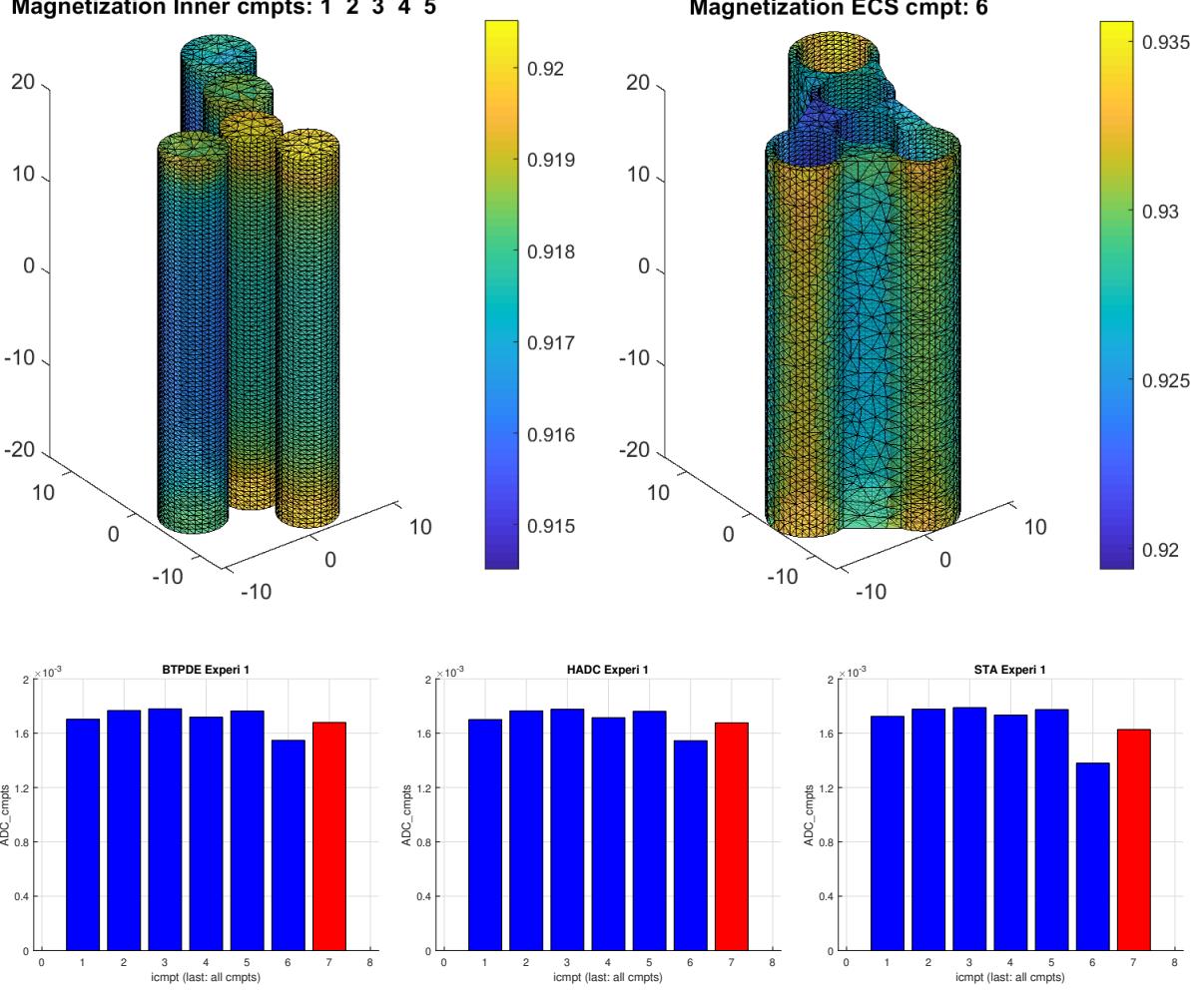


Figure 5: Geometry: 5 cylinders, tight wrap ECS, ECS gap = 0.2,  $\mathbf{u}_g = [1, 1, 1]$ ,  $\sigma^{out} = \sigma^{ecs} = 2 \times 10^{-3} \text{ mm}^2/\text{s}$ ,  $\kappa = 0 \text{ m/s}$ , OGSE cosine ( $\delta = 14\text{ms}$ ,  $\Delta = 14\text{ms}$ , number of periods = 6). The vertical bars indicate the ADC in each compartment. The ADC in the rightmost position is the ADC that takes into account the diffusion in all the compartments.

## 6.2 Permeable membranes

In Fig. 6 we show the effect of permeability: the Bloch-Torrey PDE model includes permeable membranes ( $\kappa = 1 \times 10^{-3} \text{ m/s}$ ) whereas the HADC has impermeable membranes. We see in the permeable case, the ADC in the spheres are higher than in the impermeable case, whereas the ECS show reduced ADC because the faster diffusing spins in the ECS are allowed to move into the slowly diffusing spherical cells. We note that in the permeable case, the ADC in each compartment is obtained by using the fitting formula involving the logarithm of the dMRI signal, and we defined the "signal" in a compartment as the total magnetization in that compartment at TE, which is just the integral of the solution of the Bloch-Torrey PDE in that compartment.

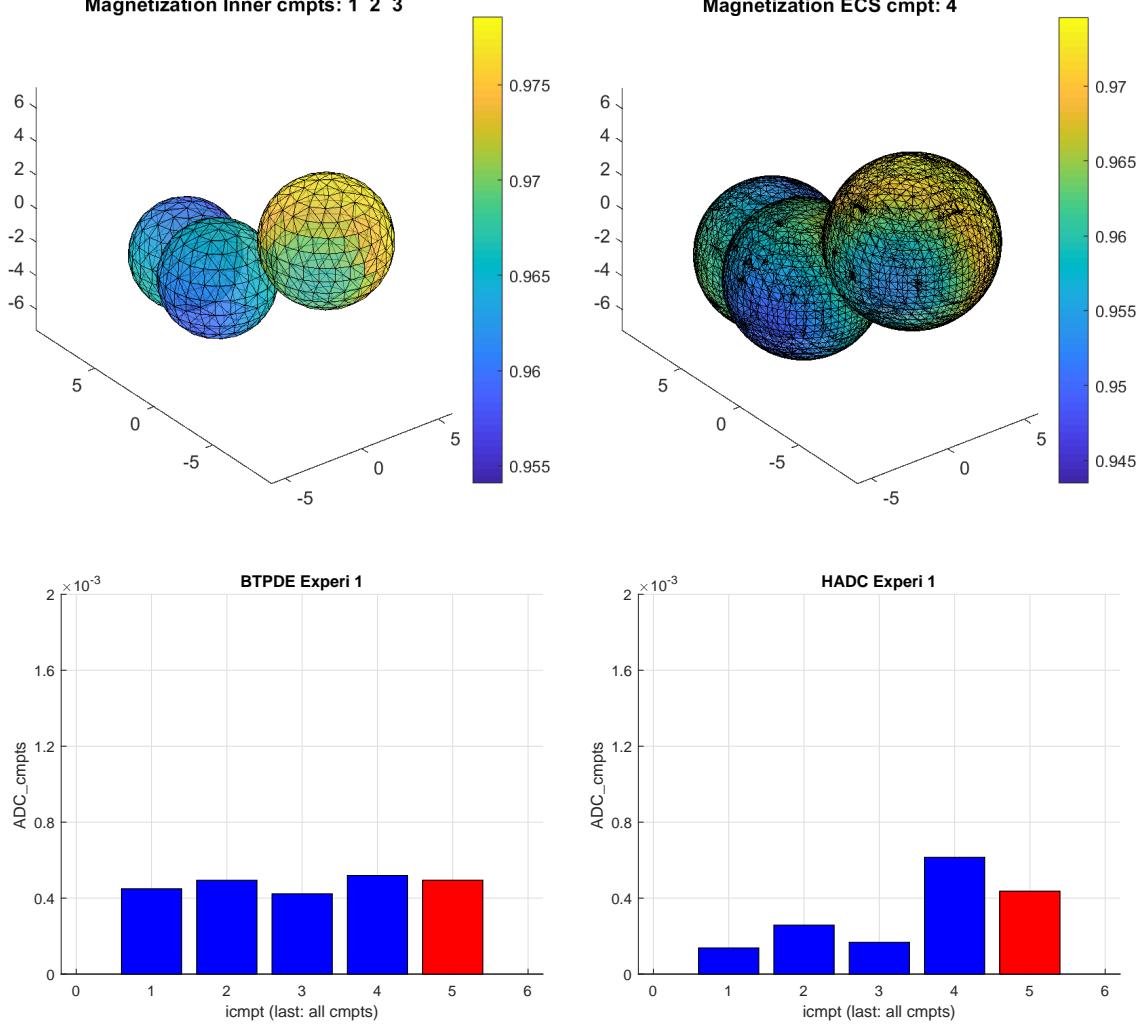


Figure 6: Geometry: 3 spheres, tight wrap ECS, ECS gap = 0.3,  $\mathbf{u}_g = [1, 1, 0]$ ,  $\sigma^{in} = \sigma^{ecs} = 2 \times 10^{-3} \text{ mm}^2/\text{s}$ ,  $\kappa = 1 \times 10^{-3} \text{ m/s}$  (left),  $\kappa = 0 \text{ m/s}$  (right). PGSE ( $\delta = 5\text{ms}$ ,  $\Delta = 5\text{ms}$ ). The vertical bars indicate the ADC in each compartment. The ADC in the rightmost position is the ADC that takes into account the diffusion in all the compartments.

### 6.3 Myelin layer

In Fig. 7 we show the diffusion in cylindrical cells, the myelin layer, and the ECS. The ADC is higher in the myelin layer than in the cells, because for spins in the myelin layer diffusion occurs in the tangential direction (around the circle). At longer diffusion times, the ADC of both the myelin layer and the cells becomes very low. The ADC is the highest in the ECS, because the diffusion distance can be longer than the diameter of a cell, since the diffusing spins can move around multiple cells.

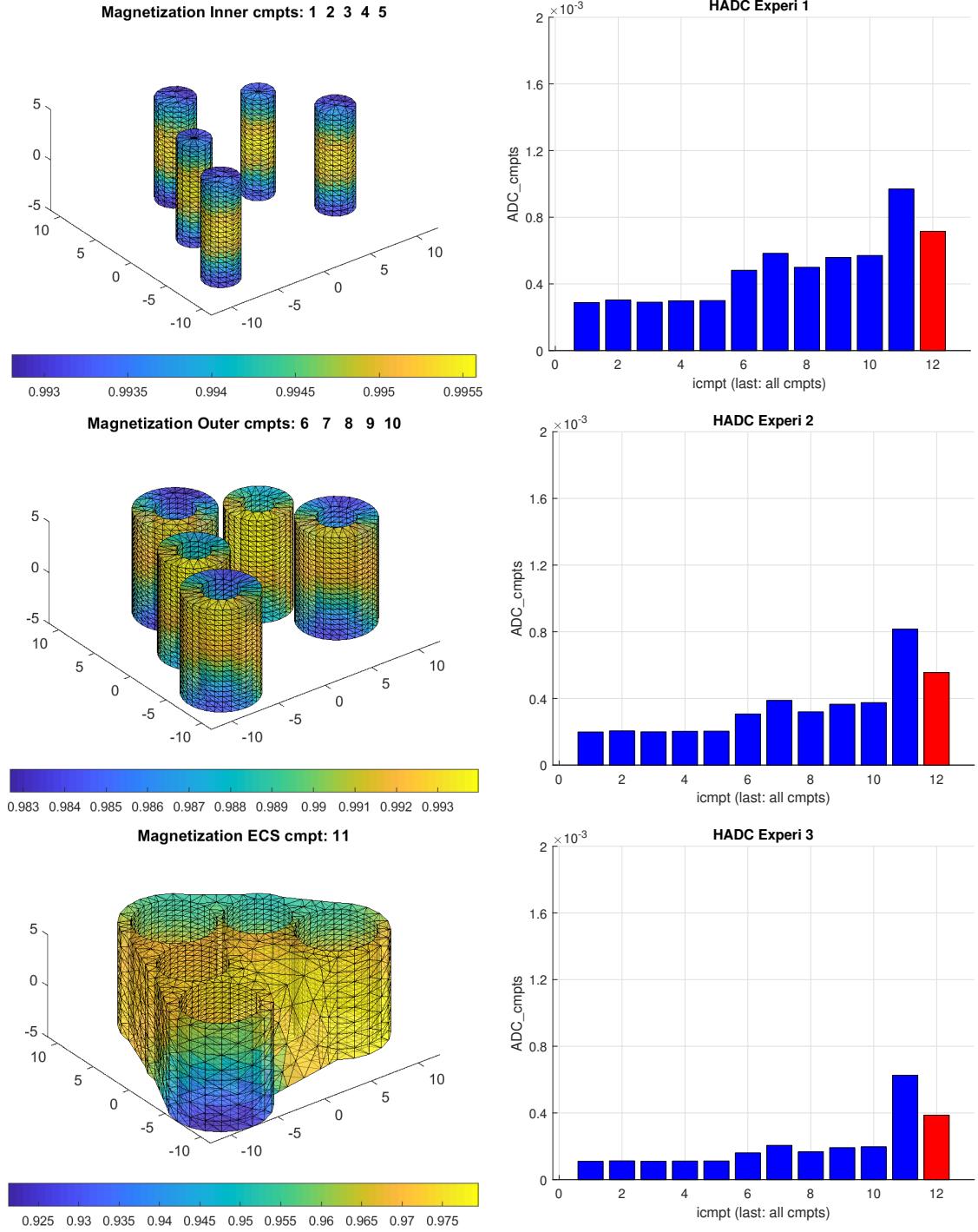


Figure 7: Geometry: 5 cylinders, myelin layer,  $R_{in}/R_{out} = 0.5$ , tight wrap ECS, ECS gap = 0.3,  $\kappa = 0 \text{ m/s}$ ,  $\mathbf{u}_g = [1, 1, 1]$ ,  $\sigma^{in} = \sigma^{out} = \sigma^{ecs} = 2 \times 10^{-3} \text{ mm}^2/\text{s}$ , 3 experiments: PGSE ( $\delta = 5\text{ms}$ ,  $\Delta = 5, 10, 20\text{ms}$ ). Left: the magnetization at  $\Delta = 5\text{ms}$ . Right: the ADC values. The vertical bars indicate the ADC in each compartment. The ADC in the rightmost position is the ADC that takes into account the diffusion in all the compartments.

## 6.4 Twisting and bending

In Fig. 8 we show the effect of bending and twisting in cylindrical cells in multiple gradient directions. The HADC is obtained in 20 directions uniformly distributed in the sphere. We used spherical harmonics interpolation to interpolate the HADC in the entire sphere. Then we deformed the radius of the unit sphere to be proportional to the interpolated HADC and plotted the 3D shape. The color axis also

indicates the value of the interpolated HADC.

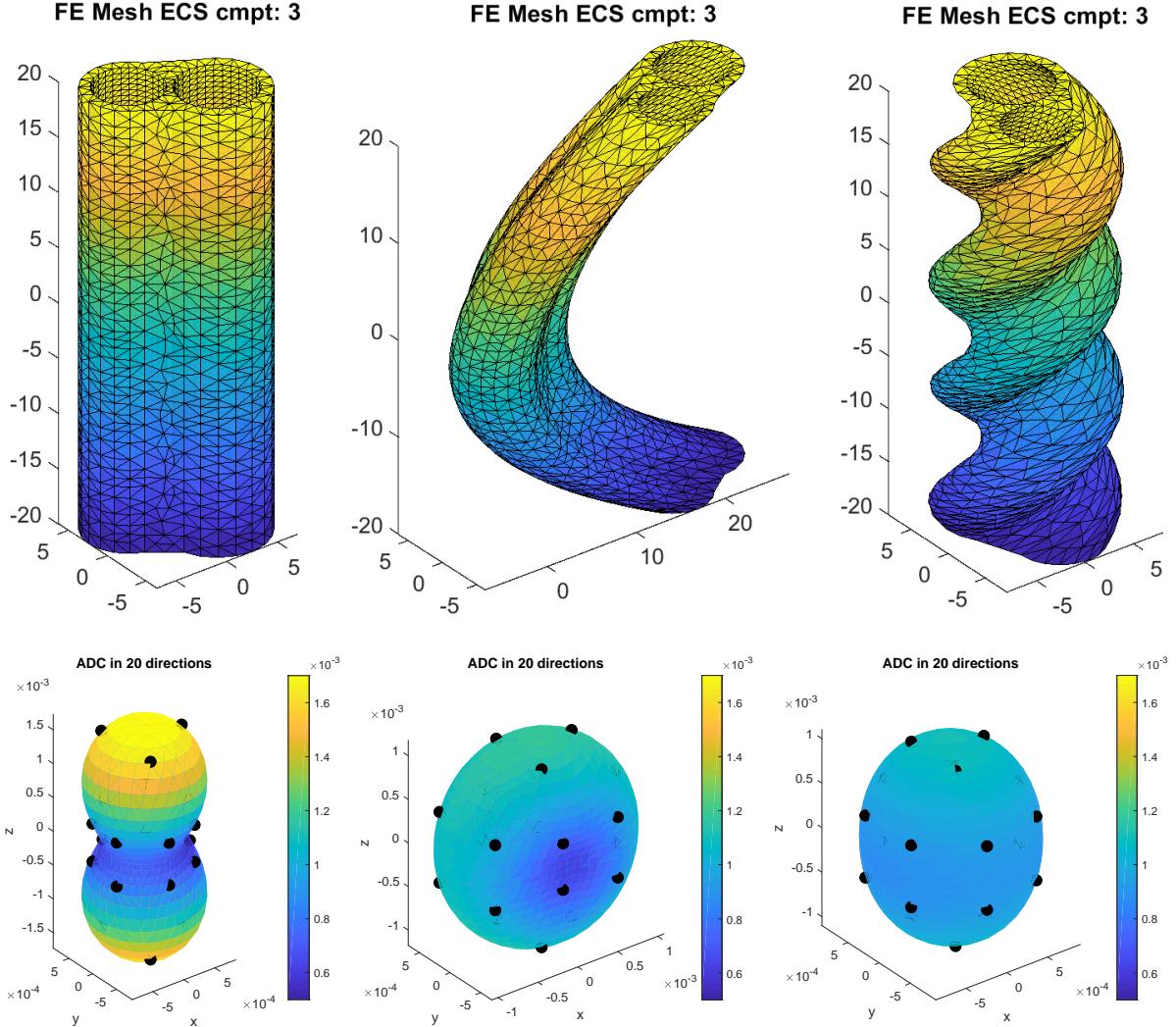


Figure 8: Geometry: 2 cylinders, no myelin layer, tight wrap ECS, ECS gap = 0.3,  $\kappa = 0 \text{ m/s}$ ,  $\sigma^{out} = \sigma^{ecs} = 2 \times 10^{-3} \text{ mm}^2/\text{s}$ , PGSE ( $\delta = 2.5\text{ms}$ ,  $\Delta = 5\text{ms}$ ).

Left: canonical configuration. Middle: bend parameter = 0.05. Right: twist parameter = 0.30. Top: FE mesh of the ECS (the FE mesh of the axon compartments numbered 1 and 2 not shown). Bottom: interpolated values of the HADC on the unit sphere, and then the sphere was distorted to reflect the value of the HADC. The color axis also gives the value of the HADC in the various gradient directions. The black dots indicate the 20 original gradient-directions in which the HADC was simulated. The spherical harmonics interpolation takes the 20 original directions into 900 directions uniformly distributed on the sphere.

## 7 Neuron Module

The diffusion MRI signal arising from neurons can be numerically simulated by solving the Bloch-Torrey partial differential equation. In order to facilitate the diffusion MRI simulation of realistic neurons by the research community, we constructed finite element meshes for a group of 36 pyramidal neurons and a group of 29 spindle neurons whose morphological descriptions were found in the publicly available neuron repository *NeuroMorpho.Org*. These finite elements meshes range from having 15163 nodes to 622553 nodes. We also broke the neurons into the soma and dendrite branches and created finite elements meshes for these cell components. Through the Neuron Module, these neuron and components finite element meshes can be seamlessly coupled with the functionalities of SpinDoctor to provide the diffusion MRI signal attributable to spins inside neurons.

## 7.1 Work flow

The Neuron Module follows the same workflow as SpinDoctor and builds upon the functionalities of SpinDoctor. The Neuron Module uses the same set of three input files as SpinDoctor. SpinDoctor allows the easy construction of multiple compartment models of the brain white matter, with the possibility of coupling water diffusion between the geometrical compartments by permeable membranes. As this time, we have not implemented the Neuron Module for coupled compartments linked by permeable membranes. Rather, the diffusion MRI signal is computed with zero permeability on the compartment boundaries. The current emphasis of the Neuron Module is to show how the geometrical structure of neurons affect the diffusion MRI signal. Thus, some of the input parameters related to multiple compartment models in SpinDoctor are not applicable in the current version of the Neuron Module. However, we have kept the exactly same input file formats as SpinDoctor in anticipation of the future developement of the Neuron Module for permeable membranes. In particular, the various compartments in SpinDoctor have designations as IN, OUT, and ECS, and in the Neuron Module, the geometry defined by the finite element mesh is designated as the OUT compartment.

## 7.2 User provided input files

In SpinDoctor, there are three input files in which the user specifies the parameters of the desired simulations. They are :

1. *params\_cells.in*: contains the cells parameters
2. *params\_simul\_domain.in*: contains the simulation domain parameters
3. *params\_simul\_experi.in*: contains the simulation experiment parameters

We list the input files the user must provide in order to use the Neuron Module, noting where relevant, the input parameters that are not applicable (marked by "na") to the current version of the Neuron Module.

### 7.2.1 Read cells parameters

The user provides an input file for the cells parameters, in the format described in Table 9. To simulate the diffusion MRI signal of a neuron, the user chooses option 3 for the cell shape. The user specifies the name of the neuron to be simulated in line 2.

Line	Variable name	Example	Explanation
1	cell_shape	3	1 = spheres; 2 = cylinders; 3 = neuron;
2	fname	'msh_files/pyramidal/ 02b_pyramidal1aACC'	file name of neuron mesh
3	ncell	1	number of cells
4	Rmin	na	min Radius
5	Rmax	na	max Radius
6	dmin	na	min (%) distance between cells
7	dmax	na	max (%) distance between cells
8	para_deform	na	$[\alpha \ \beta]$ ; $\alpha$ defines the amount of bend; $\beta$ defines the amount of twist
9	Hcyl	na	height of cylinders

Table 9: Input file containing cells parameters. "na" means not applicable to neuron simulation.

### 7.2.2 Read simulation domain parameters

The user provides an input file for the simulation domain parameters, in the format described in Table 10.

Line	Variable name	Example	Explanation
1	Rratio	na	
2	include_ECS	na	
3	ECS_gap	na	
4	dcoeff_IN	na	
5	dcoeff_OUT	0.002	diffusion coefficient in OUT cmpt
6	dcoeff_ECS	na	
7	ic_IN	na	
8	ic_OUT	1	initial spin density in OUT cmpt
9	ic_ECS	na	
10	kappa_IN_OUT	na	
11	kappa_OUT_ECS	na	
12	Htetgen	-1	Requested tetgen mesh size; -1 = Use tetgen default;
13	tetgen_cmd	'SRC/TETGEN/ tetGen/win64/ tetgen'	path to tetgen_cmd

Table 10: Input file of simulation domain parameters. "na" means not applicable to neuron simulation.

### 7.2.3 Read simulation experiment parameters

The user provides an input file for the simulation experiment parameters, in the format described in Table 11.

Line	Variable name	Example	Explanation
1	ngdir	300	number of gradient direction; if $ngdir > 1$ , the gradient directions are distributed uniformly on a sphere; if $ngdir = 1$ , take the gradient direction from the line below;
2	gdir	1.0 0.0 0.0	gradient direction; No need to normalize;
3	nexperi	1	number of sequences to simulate;
4	sdeltavec	2500 10000	small delta;
5	bdeltavec	5000 43000	big delta;
6	seqvec	1 1 1	diffusion sequence of experiment; 1 = PGSE; 2 = OGSEsin; 3 = OGSEcos;
7	npervec	0 0 0	number of period of OGSE;
8	solve_hadc	na	0 = do not solve HADC; Otherwise solve HADC;
9	rtol_deff, atol_deff	na	[ $r_{tol}$ $a_{tol}$ ]; relative and absolute tolerance for HADC ODE solver;
10	solve_btpde	1	0: do not solve BTPDE; Otherwise solve BTPDE;
11	rtol_bt, atol_bt	1e-2 1e-4	[ $r_{tol}$ $a_{tol}$ ]; relative and absolute tolerance for BTPDE ODE solver;
12	nb	4	number of b-values;
13	blimit	0	0=specify bvec; 1=specify[bmin,bmax]; 2 = specify[gmin,gmax];
14	const_q	0	0: use input bvalues for all experiments; 1: take input bvalues for the first experiment and use the same q for the remaining experiments
15	b-values	0 1000 2000 3000	b-values or [bmin, bmax] or [gmin, gmax]; depending on line 13;

Table 11: Input file for simulation experiment parameters.

### 7.3 Neuron Module examples

Below we display figures from the functions PLOT\_FEMESH, PLOT\_PDESOLUTION, PLOT\_SIGNAL, PLOT\_HARDI PTS. The geometrical configuration is the spindle neuron *03b\_spindle6aFI*. The intrinsic diffusion coefficient is set to  $D_0 = 2 \times 10^{-3} \text{ mm}^2/\text{s}$ , with nonpermeable membrane. We simulated 2 diffusion-encoding sequences:

Experiment 1:  $f_1$  is PGSE ( $\delta = 10.6\text{ms}$ ,  $\Delta = 13\text{ms}$ ),

Experiment 2:  $f_2$  is PGSE ( $\delta = 10.6\text{ms}$ ,  $\Delta = 73\text{ms}$ ).

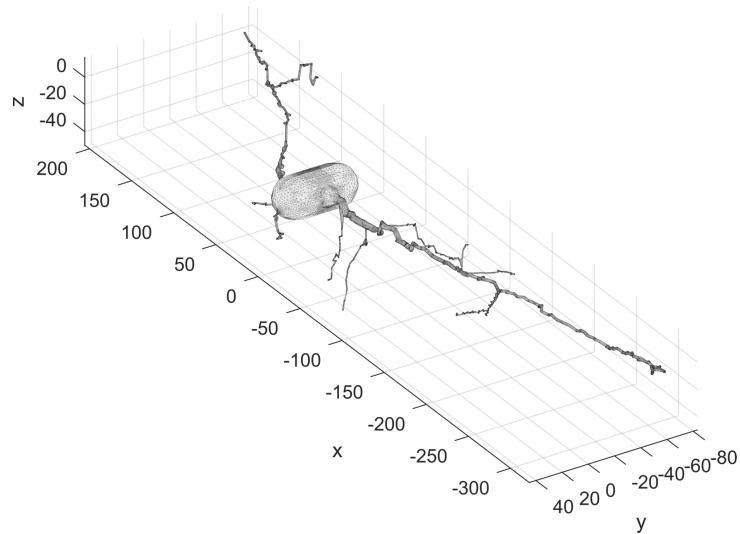


Figure 9: The finite elements mesh of the neuron *03b\_spindle6aFI*. The unit is  $\mu\text{m}$ . (Using the command PLOT\_FEMESH)

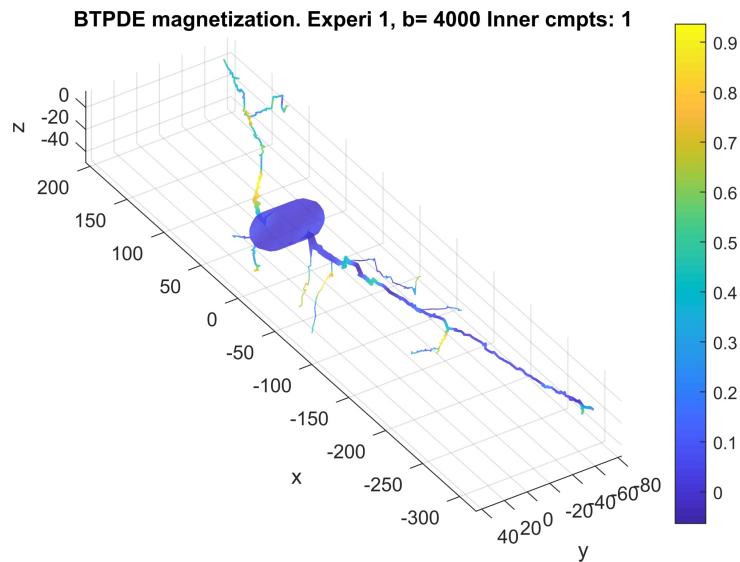


Figure 10: The PDE solution (magnetization) on the neuron *03b\_spindle6aFI* in the diffusion-encoding direction  $(1, 1, 1)$ . The diffusion-encoding sequence is PGSE ( $\delta = 10.6\text{ms}$ ,  $\Delta = 13\text{ms}$ ). The b-value is  $b = 4000 \text{s/mm}^2$ . (Using the command PLOT\_PDESOLUTION)

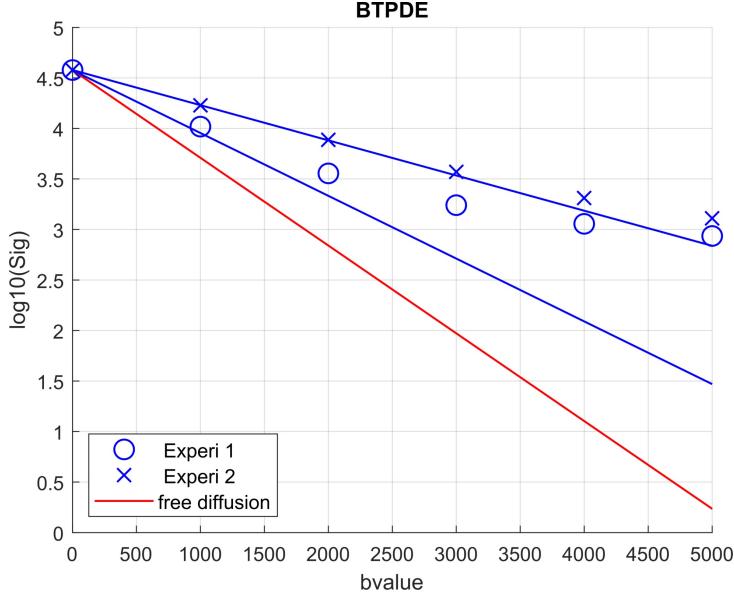


Figure 11: The (non-normalized) simulated signal of the neuron *03b\_spindle6aFI* in the diffusion-encoding direction (1, 1, 1). For Experi 1, the diffusion-encoding sequence is PGSE ( $\delta = 10.6\text{ms}$ ,  $\Delta = 13\text{ms}$ ). For Experi 2, the diffusion-encoding sequence is PGSE ( $\delta = 10.6\text{ms}$ ,  $\Delta = 73\text{ms}$ ). (Using the command PLOT\_SIGNAL)

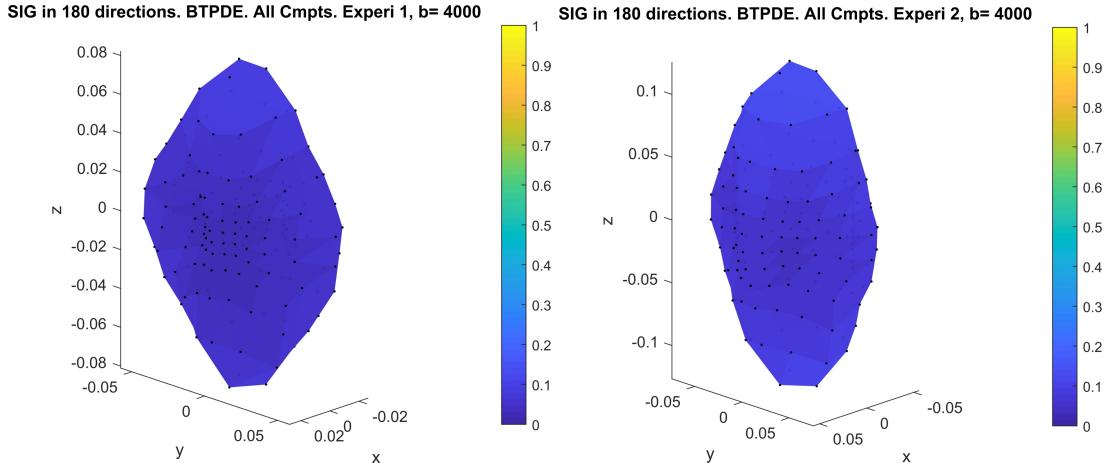


Figure 12: The simulated signal in 180 diffusion-encoding directions for the neuron *03b\_spindle6aFI*. Left: The signal in 180 directions with PGSE ( $\delta = 10.6\text{ms}$ ,  $\Delta = 13\text{ms}$ ) and  $b=4000 \text{ s/mm}^2$ . Right: The signal in 180 directions with PGSE ( $\delta = 10.6\text{ms}$ ,  $\Delta = 73\text{ms}$ ) and  $b=4000 \text{ s/mm}^2$ . (Using the command PLOT\_HARDI PTS)

## 7.4 Finite elements meshes of neurons

Table 12 lists the names and the finite element mesh sizes of the group of 36 pyramidal neurons and the group of 29 spindle neurons. Table 13 shows the morphological characteristics for the neurons. The neuron models and the measurement data are from [13] and [14].

Neuron ID	Num of FE mesh nodes	Neuron ID	Num of FE mesh nodes
<i>03a_spindle2aFI</i>	38202	<i>02a_pyramidal2aFI</i>	119156
<i>03a_spindle6aFI</i>	44000	<i>02b_pyramidal1aACC</i>	45216
<i>03b_spindle4aACC</i>	17370	<i>02b_pyramidal1aFI</i>	105384
<i>03b_spindle5aACC</i>	26345	<i>03a_pyramidal9aFI</i>	81530
<i>03b_spindle6aACC</i>	26792	<i>03b_pyramidal2aACC</i>	28183
<i>03b_spindle7aACC</i>	21618	<i>03b_pyramidal3aACC</i>	27607
<i>04b_spindle3aFI</i>	51265	<i>03b_pyramidal3aFI</i>	151362
<i>05b_spindle5aFI</i>	22457	<i>03b_pyramidal4aFI</i>	96177
<i>06b_spindle8aACC</i>	15163	<i>03b_pyramidal9aFI</i>	66162
<i>07b_spindle9aACC</i>	54952	<i>04a_pyramidal4aACC</i>	150897
<i>08a_spindle13aACC</i>	46293	<i>04a_pyramidal5aACC</i>	89256
<i>09o_spindle7aFI</i>	38992	<i>04b_pyramidal5aFI</i>	95784
<i>09o_spindle8aFI</i>	60755	<i>04b_pyramidal6aACC</i>	87195
<i>10a_spindle18aACC</i>	25797	<i>04b_pyramidal6aFI</i>	90482
<i>12a_spindle19aACC</i>	31841	<i>04b_pyramidal7aACC</i>	622553
<i>12o_spindle9aFI</i>	29320	<i>05a_pyramidal10aACC</i>	201506
<i>13o_spindle10aFI</i>	43081	<i>05a_pyramidal8aACC</i>	139975
<i>15o_spindle12aFI</i>	101548	<i>05b_pyramidal7aFI</i>	208203
<i>16o_spindle13aFI</i>	18266	<i>05b_pyramidal8aFI</i>	124350
<i>19o_spindle14aFI</i>	25786	<i>05b_pyramidal9aACC</i>	366659
<i>21o_spindle15aFI</i>	28822	<i>06a_pyramidal11aACC</i>	319574
<i>23o_spindle16aFI</i>	30073	<i>06b_pyramidal10aFI</i>	106808
<i>25o_spindle17aFI</i>	52919	<i>06b_pyramidal12aACC</i>	277718
<i>26o_spindle18aFI</i>	36239	<i>07a_pyramidal13aACC</i>	155854
<i>27o_spindle19aFI</i>	50807	<i>07b_pyramidal14aACC</i>	309789
<i>28o_spindle20aFI</i>	56036	<i>08o_pyramidal11aFI</i>	419651
<i>28o_spindle21aFI</i>	17581	<i>10a_pyramidal15aACC</i>	56184
<i>29o_spindle22aFI</i>	18414	<i>11a_pyramidal16aACC</i>	222732
<i>30o_spindle23aFI</i>	26357	<i>11o_pyramidal12aFI</i>	380293
<i>22o_pyramidal16aFI</i>	389878	<i>17o_pyramidal13aFI</i>	326989
<i>24o_pyramidal17aFI</i>	245058	<i>18o_pyramidal14aFI</i>	338453
<i>25o_pyramidal18aFI</i>	71209	<i>20o_pyramidal15aFI</i>	247116
<i>31o_pyramidal19aFI</i>	619390		

Table 12: Names and sizes of all the neuron finite elements meshes generated by Tetgen with default settings ( $Htetgen = -1$ ). The number of FE elements (not shown) is approximately four times the number of FE nodes.

Neuron ID	Brain region	Average diameter ( $\mu m$ )	Overall height ( $\mu m$ )	Soma volume ( $\mu m^3$ )	Total volume ( $\mu m^3$ )
<i>02a_pyramidal2aFI</i>	fronto-insula	1.27	404.85	19701.05	25639.61
<i>02b_pyramidal1aACC</i>	anterior cingulate	1.58	363.08	9065.56	11579.71
<i>02b_pyramidal1aFI</i>	fronto-insula	1.62	381.56	22475.52	29804.96
<i>03a_pyramidal9aFI</i>	fronto-insula	2.15	532.30	22557.27	30189.28
<i>03a_spindle2aFI</i>	fronto-insula	1.74	387.16	13406.27	17684.23
<i>03a_spindle6aFI</i>	fronto-insula	1.66	501.47	33458.19	37812.72
<i>03b_pyramidal2aACC</i>	anterior cingulate	1.48	189.29	2977.21	4487.44
<i>03b_pyramidal3aACC</i>	anterior cingulate	1.14	188.45	6005.06	6891.04
<i>03b_pyramidal3aFI</i>	fronto-insula	1.84	496.35	32510.62	46154.08
<i>03b_pyramidal4aFI</i>	fronto-insula	1.33	414.70	35253.85	39324.87
<i>03b_pyramidal9aFI</i>	fronto-insula	1.92	430.06	15263.14	20532.57
<i>03b_spindle4aACC</i>	anterior cingulate	1.43	336.33	3098.39	4070.19
<i>03b_spindle5aACC</i>	anterior cingulate	1.49	221.52	11925.78	13242.53
<i>03b_spindle6aACC</i>	anterior cingulate	1.33	398.36	4027.74	6058.67
<i>03b_spindle7aACC</i>	anterior cingulate	1.18	369.51	4982.41	6076.52

<i>04a_pyramidal4aACC</i>	anterior cingulate	1.52	705.96	5684.55	13637.33
<i>04a_pyramidal5aACC</i>	anterior cingulate	1.86	410.59	15010.43	24648.35
<i>04b_pyramidal5aFI</i>	fronto-insula	1.78	480.13	10312.87	17184.26
<i>04b_pyramidal6aACC</i>	anterior cingulate	1.41	465.88	3129.97	7497.12
<i>04b_pyramidal6aFI</i>	fronto-insula	1.56	310.21	14718.05	21708.21
<i>04b_pyramidal7aACC</i>	anterior cingulate	1.3	610.42	17060.60	28552.49
<i>04b_spindle3aFI</i>	fronto-insula	2.71	391.14	22569.99	28404.13
<i>05a_pyramidal10aACC</i>	anterior cingulate	1.74	281.20	16604.06	21826.41
<i>05a_pyramidal8aACC</i>	anterior cingulate	1.29	430.37	24709.77	29778.79
<i>05b_pyramidal7aFI</i>	fronto-insula	2.18	281.02	25720.11	32731.05
<i>05b_pyramidal8aFI</i>	fronto-insula	1.66	361.45	32527.06	44679.46
<i>05b_pyramidal9aACC</i>	anterior cingulate	1.56	650.60	23948.05	40014.54
<i>05b_spindle5aFI</i>	fronto-insula	2.35	381.88	15383.08	18190.63
<i>06a_pyramidal11aACC</i>	anterior cingulate	1.46	437.60	17222.02	29995.02
<i>06b_pyramidal10aFI</i>	fronto-insula	1.92	365.18	43127.81	52179.53
<i>06b_pyramidal12aACC</i>	anterior cingulate	1.52	324.94	17181.33	24931.32
<i>06b_spindle8aACC</i>	anterior cingulate	1.92	342.21	18237.49	19462.92
<i>07a_pyramidal13aACC</i>	anterior cingulate	1.37	325.73	6254.53	8738.01
<i>07b_pyramidal14aACC</i>	anterior cingulate	1.67	350.40	16053.07	22772.96
<i>07b_spindle9aACC</i>	anterior cingulate	1.75	437.87	21344.83	27307.48
<i>08a_spindle13aACC</i>	anterior cingulate	1.74	814.45	9911.07	14113.32
<i>08o_pyramidal11aFI</i>	fronto-insula	1.91	421.68	11512.38	24326.94
<i>09o_spindle7aFI</i>	fronto-insula	2.90	472.87	22052.10	27905.89
<i>09o_spindle8aFI</i>	fronto-insula	2.05	376.73	11923.76	15189.32
<i>10a_pyramidal15aACC</i>	anterior cingulate	1.40	341.48	8522.11	10960.84
<i>10a_spindle18aACC</i>	anterior cingulate	1.57	457.90	5895.17	7219.28
<i>11a_pyramidal16aACC</i>	anterior cingulate	1.27	486.31	8807.01	12263.84
<i>11o_pyramidal12aFI</i>	fronto-insula	1.91	369.34	70786.62	79516.92
<i>12a_spindle19aACC</i>	anterior cingulate	2.05	431.22	12178.08	15618.67
<i>12o_spindle9aFI</i>	fronto-insula	3.41	305.31	29983.79	36678.18
<i>13o_spindle10aFI</i>	fronto-insula	2.69	516.92	39866.55	46022.15
<i>15o_spindle12aFI</i>	fronto-insula	3.60	604.57	53192.65	79170.43
<i>16o_spindle13aFI</i>	fronto-insula	2.17	364.66	17467.88	18888.13
<i>17o_pyramidal13aFI</i>	fronto-insula	1.89	340.77	11004.30	21167.19
<i>18o_pyramidal14aFI</i>	fronto-insula	1.74	288.41	69851.56	78999.20
<i>19o_spindle14aFI</i>	fronto-insula	2.18	232.21	10507.15	12905.43
<i>20o_pyramidal15aFI</i>	fronto-insula	1.82	383.18	22344.32	27667.19
<i>21o_spindle15aFI</i>	fronto-insula	2.36	286.33	17567.69	29466.53
<i>22o_pyramidal16aFI</i>	fronto-insula	1.94	585.35	18776.05	29441.43
<i>23o_spindle16aFI</i>	fronto-insula	1.67	420.05	10429.13	13482.93
<i>24o_pyramidal17aFI</i>	fronto-insula	2.04	371.99	40986.40	47377.09
<i>25o_pyramidal18aFI</i>	fronto-insula	1.80	364.05	18587.13	23572.15
<i>25o_spindle17aFI</i>	fronto-insula	1.79	358.70	7897.44	13563.26
<i>26o_spindle18aFI</i>	fronto-insula	2.27	442.65	52911.93	56084.44
<i>27o_spindle19aFI</i>	fronto-insula	1.73	275.08	20640.14	25423.96
<i>28o_spindle20aFI</i>	fronto-insula	3.00	520.69	35442.59	51267.07
<i>28o_spindle21aFI</i>	fronto-insula	2.62	298.57	35579.06	37783.31
<i>29o_spindle22aFI</i>	fronto-insula	3.52	402.84	62928.22	83279.12
<i>31o_pyramidal19aFI</i>	fronto-insula	2.26	303.55	65950.80	86376.72

Table 13: The morphological characteristics of the neurons.

As an illustration, in Figure 13 we show the finite elements meshes of 4 spindle neurons and in Figure 14 we show the finite elements meshes of 4 pyramidal neurons.

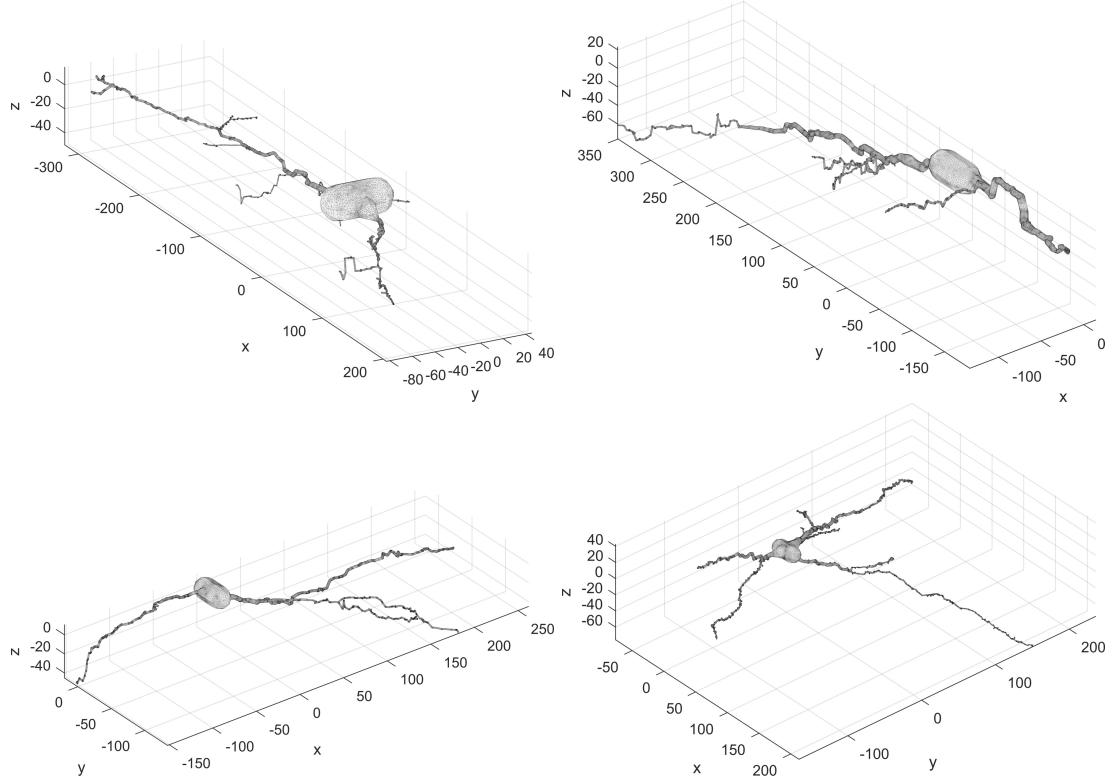


Figure 13: The finite elements meshes of four spindle neurons. The unit is  $\mu\text{m}$ . Top left: *03a\_spindle6aFI*. Top right: *28o\_spindle20aFI*. Bottom left: *09o\_spindle8aFI*. Bottom right: *25o\_spindle17aFI*.

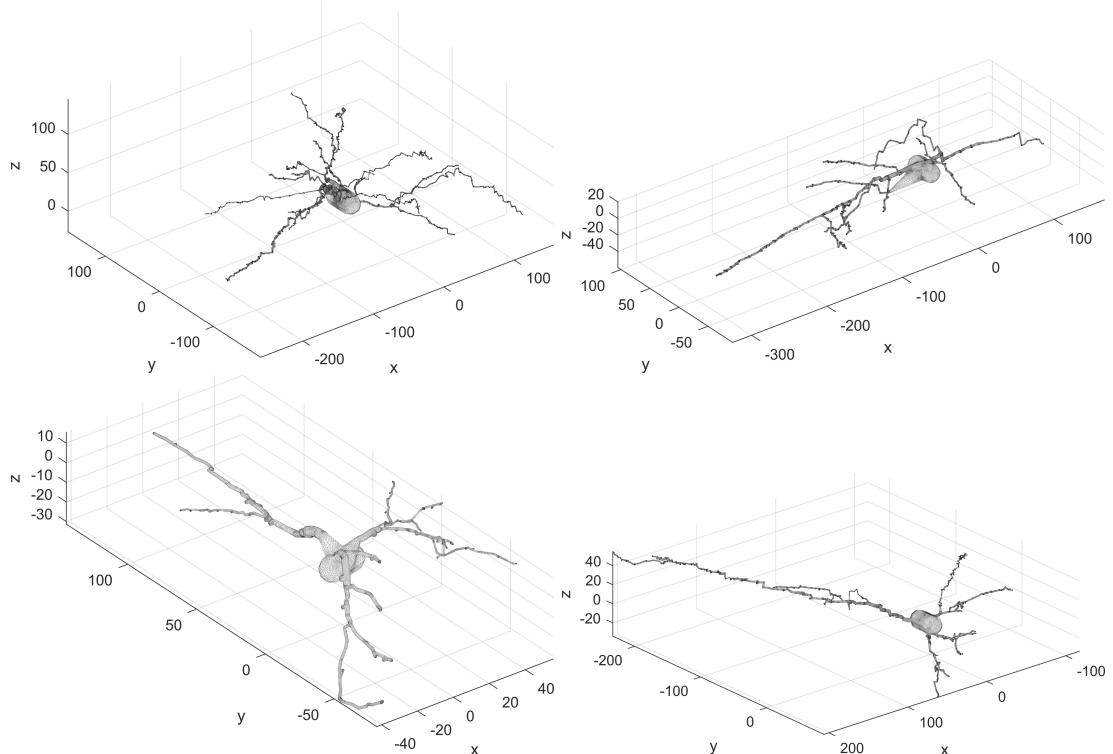


Figure 14: The finite elements meshes of four pyramidal neurons. The unit is  $\mu\text{m}$ . Top left: *02a\_pyramidal2aFI*. Top right: *03b\_pyramidal9aFI*. Bottom left: *03b\_pyramidal2aACC*. Bottom right: *10a\_pyramidal15aACC*.

## 7.5 Commented driver for typical simulations

```

1 %%% The DISTRIBUTE folder contains a commented general purpose driver
2 %%% called driver_spinductor_neuronmodule_commented.m.
3 %%% It is highly recommended to read this driver to understand the
4 %%% workflow
5 %%% driver_spinductor_neuronmodule_commented.m does not use saved
6 %%% simulation data,
7 %%% all simulations are run from scratch.
8 %%% The user is advised to read the latest version
9 %%% from \url{https://github.com/jingrebeccali/SpinDoctor}
10 clear all;
11 close all;
12 restoredefaultpath;
13
14 %%% User needs to choose to see some of the typical plots or not.
15 DO_PLOTS = true;
16
17 %%% User needs to choose whether to output the solution of the BTPDE,
18 %%% in the case only one diffusion direction is simulated.
19 %%% If the finite elements mesh is large, it is
20 %%% recommended not to output the magnetization.
21 %%% If the flag is set to false, the function PLOT_PDESOLUTION cannot
22 %%% be
23 %%% called.
24 %%% For multiple diffusion directions (HARDI), the magnetization is
25 %%% never
26 %%% outputted and the flag below is not taken into account.
27 OUTPUT_MAGNETIZATION = true;
28
29 %%% User needs to define the directory where the 3 user provided input
30 %%% files are kept.
31 user_inputfiles_dir = 'params_files/
32 %%% params_spinductor_neuronmodule_commented';
33
34 %%% User needs to define the names of the 3 user provided input files
35 %%% found in user_inputfiles_dir
36 fname_params_cells = 'params_cells_neuron.in';
37 fname_params_simul_domain = 'params_simul_domain_neuron.in';
38 fname_params_simul_experi = 'params_simul_experi_neuron.in';
39
40 %%% We add the path to the directory of the 3 user provided input
41 %%% files.
42 eval(['addpath ', user_inputfiles_dir]);
43
44 %%% We add the path to the top level source code directory.
45 addpath SRC
46 %%% We add the path to the deeper level source code directories.
47 %%% We note the functions that need to be called in a typical
48 %%% simulation
49 %%% situation are contained in the top level directory 'SRC'.
50 %%% The less often used functions are kept in the deeper levels.
51 addpath SRC/PDE SRC/DMRI SRC/FEM SRC/GEOM SRC/TETGEN
52
53 %%% We read the params_cells input file and create the geometrical
54 %%% configuration

```

```

48 %%% In the first line of the params_cells input file, we expect a
49     number
50 %%% between 1 and 3;
51 %% 1 = ellipsoids, 2 = cylinders, 3 = neuron from msh file;
52 [params_cells, fname_cells] = create_geom(fname_params_cells);
53
54 %%% We read the params_simul_domain input file;
55 [params_domain_geom, params_domain_pde, params_domain_femesh] ...
56     = read_params_simul_domain(fname_params_simul_domain);
57
58 %%% We set up the PDE model in the geometrical compartments.
59 [DIFF_cmpts, kappa_bdys, IC_cmpts, OUT_cmpts_index, ECS_cmpts_index,
60     IN_cmpts_index, Ncmpt, Nboundary] ...
61     = PREPARE_PDE(params_cells.ncell, params_cells.cell_shape,
62                     params_domain_geom, params_domain_pde);
63
64 %%% We make a directory to store the finite elements mesh for this
65     simulation;
66 save_meshdir_path = [fname_cells, '_dir'];
67 tf = isfolder(save_meshdir_path);
68 if (~tf)
69     call_cmd = ['mkdir ', save_meshdir_path];
70     disp(call_cmd);
71     eval([call_cmd]);
72 end
73 %%% We use an existing finite elements mesh or we create a new finite
74 %%% elements mesh.
75 %% The name of the finite elements mesh is stored in the string
76     fname_tetgen_femesh
77 ns = regexp(fname_cells, '/');
78 save_mesh_name = [fname_cells(ns(end)+1:end), 'Htetgen', num2str(
79     params_domain_femesh.Htetgen), 'msh'];
80 fname_tetgen = [save_meshdir_path, '/', save_mesh_name];
81 fname_tmp = [fname_tetgen, '.1'];
82 tf = isfile([fname_tmp, '.node']);
83 if (tf)
84     fname_tetgen_femesh = fname_tmp;
85 else
86     [fname_tetgen_femesh] = ...
87         create_femesh_fromcells(params_cells, fname_cells,
88             params_domain_geom, params_domain_femesh, fname_tetgen);
89 end
90
91 %%% We do not deform the finite elements mesh if in the Neuron module.
92 %%% We do deform the finite elements mesh if in SpinDoctor White
93     Matter
94 %%% mode.
95 %% The finite elements mesh is stored in mymesh
96 if (params_cells.cell_shape == 3)
97     params_cells.para_deform = [0,0]';
98     [mymesh, cmpts_bdys_mat] = read_tetgen(fname_tetgen_femesh,
99         params_cells.para_deform, Ncmpt, Nboundary);
100 else
101     [mymesh, cmpts_bdys_mat] = read_tetgen(fname_tetgen_femesh,
102         params_cells.para_deform, Ncmpt, Nboundary);
103 end

```

```

96 %%% We plot the finite elements mesh
97 if (DO_PLOTS)
98     PLOT_FEMESH(mymesh,OUT_cmpts_index,ECS_cmpts_index,IN_cmpts_index);
99 end
100
101 %%% We read the params_simul_experi input file;
102 [experi_common,experi_hadc,experi_btpde] ...
103 = read_params_simul_experi(fname_params_simul_experi);
104
105 %%% We get the volume and the surface area quantities from the mesh
106 [VOL_cmpts,SA_cmpts,SAu_cmpts,VOL_allcmpts,WF_cmpts,SoV_cmpts] ...
107 = GET_VOL_SA(mymesh,experi_common.gdir);
108
109 nexperi = length(experi_common.sdeltavec);
110
111 %%% We run the simulation for one diffusion-encoding direction
112 if (experi_common.ngdir_total == 1)
113     if (~isempty(experi_btpde))
114         nb = size(experi_btpde.bvalues,2);
115         %%% We solve the BTPDE
116         [SOL,SIG_BTPDE_cmpts,SIG_BTPDE_allcmpts,difftime,ctime_btpde]
117         ...
118         = BTPDE(experi_btpde,mymesh,DIFF_cmpts,kappa_bdys,IC_cmpts,
119             OUTPUT_MAGNETIZATION);
120         %%% We fit the ADC of the signal
121         [ADC_BTPDE_cmpts,ADC_BTPDE_allcmpts,ADC_allcmpts_S0] ...
122         = FIT_SIGNAL(SIG_BTPDE_cmpts,SIG_BTPDE_allcmpts,
123             experi_btpde.bvalues);
124         %%% We obtain the signal of free diffusion
125         [Sig_free,ADC_free_allcmpts] = ADCFREE(experi_btpde.bvalues,
126             DIFF_cmpts,VOL_cmpts,IC_cmpts);
127         %%% We plot the BTPDE signal, the S0*exp(-ADC*b) curve, and
128         %%% the
129         %%% free diffusion curves together.
130         if (DO_PLOTS)
131             PLOT_SIGNAL(experi_btpde.bvalues,SIG_BTPDE_allcmpts,
132                 Sig_free,ADC_allcmpts_S0,ADC_BTPDE_allcmpts,'BTPDE');
133         end
134         %%% We plot the magnetization solution of the BTPDE if the
135         %%% user
136         %% has set the flag OUTPUT_MAGNETIZATION to true
137         if (DO_PLOTS & OUTPUT_MAGNETIZATION)
138             for iexperi = 1:nexperi
139                 for ib = 1:nb
140                     bv = experi_btpde.bvalues(iexperi,ib);
141                     title_str = ['BTPDE magnetization. ', 'Experi ',...
142                         num2str(iexperi), ', b= ', num2str(bv)];
143                     PLOT_PDESOLUTION(mymesh,SOL{iexperi}{ib},
144                         OUT_cmpts_index,ECS_cmpts_index,IN_cmpts_index,
145                         title_str);
146                 end
147             end
148         end
149     end
150     if (~isempty(experi_hadc))
151         %%% We solve the HADC model
152         [ADC_HADC_cmpts,ADC_HADC_allcmpts,ctime_btpde] ...
153         = HADC(experi_hadc,mymesh,DIFF_cmpts,IC_cmpts);

```

```

145 %%% We find the short time approximation (STA) of the
146     effective
147     %% diffusion.
148     [ADC_STA_cmpts,ADC_STA_allcmpts] = STA(experi_common,DIFF_cmpts
149         ,VOL_cmpts,SAu_cmpts,IC_cmpts);
150     %% We plot the HADC and the STA
151     if (DO_PLOTS)
152         PLOT_ADC(ADC_HADC_cmpts,ADC_HADC_allcmpts,DIFF_cmpts,'HADC')
153             );
154         PLOT_ADC(ADC_STA_cmpts,ADC_STA_allcmpts,DIFF_cmpts,'STA');
155     end
156 end
157 %% We run the simulation for many diffusion-encoding directions
158     uniformly
159 %% distributed in the unit sphere in 3 dimensions.
160 if (experi_common.ngdir_total > 1)
161
162     ngdir_total = experi_common.ngdir_total;
163     %% We obtain the multiple diffusion-encoding directions uniformly
164     %% distributed in the unit sphere in 3 dimensions.
165     [points_gdir,graddir_index,negii] = HARDI PTS(ngdir_total);
166
167
168 if (~isempty(experi_btpde))
169     %% We solve the BTPDE in many diffusion directions.
170     [SIG_BTPDE_cmpts_hardi,SIG_BTPDE_allcmpts_hardi,
171         ctime_btpde_hardi] ...
172         = SIG_BTPDE_HARDI(experi_btpde,mymesh,DIFF_cmpts,kappa_bdys
173             ,IC_cmpts,...)
174         points_gdir,graddir_index,negii);
175     %% We plot the normalized signal in many diffusion directions
176         .
177
178     if (DO_PLOTS)
179         S0 = sum((IC_cmpts.*VOL_cmpts));
180         nb = size(experi_btpde.bvalues,2);
181         for iexperi = 1:nexperi
182             for ib = 1:nb
183                 bv = experi_btpde.bvalues(iexperi,ib);
184                 title_str = ['BTPDE. All Cmpts. ','Experi ',...
185                     num2str(iexperi),', b= ',num2str(bv)];
186                 PLOT_HARDI_PT(points_gdir, ...
187                     squeeze(real(SIG_BTPDE_allcmpts_hardi(:,iexperi
188                         ,ib))/S0,title_str);
189                 end
190             end
191         end
192     end
193 %% We solve the HADC model
194 if (~isempty(experi_hadc))
195     %% We solve the HADC model in many diffusion directions.
196     [ADC_HADC_cmpts_hardi,ADC_HADC_allcmpts_hardi,ctime_hadc_hardi]
197         ...
198         = HADC_HARDI(experi_hadc,mymesh,DIFF_cmpts,IC_cmpts, ...
199             points_gdir,graddir_index,negii);
200     %% We plot the normalized ADC (ADC/DO) in many diffusion
201     %% directions.
202     if(DO_PLOTS)

```

```
193 ADC0 = sum((DIFF_cmpts.*VF_cmpts));
194 for iexperi = 1:nexperi
195     title_str = ['HADC. All Cmpts. ', 'Experi ', num2str(
196         iexperi)];
197     PL0T_HARDI_PT(points_gdir, ...
198                     squeeze(ADC_HADC_allcmpts_hardi(:, iexperi)/ADCO),
199                     title_str);
200     end
201 end
```

## References

- [1] J.-R. Li, V.-D. Nguyen, T. N. Tran, J. Valdman, C.-B. Trang, K. V. Nguyen, D. T. S. Vu, H. A. Tran, H. T. A. Tran, T. M. P. Nguyen, Spindoror: A matlab toolbox for diffusion mri simulation, NeuroImage 202 (2019) 116120. doi:<https://doi.org/10.1016/j.neuroimage.2019.116120>.  
URL <http://www.sciencedirect.com/science/article/pii/S1053811919307116>
- [2] H. Si, Tetgen, a delaunay-based quality tetrahedral mesh generator, ACM Trans. Math. Softw. 41 (2) (2015) 11:1–11:36. doi:10.1145/2629697.  
URL <http://doi.acm.org/10.1145/2629697>
- [3] T. Rahman, J. Valdman, Fast matlab assembly of fem matrices in 2d and 3d: nodal elements, Applied Mathematics and Computation 219 (13) (2013) 7151–7158.
- [4] D. V. Nguyen, J.-R. Li, D. Grebenkov, D. L. Bihan, A finite elements method to solve the bloch–torrey equation applied to diffusion magnetic resonance imaging, Journal of Computational Physics 263 (0) (2014) 283 – 302. doi:10.1016/j.jcp.2014.01.009.  
URL <http://www.sciencedirect.com/science/article/pii/S0021999114000308>
- [5] E. O. Stejskal, 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) 288–292. doi:10.1063/1.1695690.
- [6] P. T. Callaghan, J. Stepinak, Frequency-domain analysis of spin motion using modulated-gradient NMR, Journal of Magnetic Resonance, Series A 117 (1) (1995) 118–122.  
URL <http://www.sciencedirect.com/science/article/pii/S1064185885799597>
- [7] M. D. Does, E. C. Parsons, J. C. Gore, Oscillating gradient measurements of water diffusion in normal and globally ischemic rat brain, Magn. Reson. Med. 49 (2) (2003) 206–215. doi:10.1002/mrm.10385.
- [8] J. Xu, M. Does, J. Gore, Numerical study of water diffusion in biological tissues using an improved finite difference method, Physics in medicine and biology 52 (7).  
URL <http://view.ncbi.nlm.nih.gov/pubmed/17374905>
- [9] S. Schiavi, H. Haddar, J.-R. Li, A macroscopic model for the diffusion mri signal accounting for time-dependent diffusivity, SIAM Journal on Applied Mathematics Accepted.
- [10] P. P. Mitra, P. N. Sen, L. M. Schwartz, P. Le Doussal, Diffusion propagator as a probe of the structure of porous media, Physical review letters 68 (24) (1992) 3555–3558.
- [11] P. P. Mitra, P. N. Sen, L. M. Schwartz, Short-time behavior of the diffusion coefficient as a geometrical probe of porous media, Phys. Rev. B 47 (1993) 8565–8574.
- [12] D. V. Nguyen, J.-R. Li, D. Grebenkov, D. Le Bihan, A finite elements method to solve the Bloch–Torrey equation applied to diffusion magnetic resonance imaging, Journal of Computational Physics 263 (0) (2014) 283–302.  
URL <http://www.sciencedirect.com/science/article/pii/S0021999114000308>
- [13] G. A. Ascoli, D. E. Donohue, M. Halavi, Neuromorpho.org: A central resource for neuronal morphologies, Journal of Neuroscience 27 (35) (2007) 9247–9251. arXiv:<http://www.jneurosci.org/content/27/35/9247.full.pdf>, doi:10.1523/JNEUROSCI.2055-07.2007.  
URL <http://www.jneurosci.org/content/27/35/9247>
- [14] K. K. Watson, T. K. Jones, J. M. Allman, Dendritic architecture of the von economo neurons, Neuroscience 141 (3) (2006) 1107–1112.