Journal of Computational Physics 375 (2018) 271–290

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A partition of unity finite element method for computational diffusion MRI

Van-Dang Nguyen [a,*], Johan Jansson [a], Johan Hoffman [a], Jing-Rebecca Li [b]

[a] Department of Computational Science and Technology, KTH Royal Institute of Technology, Sweden
[b] INRIA Saclay-Equipe DEFI, CMAP, Ecole Polytechnique Route de Saclay, 91128, Palaiseau Cedex, France

ARTICLE INFO

Article history:
Received 13 December 2017
Received in revised form 15 August 2018
Accepted 23 August 2018
Available online 29 August 2018

Keywords:
Computational diffusion MRI
Bloch–Torrey equation
Partition of unity finite element method
Interface conditions
Weak pseudo-periodic conditions
FEniCS/FEniCS-HPC

ABSTRACT

The Bloch–Torrey equation describes the evolution of the spin (usually water proton) magnetization under the influence of applied magnetic field gradients and is commonly used in numerical simulations for diffusion MRI and NMR. Microscopic heterogeneity inside the imaging voxel is modeled by interfaces inside the simulation domain, where a discontinuity in the magnetization across the interfaces is produced via a permeability coefficient on the interfaces. To avoid having to simulate on a computational domain that is the size of an entire imaging voxel, which is often much larger than the scale of the microscopic heterogeneity as well as the mean spin diffusion displacement, smaller representative volumes of the imaging medium can be used as the simulation domain. In this case, the exterior boundaries of a representative volume either must be far away from the initial positions of the spins or suitable boundary conditions must be found to allow the movement of spins across these exterior boundaries.

Many approaches have been taken to solve the Bloch–Torrey equation but an efficient high performance computing framework is still missing. In this paper, we present formulations of the interface as well as the exterior boundary conditions that are computationally efficient and suitable for arbitrary order finite elements and parallelization. In particular, the formulations are based on the partition of unity concept which allows for a discontinuous solution across interfaces conforming with the mesh with weak enforcement of real (in the case of interior interfaces) and artificial (in the case of exterior boundaries) permeability conditions as well as an operator splitting for the exterior boundary conditions. The method is straightforward to implement and it is available in FEniCS for moderate-scale simulations and in FEniCS-HPC for large-scale simulations. The order of accuracy of the resulting method is validated in numerical tests and a good scalability is shown for the parallel implementation. We show that the simulated dMRI signals offer good approximations to reference signals in cases where the latter are available and we performed simulations for a realistic model of a neuron to show that the method can be used for complex geometries.

© 2018 Elsevier Inc. All rights reserved.

* Corresponding author.
E-mail addresses: vdnguyen@kth.se (V.-D. Nguyen), jjan@kth.se (J. Jansson), jhoffman@kth.se (J. Hoffman), jingrebecca.li@inria.fr (J.-R. Li).

https://doi.org/10.1016/j.jcp.2018.08.039
0021-9991/© 2018 Elsevier Inc. All rights reserved.

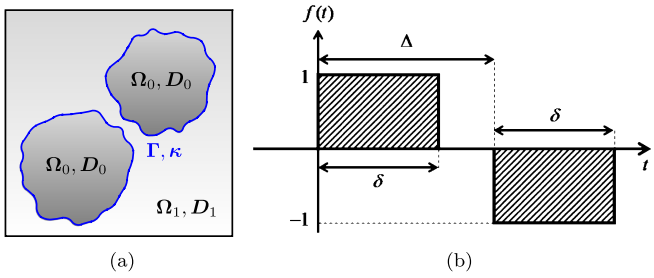


Fig. 1. A composed domain $\Omega = \Omega_0 \cup \Omega_1$ (a), and a PGSE sequence (b).

1. Introduction

Diffusion nuclear magnetic resonance (NMR) and its medical application, diffusion magnetic resonance imaging (MRI), are powerful tools to non-invasively probe microstructure. Information about the microstructure can be inferred from the diffusion characteristics of water molecules, reflected in the signal attenuation of the transverse magnetization in the imaging voxels after the application of a sequence of magnetic field gradients called diffusion-encoding gradients.

The evolution of the complex transverse magnetization can be described by the Bloch–Torrey equation [1]. Thus, this equation plays a vital role in numerical simulation of diffusion MRI and diffusion NMR. Microscopic heterogeneity inside the imaging voxel is modeled by interfaces inside the simulation domain, where a discontinuity in the magnetization across the interfaces is produced via the imposition of a permeability condition.

For simplicity we consider a medium composed of two compartments: $\Omega = \Omega_0 \cup \Omega_1$. We note that each compartment may contain disconnected parts (see Fig. 1a). The Bloch–Torrey equation for the complex transverse magnetization $u(\boldsymbol{x}, t)$ is

$$\frac{\partial u(\boldsymbol{x}, t)}{\partial t} = -\mathcal{I}\gamma f(t)\, \boldsymbol{g} \cdot \boldsymbol{x}\, u(\boldsymbol{x}, t) + \nabla \cdot \Big(\boldsymbol{D}(\boldsymbol{x})\, \nabla u(\boldsymbol{x}, t) \Big) \tag{1}$$

where $\mathcal{I}$ is the complex unit ($\mathcal{I}^2 = -1$), $\gamma = 2.67513 \times 10^8$ rad s^{-1}T^{-1} is the gyromagnetic ratio of the water proton, $\boldsymbol{D}(\boldsymbol{x})$ is the diffusion tensor, and $\boldsymbol{g} = (g_1, \cdots, g_d)$ is the diffusion gradient including gradient strength and gradient direction, d is the space dimension. For the simplest case, the diffusion tensor is isotropic, i.e. $\boldsymbol{D}(\boldsymbol{x}) = D(\boldsymbol{x})\, \mathbb{I}$ where $\mathbb{I}$ is the identity matrix and $D(\boldsymbol{x})$ is the diffusion coefficient which is homogeneous on each compartment and can differ on different compartments. In multidimensional diffusion MRI, one can use an anisotropic diffusion tensor $\boldsymbol{D}(\boldsymbol{x})$ which is symmetric with 6 unique entries (see [2] and references therein).

The temporal profile $f(t)$ can vary for different applications and the most commonly used diffusion-encoding sequence is called the pulsed-gradient spin echo (PGSE) sequence [3]. For this sequence, one can write $f(t)$ in the following way (see also Fig. 1b):

$$f(t) = \begin{cases} 1, & 0 \le t \le \delta, \\ -1, & \Delta < t \le \Delta + \delta, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The quantity δ is the duration of the diffusion-encoding gradient pulse and Δ is the time delay between the start of the two pulses.

On the interfaces the magnetization is allowed to be discontinuous via the use of a permeability coefficient κ [4]

$$\boldsymbol{D}_0(\boldsymbol{x}) \nabla u_0(\boldsymbol{x}, t) \cdot \boldsymbol{n}_0 = -\boldsymbol{D}_1(\boldsymbol{x}) \nabla u_1(\boldsymbol{x}, t) \cdot \boldsymbol{n}_1 = \kappa \Big(u_1(\boldsymbol{x}, t) - u_0(\boldsymbol{x}, t) \Big), \tag{3}$$

where $\boldsymbol{x} \in \Gamma = \partial\Omega_0 \cap \partial\Omega_1$ and $\boldsymbol{n}^k$ is a normal vector pointing outward Ω_k. Concerning the boundary conditions on the exterior boundaries $\partial\Omega$, there are two options that are very often employed. One is placing the spins to be simulated sufficiently away from $\partial\Omega$ and impose simple boundary conditions on $\partial\Omega$ such as homogeneous Neumann conditions. This supposes that the spins would have low probability of having arrived at $\partial\Omega$ during the diffusion experiment. This will not be discussed further in this paper since such boundaries are easy to implement.

Another option is to place the spins anywhere desired, but to assume that Ω is repeated periodically in all space directions to fill $\mathbb{R}^d$, for example, $\Omega = \prod_{k=1}^{d} [a_k, b_k]$. So, one can mimic the phenomenon where the water molecules can enter and exit the computational domain. Under this assumption of periodic continuation of the geometry, the magnetization satisfies pseudo-periodic boundary conditions on $\partial\Omega$ [5]

$$\begin{aligned} u_m &= u_s e^{\mathcal{I}\theta_k(t)}, \\ \boldsymbol{D}\, \nabla u_m \cdot \boldsymbol{n} &= \boldsymbol{D}\, \nabla u_s \cdot \boldsymbol{n}\, e^{\mathcal{I}\theta_k(t)}, \end{aligned} \tag{4}$$

where

$$u_m = u(\boldsymbol{x},t)|_{x_k=a_k}, \quad u_s = u(\boldsymbol{x},t)|_{x_k=b_k}$$

$$\nabla u_m \cdot \boldsymbol{n} = \nabla u(\boldsymbol{x},t)\cdot \boldsymbol{n}\Big|_{x_k=a_k}, \quad \nabla u_s \cdot \boldsymbol{n} = \nabla u(\boldsymbol{x},t)\cdot \boldsymbol{n}\Big|_{x_k=b_k}$$

and

$$\theta_k(t) := \gamma\, g_k\,(b_k - a_k)\,\mathcal{F}(t), k = 1,\cdots,d, \quad \mathcal{F}(t) = \int_0^t f(s)\,ds.$$

Here we use 'm' and 's' to indicate master and slave components of the pseudo-periodic boundary conditions. The master-slave method corresponds to the implementation of the conditions [6].

The MRI signal S is the total transverse magnetization $u(\boldsymbol{x},t)$ over Ω measured at the echo time T

$$S = \int_{\boldsymbol{x}\in\Omega} u(\boldsymbol{x},T)\,d\boldsymbol{x} \tag{5}$$

The signal is usually plotted against the gradient strength $q = \|\mathbf{g}\|$ or a quantity called the b-value which is defined as the following

$$b = \gamma^2\|\mathbf{g}\|^2 \int_0^T \left(\int_0^t f(s)\,ds\right) dt. \tag{6}$$

For the PGSE, it is

$$b = \gamma^2\|\mathbf{g}\|^2\delta^2\left(\Delta - \frac{\delta}{3}\right). \tag{7}$$

Solving the complete model, i.e., Eqs. (1), (3), (4), is challenging and many efforts have been made. In [7–9] a simplifying assumption called the narrow pulse approximation was used, where the pulse duration δ was assumed to be much smaller than the delay between pulses Δ. This assumption allows the solution of the diffusion equation instead of the more complicated Bloch–Torrey equation. More generally, numerical methods to solve the Bloch–Torrey equation with arbitrary temporal profiles have been proposed in [5,10–12]. The computational domain is discretized either by a Cartesian grid [5,13,10] or finite elements [7–9,11,12]. The unstructured mesh of a finite element discretization appeared to be better than a Cartesian grid in both geometry description and signal approximation [11]. For time discretization, both explicit and implicit methods have been used. In [9] a second order implicit time-stepping method called the generalized α-method [14] was used to allow for high frequency energy dissipation. An adaptive explicit Runge–Kutta Chebyshev (RKC) method of second order was used in [10,11]. It has been theoretically proven that the RKC allows for a much larger time-step compared to the standard explicit Euler method [15] although the RKC requires a time-step size proportional to h^2. In [11], there is an example showing that the RKC method is faster than the implicit Euler method. Recently, the Crank–Nicolson method (CN) was used in [12] to also allow for second order convergence in time.

The jump conditions (Eq. (3)) were treated differently in previous works. An average diffusion coefficient was introduced to approximate the permeability condition at the interfaces in [10]. The matrices for jump conditions were explicitly calculated and imposed directly in the stiffness matrix for linear finite elements in [11]. It was generalized to allow higher orders in [12].

The pseudo-periodic boundary conditions were implemented in a finite difference method [5] and a finite volume method [10] with Cartesian grids. Its direct implementation is inefficient on a triangulation with a finite element method as discussed in [11] since the boundary conditions are complex-valued and time-dependent. The amount of work to impose the boundary conditions is doubled and it needs to be repeated for each time step. A PDE transformation is used to transform the pseudo-periodic to periodic conditions. The time-dependent terms are removed from the bilinear form and the performance is, therefore, improved. Still, the periodic boundary conditions are strongly enforced on specially generated meshes where the nodes on the opposite faces should match each other. To generalize such an approach to more realistic problems is challenging due to several reasons. First, generating periodic unstructured meshes for biological tissues is difficult in general, given that often the biological cells themselves cut the exterior boundaries in a non-periodic fashion. Therefore, to simulate on more realistic geometries, the pseudo-periodic boundary conditions must be able to be imposed on non-periodic meshes. The weak imposition of the periodic boundary conditions for flow problems was considered in [16–18] where either the Lagrange multipliers are discretized with piecewise polynomials or the displacement is interpolated by polynomials. This allows for imposing the periodic boundary conditions on non-matching meshes. Although these

methods are efficient, the finite element matrices need to be constructed in a special way in the implementation. Thus, they are not preferable in terms of parallelization.

A high performance computing framework was proposed in [19] for large-scale simulations of diffusion MRI on supercomputers. Since the pseudo-periodic boundary conditions were very complicated to parallelize in that framework, we approximated them by using an artificial permeability at the external boundaries. The framework shows good parallel scalability but the time-step size needed to be quite small at high permeability to fulfill the CFL constraint. Also, the implementation is complicated by a large number of MPI communications.

This paper can be thought of as a continuation of the HPC framework with significant improvements to simplify the implementation and increase the efficiency. Many choices have been made for this purpose. First, we adapt a partition of unity finite element method (PUFEM) [20] and a cut finite element method (CutFEM) [21,22] to weakly enforce the interface and external boundary conditions. Thus, no explicit parallel implementation is needed to impose the permeability condition. It also allows for easy generalization to arbitrary order finite elements, in contrast to [11,12]. Second, we use the CN method as the time stepping method (as in [12]) as well as an operator splitting, so as to produce an unconditionally stable scheme even in the presence of a large artificial permeability at the external boundaries.

The paper is organized as follows. In Section 2 we introduce a simple idea for an L_2-projection of a discontinuous function using a continuous function space and a doubling of the solution field. Following this simple idea, in Section 3, we introduce the PUFEM for the Bloch–Torrey equation where the jump conditions are imposed directly to the weak forms. We go into more details in Section 4 about the space–time discretization where the θ-method is used as the time discretization. In Section 5, we discuss the implementation of the artificial boundary conditions on the external boundaries using a large artificial permeability coefficient. In Section 6 we describe the FEniCS implementation as a serial branch for moderate-scale simulations and FEniCS-HPC as a high performance branch with a good scalability. In Section 7, numerical results are presented concerning the choice of the time stepping method, the choice of the artificial permeability coefficient on the external boundaries, the convergence and scalability of the overall numerical method, ending with large-scale simulations of a pyramidal neuron to show that the method can be used to simulate diffusion MRI on complex geometries.

2. An L_2-projection of a discontinuous function

The interface conditions (Eq. (3)) give rise to solutions that may be discontinuous between two compartments. The question is how to approximate a function with a discontinuity across an internal interface conforming with the mesh using the continuous Galerkin method. We start from a simple idea to establish an L_2-projection of a discontinuous function p onto a piecewise continuous subspace $\boldsymbol{Q}^h$ of a Sobolev space $\boldsymbol{Q} = \boldsymbol{H}^1(\Omega)$.

The standard finite element method states: find $U \in \boldsymbol{Q}^h$ such that

$$(U, v^h)_{\Omega^h} = (p^h, v^h)_{\Omega^h}, \quad \forall v^h \in \boldsymbol{Q}^h \tag{8}$$

where $(\cdot,\cdot)_{\Omega^h}$ is the standard L_2-inner product in $\boldsymbol{Q}$, and Ω^h is the discretization of Ω based on a triangulation. This would give a bad approximation since U is a piecewise continuous function.

Now we consider a better approach based on the idea that every function p in Ω can be expressed as $p = (1-\Phi)\,p_0 + \Phi\,p_1$ where $p_k \in \boldsymbol{Q}$ and $p_k = p\big|_{\Omega_k}$ where the piecewise constant phase function Φ is defined as

$$\Phi(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in \Omega_0 \\ 1 & \text{if } \boldsymbol{x} \in \Omega_1 \end{cases} \tag{9}$$

This allows us to use a piecewise continuous function space $\boldsymbol{V}^h = [\boldsymbol{Q}^h]^2$ on Ω^h when searching for a discontinuous approximate function defined on Ω^h.

Let us consider the bilinear form

$$\boldsymbol{a}(U, v^h) = (U, v^h)_{\Omega^h} = (U, v^h)_{\Omega_0^h} + (U, v^h)_{\Omega_1^h} = (U_0, v_0^h)_{\Omega_0^h} + (U_1, v_1^h)_{\Omega_1^h} \tag{10}$$

One can, therefore, define U_k on the function space $\boldsymbol{Q}_k^h$ corresponding to Ω_k^h to couple the system but here we extend the solution U_i to the whole domain following the PUFEM [20]. The extension facilitates better for automation and parallelization as will be clear later. So, we write

$$\boldsymbol{a}(U, v^h) = \left((1-\Phi^h)\,U_0, v_0^h\right)_{\Omega^h} + \left(\Phi^h U_1, v_1^h\right)_{\Omega^h} \tag{11}$$

here Φ^h is a discretized function of Φ and it is element-wise constant.

The L_2-projection is stated as: find $U = (U_0, U_1) \in \boldsymbol{V}^h \equiv [\boldsymbol{Q}^h]^2$ such that

$$\boldsymbol{a}(U, v^h) = \boldsymbol{L}(v^h), \quad \forall v^h = (v_0^h, v_1^h) \in \boldsymbol{V}^h \tag{12}$$

where

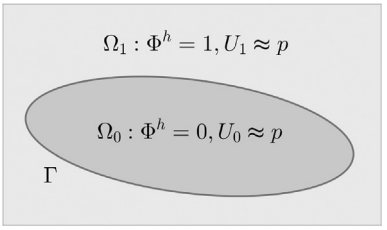


Fig. 2. Phase function Φ and the approximate solutions (U_0, U_1) on subdomains. The discontinuous function $\bar{U} = (1-\Phi)\,U_0 + \Phi\,U_1$ would be a better approximation to p.

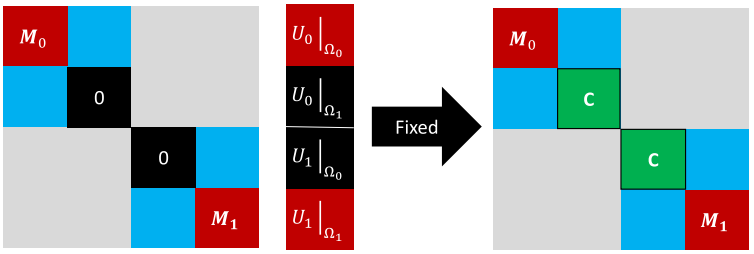


Fig. 3. Matrix coupling for the L_2-projection problem: the resulting matrix consists of two zero blocks which makes the matrix singular (left figure). One adds the matrix $\boldsymbol{C}$ to obtain a stabilized matrix (right figure).

$$\boldsymbol{L}(v^h) = \left(\left(1-\Phi^h\right) p_0^h, v_0^h\right)_{\Omega^h} + \left(\Phi^h\, p_1^h, v_1^h\right)_{\Omega^h}$$

Now $\bar{U} = (1-\Phi^h)\,U_0 + \Phi^h\,U_1$ is discontinuous and would be a better approximation to p. Fig. 2 shows how the phase function Φ^h and the approximation $U = (U_0, U_1)$ are defined in the subdomains.

The coupling Eq. (11) is singular since it contains two zero blocks. One needs, therefore, add to Eq. (11) a stabilization term. In [21], a stabilization term with projection operators was proposed to maintain the optimal convergence rate and the optimal condition number estimates as the following

$$\boldsymbol{a}_{mstab}(U, v^h) = \beta\left(\mathcal{P}_0^h(U_0), \mathcal{P}_0^h(v_0^h)\right)_{\Omega^h} + \beta\left(\mathcal{P}_1^h(U_1), \mathcal{P}_1^h(v_1^h)\right)_{\Omega^h} \tag{13}$$

where β is a positive constant and $\mathcal{P}_k^h (k = 0, 1)$ is the projection operator defined for conforming interface meshes as

$$\mathcal{P}_k^h(w_j^h) = \begin{cases} 0 & \text{if } \boldsymbol{x}_j \in \overline{\Omega}_k^h \\ w_j^h & \text{otherwise} \end{cases}$$

Here j represents the index of a degree of freedom corresponding to the coordinate $\boldsymbol{x}_j$, $w_j^h = w^h(\boldsymbol{x}_j)$, and $\overline{\Omega}_k^h$ should include all degrees of freedoms in Ω_k^h and on the interface Γ.

Fig. 3 shows what the matrix coupling looks like. The resulting matrix consists of two zero blocks which makes the matrix singular (left figure). One then adds a parametric matrix $\boldsymbol{C}$ corresponding to Eq. (13) to obtain a stabilized matrix (right figure). We note that although Eq. (13) is straightforward to implement as a bilinear form, β varies for different problems [21]. So, we need to choose an appropriate β to maintain the optimal convergence rate. Alternatively, we choose $\boldsymbol{C}$ to be the identity matrix, i.e. we insert one on the diagonal for all zero rows. These zeros rows certainly do not include the degrees of freedoms on the interface Γ. Actually, this technique is a special case of Eq. (13). We refer this as the `ident_zeros` technique. A numerical comparison between the projection technique and `ident_zeros` technique is shown in Appendix A.

3. A PUFEM for the Bloch–Torrey equation

Multiply both sides of Eq. (1) with a test function $v_k \in \boldsymbol{H}^1(\Omega_k)$ and integrate it over $\Omega_k\,(k = 0, 1)$, we obtain

$$\left(\frac{\partial u_k}{\partial t}, v_k\right)_{\Omega_k} = \left(-\mathcal{I}\gamma f(t)\,\boldsymbol{g}\cdot\boldsymbol{x}\,u_k, v_k\right)_{\Omega_k} + \left(\nabla\cdot\left(\boldsymbol{D}\,\nabla u_k\right), v_k\right)_{\Omega_k} \tag{14}$$

where $(\cdot,\cdot)$ is the inner product in $L_2(\Omega_k)$ and $u_k = u\big|_{\Omega_k}$. Apply the Green's first identity for the diffusion term and the homogeneous Neumann boundary conditions on $\partial\Omega$, Eq. (14) becomes

$$\left(\frac{\partial u_k}{\partial t}, v_k\right)_{\Omega_k} = \left(-\mathcal{I}\,\gamma\, f(t)\, \boldsymbol{g}\cdot\boldsymbol{x}\, u_k, v_k\right)_{\Omega_k} - \left(\boldsymbol{D}\,\nabla u_k, \nabla v_k\right)_{\Omega_k} + \int_\Gamma \boldsymbol{D}\,\nabla u_k\cdot\boldsymbol{n}_k\, v_k\, dS \tag{15}$$

where $\boldsymbol{n}_k$ is the normal vector pointing outward Ω_k. Now we couple Eq. (15) for the two domains to obtain

$$\begin{aligned}\sum_{k=0}^{1}\left(\frac{\partial u_k}{\partial t}, v_k\right)_{\Omega_k} &= \sum_{k=0}^{1}\left(-\mathcal{I}\,\gamma\, f(t)\, \boldsymbol{g}\cdot\boldsymbol{x}\, u_k, v_k\right)_{\Omega_k} \\ &\quad - \sum_{k=0}^{1}\left(\boldsymbol{D}\,\nabla u_k, \nabla v_k\right)_{\Omega_k} + \sum_{k=0}^{1}\int_\Gamma \boldsymbol{D}\,\nabla u_k\cdot\boldsymbol{n}_k\, v_k\, dS\end{aligned} \tag{16}$$

Let $\langle a\rangle := \int_\Gamma a\, dS$, $[\![a]\!] := a_0 - a_1$, and $\{a\} = \dfrac{a_0+a_1}{2}$, Eq. (16) becomes

$$\begin{aligned}\sum_{k=0}^{1}\left(\frac{\partial u_k}{\partial t}, v_k\right)_{\Omega_k} &= \sum_{k=0}^{1}\left(-\mathcal{I}\,\gamma\, f(t)\, \boldsymbol{g}\cdot\boldsymbol{x}\, u_k, v_k\right)_{\Omega_k} \\ &\quad - \sum_{k=0}^{1}\left(\boldsymbol{D}\,\nabla u_k, \nabla v_k\right)_{\Omega_k} + \left[\!\left[\left\langle \boldsymbol{D}\,\nabla u\cdot\boldsymbol{n}_0, v\right\rangle\right]\!\right]\end{aligned} \tag{17}$$

and the jump conditions (Eq. (3)) can be expressed as

$$\begin{aligned}&\left[\!\left[\boldsymbol{D}\,\nabla u\cdot\boldsymbol{n}^0\right]\!\right] = 0 \\ &\left\{\boldsymbol{D}\,\nabla u\cdot\boldsymbol{n}^0\right\} = -\kappa\,[\![u]\!]\end{aligned} \tag{18}$$

Using the fact that

$$[\![a\,b]\!] = \{a\}\,[\![b]\!] + [\![a]\!]\,\{b\}$$

we have

$$\left[\!\left[\left\langle \boldsymbol{D}\,\nabla u\cdot\boldsymbol{n}^0, v\right\rangle\right]\!\right] = -\kappa\left\langle [\![u]\!], [\![v]\!]\right\rangle \tag{19}$$

Finally, substituting Eq. (19) to Eq. (17) we obtain a compact variational form for the two-compartment model as

$$\sum_{k=0}^{1}\left(\frac{\partial u_k}{\partial t}, v_k\right)_{\Omega_k} = \sum_{k=0}^{1}\left(-\mathcal{I}\,\gamma\, f(t)\, \boldsymbol{g}\cdot\boldsymbol{x}\, u_k, v_k\right)_{\Omega_k} - \sum_{k=0}^{1}\left(\boldsymbol{D}\,\nabla u_k, \nabla v_k\right)_{\Omega_k} - \kappa\left\langle [\![u]\!], [\![v]\!]\right\rangle \tag{20}$$

4. Space–time discretization

The explicit RKC method [15,23] is suitable for moderately stiff ordinary differential equations and was used in [10,11] for simulations of diffusion MRI with low gradient strengths. We will show later that the method is inefficient for high gradient strength which has been used in NMR, for instance in [24]. As in [12], we will discretize the time domain by the θ-method which is defined as

$$U^\theta = \theta U^n + (1-\theta)U^{n-1}$$

Here we consider a partition of the time domain $0 = t_0 < t_1 < \cdots < t_N = T$ associated with the time intervals $I_n = (t^{n-1}, t^n]$ of length $k^n = t^n - t^{n-1}$ and U^n be an approximation of $u(\boldsymbol{x}, t)$ for a given a triangulation $\mathcal{T}^h$ at $t = t^n$.

The two extreme cases, i.e. $\theta = 0$ and $\theta = 1$, are well-known as the explicit Forward Euler (FE) and implicit Backward Euler (BE) methods. The time-step size in FE method is constrained by the CFL condition for the stability whereas the BE is unconditionally stable. Both cases are less interesting in practice since they both have first-order convergence. The most interesting method is with $\theta = 0.5$ in which we have a second-order method referred to as a Crank–Nicolson (CN) method. This method is implicit and unconditionally stable.

Similarly to Section 2, we introduce an element-wise constant function Φ^h to extend functions to the whole domain. The PUFEM corresponding to Eq. (20) with the time-stepping θ-method is stated as: Find $U^n = (U_0^n, U_1^n) \in \boldsymbol{V}^h$ such that

$$\left(\frac{U^n - U^{n-1}}{k^n}, v^h\right)_{\Omega_0\cup\Omega_1} = \left(-\mathcal{I}\,\gamma\, f(t)\,\boldsymbol{g}\cdot\boldsymbol{x}\,U^\theta, v^h\right)_{\Omega_0\cup\Omega_1} - \left(\boldsymbol{D}\,\nabla U^\theta, \nabla v^h\right)_{\Omega_0\cup\Omega_1} - \kappa\left\langle [\![U^\theta]\!], [\![v^h]\!]\right\rangle \tag{21}$$

for all $v^h = (v_0^h, v_1^h) \in \boldsymbol{V}^h$. Here $\left(a,b\right)_{\Omega_0^h\cup\Omega_1^h} = \left((1-\Phi^h)\,a_0, b_0\right)_{\Omega^h} + \left(\Phi^h a_1, b_1\right)_{\Omega^h}$.

The bilinear and linear forms are defined by

$$\begin{aligned} \boldsymbol{a}(U^n, v^h) &= \left(\frac{U^n}{k^n}, v^h\right)_{\Omega_0\cup\Omega_1} - \theta\, F(U^n, v^h, t^n) \\ \boldsymbol{L}(v^h) &= \left(\frac{U^{n-1}}{k^n}, v^h\right)_{\Omega_0\cup\Omega_1} + (1-\theta)\, F\left(U^{n-1}, v^h, t^{n-1}\right) \end{aligned} \tag{22}$$

where

$$F(u,v,t) = \left(-\mathcal{I}\,\gamma\, f(t)\,\boldsymbol{g}\cdot\boldsymbol{x}\,u, v\right)_{\Omega_0\cup\Omega_1} - \left(\boldsymbol{D}\,\nabla u, \nabla v\right)_{\Omega_0\cup\Omega_1} - \kappa\left\langle [\![u]\!], [\![v]\!]\right\rangle \tag{23}$$

The linear system of equations corresponding to the bilinear and linear forms (Eq. (22)) is

$$\boldsymbol{A}\,\boldsymbol{U}^n = \boldsymbol{F} \tag{24}$$

where

$$\boldsymbol{A} = \boldsymbol{M}\,(k^n)^{-1} - \theta\left(-\mathcal{I}\,\gamma\, f(t^n)\,\boldsymbol{J} - \boldsymbol{S} - \boldsymbol{I}\right) \tag{25}$$

Here $\boldsymbol{M}$ and $\boldsymbol{S}$ are referred to as the mass and stiffness matrices respectively, $\boldsymbol{J}$ and $\boldsymbol{I}$ are corresponding to the first and third terms on the right-hand side of F (Eq. (23)), i.e. $(\boldsymbol{g}\cdot\boldsymbol{x}\,u, v)$ and $\kappa\left\langle [\![u]\!], [\![v]\!]\right\rangle$. The approximate solution we search for is discontinuous and has the form $\bar{U}^n = (1-\Phi)\,U_0^n + \Phi\,U_1^n$.

5. Allow water exchange at the external boundaries

To avoid having to modify directly the finite element matrices, the pseudo-periodic boundary conditions (Eq. (4)) will be implemented weakly through the use of an artificial permeability coefficient, κ^e, similarly to [19] (see also in Appendix B). The artificial permeability condition at the external boundaries is of the form:

$$\begin{aligned} \boldsymbol{D}_m\nabla u_m\cdot\boldsymbol{n}_m &= \kappa^e\left(u_s\, e^{\mathcal{I}\,\theta_{ms}} - u_m\right), \\ \boldsymbol{D}_s\nabla u_s\cdot\boldsymbol{n}_s &= \kappa^e\left(u_m\, e^{\mathcal{I}\,\theta_{sm}} - u_s\right), \end{aligned} \tag{26}$$

where κ^e reflects how much the water can exchange at the boundaries and

$$\theta_{ms} = -\theta_{sm} = \gamma\;\boldsymbol{g}\cdot(\boldsymbol{x}_s - \boldsymbol{x}_m)\,\mathcal{F}(t).$$

As $\kappa^e \to \infty$, the jump conditions (Eq. (26)) become the pseudo-periodic conditions (Eq. (4)). We can use the jump conditions with a large artificial permeability κ^e to approximate the pseudo-periodic boundary conditions. Based on the fact that the transverse diffusion coefficient inside the membrane layer (which is perpendicular to Γ) D^n is related to the permeability κ and the thickness η of Γ in the following way (see [25]):

$$\kappa \approx \frac{D^n}{\eta},$$

we propose to use $\kappa^e = \frac{D}{h}$ where h is the element size. This choice also has the same form as the penalty parameter used in the Nitsche's method for the Dirichlet boundary conditions [26] (see also a review in [27] and references therein).

The stability is constrained by the CFL condition and the time-step size is inversely proportional to κ^e when we chose to approximate $u_s(t^n) \approx U_s^{n-1}$ and $u_m(t^n) \approx U_m^{n-1}$ [19]. The time step needs to be very small for fine meshes to ensure the stability. Here we use the following approximation (operator splitting) to have an unconditionally stable scheme

$$\begin{aligned}\boldsymbol{D}_m\nabla u_m\cdot\boldsymbol{n}_m &\approx \kappa^e\left(U_s^{n-1}\,e^{\mathcal{I}\,\theta_{ms}^n}-U_m^n\right),\\ \boldsymbol{D}_s\nabla u_s\cdot\boldsymbol{n}_s &\approx \kappa^e\left(U_m^{n-1}\,e^{\mathcal{I}\,\theta_{sm}^n}-U_s^n\right).\end{aligned}\tag{27}$$

This approach is more straightforward for parallelization than those in [16–18] since no matrix modification is needed to impose the permeability κ^e allowing a high-level, maintainable implementation. It also allows for non-matching meshes.

6. Implementation in FEniCS and FEniCS-HPC

FEniCS [28,29] is a collection of open-source packages to enable automated solution of differential equations. It provides automated evaluation of variational forms given a high-level description in mathematical notation. FEniCS-HPC [30,31] is a branch optimized for massively parallel architectures, and implements parallel time-dependent duality-based adaptive error control, implicit parameter-free turbulence modeling for flow and fluid-structure interaction by the use of stabilized FEM and shows strong linear scaling up to thousands of cores [32–37].

The proposed method has been implemented both in FEniCS (Python, C++) for moderate-scale simulations and in FEniCS-HPC (C++) for large-scale simulations. The bilinear and linear forms for the θ-method (Eq. (21)) can be implemented in Python-FEniCS in the form of a compact code with a few command lines as you can see in Listing 1 in the Appendix D. Specifically, since U^n is a complex function $U^n = U_r^n + \mathcal{I}\,U_{\mathcal{I}}^n$, the real and imaginary parts are solved in a system of equations

$$\begin{aligned}\left(\frac{U_r^n-U_r^{n-1}}{k^n},v_r^h\right)_{\Omega_0\cup\Omega_1} &= \left(\gamma\,f(t)\,\boldsymbol{g}\cdot\boldsymbol{x}\,U_{\mathcal{I}}^\theta,v_r^h\right)_{\Omega_0\cup\Omega_1}-\left(\boldsymbol{D}\,\nabla U_r^\theta,\nabla v_r^h\right)_{\Omega_0\cup\Omega_1}\\ &\quad-\kappa\left\langle[\![U_r^\theta]\!],[\![v_r^h]\!]\right\rangle\\ \left(\frac{U_{\mathcal{I}}^n-U_{\mathcal{I}}^{n-1}}{k^n},v_{\mathcal{I}}^h\right)_{\Omega_0\cup\Omega_1} &= \left(-\gamma\,f(t)\,\boldsymbol{g}\cdot\boldsymbol{x}\,U_r^\theta,v_{\mathcal{I}}^h\right)_{\Omega_0\cup\Omega_1}-\left(\boldsymbol{D}\,\nabla U_{\mathcal{I}}^\theta,\nabla v_{\mathcal{I}}^h\right)_{\Omega_0\cup\Omega_1}\\ &\quad-\kappa\left\langle[\![U_{\mathcal{I}}^\theta]\!],[\![v_{\mathcal{I}}^h]\!]\right\rangle\end{aligned}$$

To speed up the computation, it is more efficient to avoid assembling matrices at each time step. In practice, we can prepare some matrices beforehand for this purpose (see Listing 2 in Appendix D). The complement matrix $\boldsymbol{C}$ is set to identity by simply calling `M.ident_zeros()`. The software is available upon request.

7. Numerical results

Unless specified differently, a Krylov solver is used with the biconjugate gradient stabilized method, and the block-Jacobi preconditioner from the PETSc library.

7.1. Optimal convergence in space discretization

We consider a steady-state problem on $\Omega=\Omega_0\cup\Omega_1$

$$-\Delta u+u=p\tag{28}$$

where Ω_0 is either a circle of radius $r=0.5$, i.e. $\Omega_0=\left\{(x,y)\middle|x^2+y^2\le r^2\right\}$ or a square $\Omega_0=[-0.4,0.4]^2$ which is embedded in a square $\Omega=[-1,1]^2$.

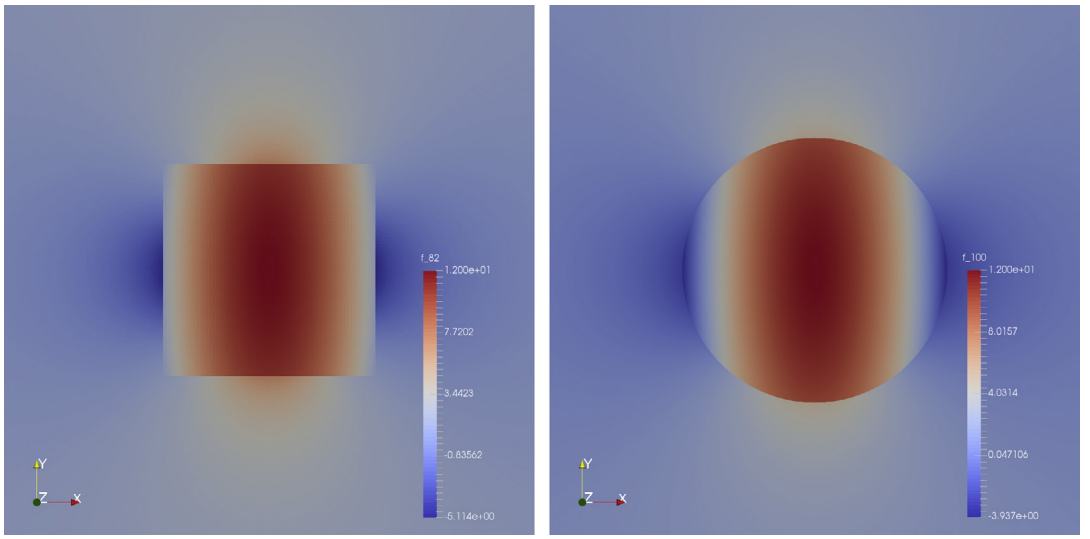
At the interface $\Gamma=\partial\Omega_0\cap\partial\Omega_1$, we impose the flux conditions

$$\begin{aligned}[\![\nabla u\cdot\boldsymbol{n}]\!] &= g_j\\ \{\nabla u\cdot\boldsymbol{n}\} &= g_a\end{aligned}\tag{29}$$

where the source p, the jump and average in flux g_j and g_a are computed corresponding to the reference solution (see Fig. 4)

$$u_e=\begin{cases}16\,(\frac{1}{2}-x^2)\,(4-y^2)-20 & \text{in }\Omega_0\\ 1+\frac{-x^2+y^2}{(x^2+y^2)^2} & \text{in }\Omega_1\end{cases}\tag{30}$$

The PUFEM: Find $(U_0,U_1)\in\boldsymbol{V}^h$ such that



(a) (b)

Fig. 4. The reference solutions for the test problem (Eq. (28)) on $\Omega = \Omega_0 \cup \Omega_1$ with the flux conditions (Eq. (29)). Ω_0 is either a square $\Omega_0 = [-0.4, 0.4]^2$ (a) or a circle of radius $r = 0.5$, i.e. $\Omega_0 = \left\{(x, y)|x^2 + y^2 \leq r^2\right\}$ (b) which is embedded in a square $\Omega = [-1, 1]^2$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$\boldsymbol{a}(U, v^h) = \boldsymbol{L}(v^h), \quad \forall v^h \in \boldsymbol{V}^h \tag{31}$$

where the bilinear and linear forms are defined as

$$\begin{aligned} \boldsymbol{a}(u, v) &= \Big((1-\Phi)\nabla u_0, \nabla v_0\Big) + \Big((1-\Phi)u_0, v_0\Big) + \Big(\Phi\nabla u_1, \nabla v_1\Big) + (\Phi u_1, v_1) \\ \boldsymbol{L}(v) &= \Big((1-\Phi)\, p, v_0\Big) + (\Phi\, p, v_1) + \langle g_j, \{v\}\rangle + \langle g_a, [\![v]\!]\rangle \end{aligned} \tag{32}$$

The Python code in Appendix C shows how to define them in the FEniCS platform.

We perform the convergence test with initial meshes shown in Fig. 5a and 5b in which a regular mesh with right angled triangles is considered for the square cell and arbitrary triangles for the circular cell. The numerical results show optimal convergence in L_2-norm and optimal condition number estimates for both cases when the linear (P1) and quadratic (P2) basis functions are considered (see 5c). Here we use the `ident_zeros` technique to fix the matrix singularity. A comparison with the projection technique is shown in Appendix A.

7.2. A comparison between the RKC method and the Crank–Nicolson method

We perform simulations on a two-layer cylinder of radius $R_0 = 10\,\mu\text{m}$ and $R_1 = 20\,\mu\text{m}$ (Fig. 6a) to compare the RKC and the CN methods. The diffusion coefficient in both compartments is $D_0 = D_1 = 3 \times 10^{-3}\,\text{mm}^2/\text{s}$. Between two layers a permeability of $\kappa = 5 \times 10^{-5}\,\text{m/s}$ was imposed and the PGSE with $\delta = 5000\,\mu\text{s}$ and $\Delta = 10{,}000\,\mu\text{s}$ was used. The gradient strength varies between 0 and 1.83 T/m. Fig. 6b shows a snapshot of the real part of the solution at $t = 3000\,\mu\text{s}$ for $q = 0.82\,\text{T/m}$.

The RKC controls the time-step sizes by a given error control `rtol` (see [23] for more details). In this simulation, we considered `rtol=1e-4` and `rtol=1e-6` which correspond to the average time-step size of 104 µs and 23 µs respectively. The number of function evaluations varies between 225 and 2358 for `rtol=1e-4` and between 445 and 5718 for `rtol=1e-6`. Three time-step sizes were considered for the Crank–Nicolson method, $\Delta t = 30, 150, 300\,\mu\text{s}$ which correspond to 500, 100 and 50 time steps.

For low gradient strength ($q < 1\,\text{T/m}$), both methods give good approximation. The RKC starts oscillating with $q > 0.5\,\text{T/m}$ for `rtol=1e-4` and the approximation is still good up to $q = 1.83\,\text{T/m}$ for `rtol=1e-6` although some oscillations start appearing when $q > 1\,\text{T/m}$ (Fig. 7a). The CN method appears to be much more stable. It works with larger time-step sizes. Especially, at $\Delta t = 30\,\mu\text{s}$, the numerical signal matches very well the reference solution (Fig. 7b).

7.3. Allow water exchange at the external boundaries

We first perform 2D simulations on a domain $\Omega = [-10\,\mu\text{m}, 10\,\mu\text{m}]^2$. A regular mesh with right-angled triangles is used with a uniform initial condition. We allow the water exchange at the external boundaries by imposing an artificial permeability $\kappa^e = \frac{D}{h}$ to mimic diffusion in the free space. The reference signals are available for two extreme cases: full

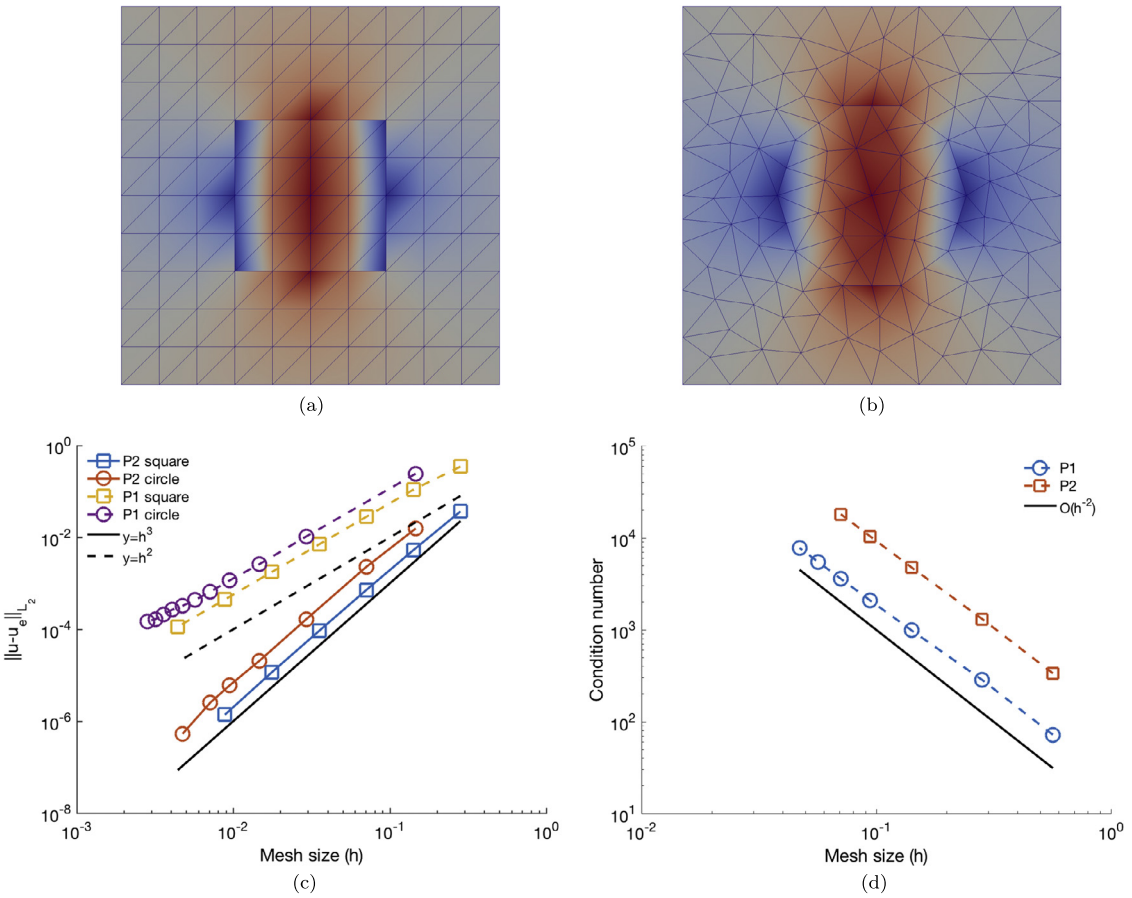


Fig. 5. The numerical solutions on initial meshes for the square cell (a) and circular cell (b). The mesh is regular with right triangle in the first case and arbitrary triangles for the second case. Both cases give optimal convergence (c) and optimal condition number estimates (d).

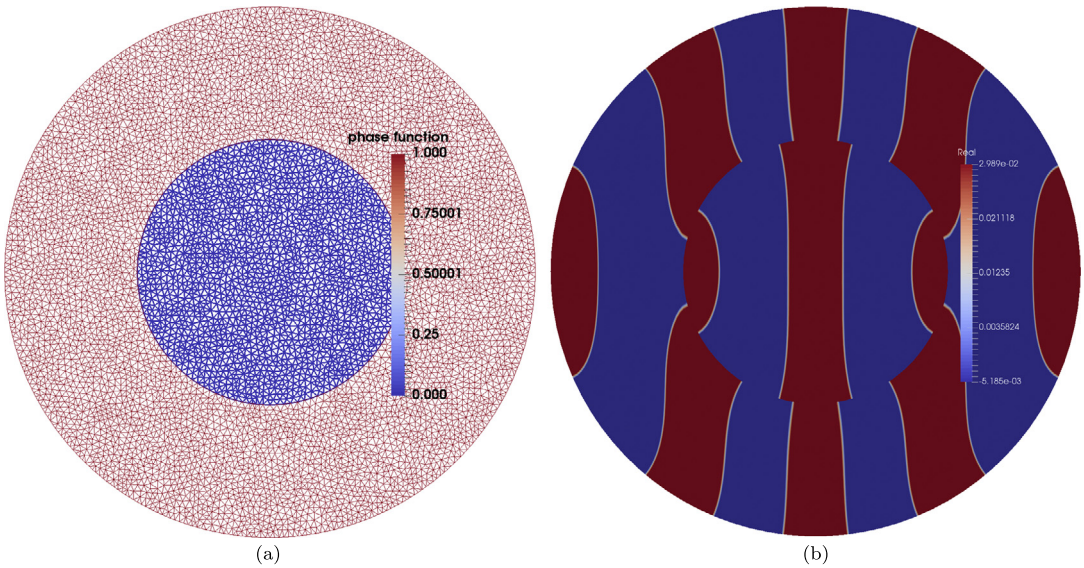
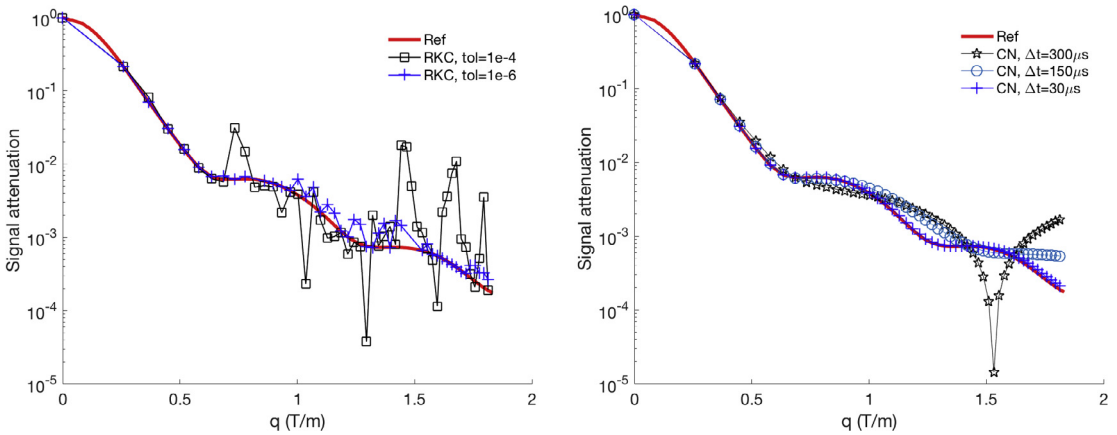


Fig. 6. Phase function with triangulation (a) and a solution at $t = 3000\,\mu s, q = 0.82$ T/m for two cylindrical layers of $R_0 = 10\,\mu m, R_1 = 20\,\mu m$ for $\delta = 5000\,\mu s$, $\Delta = 10{,}000\,\mu s$.



(a) Signals with the RKC method (b) Signals with the CN method

Fig. 7. Signals on two cylindrical layers of $R_0 = 10\,\mu m$, $R_1 = 20\,\mu m$ for $\delta = 5000\,\mu s$, $\Delta = 10{,}000\,\mu s$. For low gradient strength ($q < 1\,T/m$), both methods give good approximation. The RKC start oscillating with $q > 0.5\,T/m$ for `rtol=1e-4` and the approximation is still good up to $q = 1.83\,T/m$ for `rtol=1e-6` although some oscillations start appearing when $q > 1\,T/m$ (a). The CN method appears to be much more stable. It works with quite large time-step size. Especially, at $\Delta t = 30\,\mu s$, the numerical signal matches very well the reference solution (b).

permeability and impermeability. The exact signal for the first case is well known as diffusion in the free space with $S = e^{-b\,D}$. In the latter, the reference signal is taken from the matrix formalism [24]. The difference between the two reference signals reflects how much the spins interact with or see the external boundaries. If the two curves are close to each other, i.e., there is no significant interaction between the spins and the boundaries, we can neglect the effect of the boundaries and the choice of boundary conditions is less important. Here we chose long diffusion times such that the spins clearly see the boundaries, i.e. the two reference signal curves are quite different.

Fig. 8a shows the spatial convergence of the signals to the full permeability for $\delta = \Delta = 10{,}000\,\mu s$. The second order convergence is obtained at $b = 3000\,s/mm^2$ (Fig. 8b). Here the time step size is set to $\Delta t = 10\,\mu s$ and the mesh size varies between $0.2\,\mu m$ and $1.4\,\mu m$. The temporal convergence of the computed signals to the full permeability case for the finest mesh is shown in Fig. 8c. The convergence rate is almost quadratic for the three last b-values (Fig. 8d).

In reality, the computational domain is not free and it can include some cells. The hindrance of the cell membrane reduces the probability that water molecules see the external boundaries. This would reduce the effect of the boundaries and increase the accuracy of the approximation. To illustrate this phenomenon we consider a 3D domain $\Omega = [-10\,\mu m, 10\,\mu m]^3$ which consists of periodic cells of radius $R = 9\,\mu m$ in x-direction (Fig. 9a).

We impose a permeability of $\kappa = 10^{-5}\,m/s$ between the cell and the extracellular space. Fig. 9c shows a comparison between signals computed by the master-slave implementation on a periodic mesh as in [11] and those computed by using an artificial permeability $\kappa^e = \frac{D}{h}$ on the non-periodic mesh (Fig. 9b). Although the diffusion time is long, i.e. $\delta = \Delta = 50{,}000\,\mu s$, two curves are close to each other for quite large time-step size $\Delta t = 500\,\mu s$.

7.4. Multi-layer structures

An accurate method to calculate the NMR signals in some simple multilayered structures such as multiple slabs, cylindrical or spherical shells was proposed in [24]. Unfortunately, it is difficult to extend the method to general geometries. In this section, we show that our method can be a good alternative for more complex geometries.

We first perform simulations on a three-layered sphere of radii $R = [5, 7.5, 10]\,\mu m$ (Fig. 10a) for a PGSE with $\Delta = 50\,\delta = 50{,}000\,\mu s$. Fig. 10b shows signals for different time-step sizes, $\Delta t = 510, 102,$ and $51\,\mu s$, in comparison with the reference solution proposed in [24]. We see that the signals computed by our method converge to the reference signals. At $\Delta t = 51\,\mu s$, they match very well to the reference solution for a wide range of gradient strengths.

More generally, we show in Fig. 11d the signals for a three-layered torus (Figs. 11a and 11b) for which we do not have reference signals for an arbitrary gradient direction. Still, it is possible to verify the accuracy of our method through some special cases, for instance $\frac{\mathbf{g}}{\|\mathbf{g}\|} = [0, 0, 1]$ in which the problem is simplified to the diffusion inside a three circular layers (Fig. 11c) and the reference signal is available. The radius from the center of the hole to the center of the torus tube is $R = 20\,\mu m$ and the tube consists of three layers with radii of 5, 7.5 and $10\,\mu m$ respectively. The simulations were performed for two gradient directions $\frac{\mathbf{g}}{\|\mathbf{g}\|} = [1, 0, 0]$ and $[0, 0, 1]$. The first direction is computed with two temporal discretizations $\Delta t = 51\,\mu s$ and $25.5\,\mu s$ which give quite close approximations. The signals for the second direction are computed with $\Delta t = 25.5\,\mu s$. The computed signals approximate well the reference signals of the three circular layers.

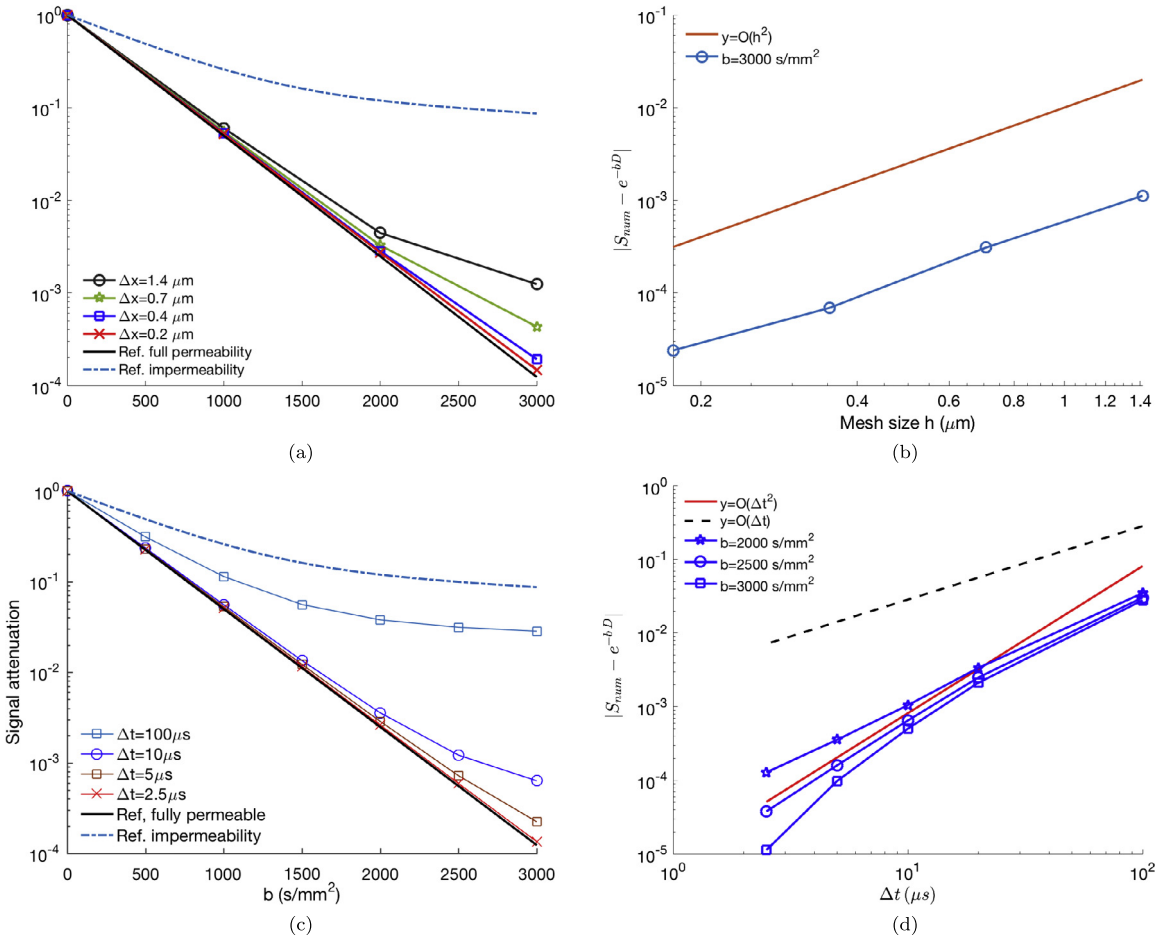


Fig. 8. Spatial convergence (a, b) and temporal convergence (c, d) of the signals with $\kappa^e = \frac{D}{h}$ to the full permeability in a free space for b between 0 and 3000 s/mm^2. The convergence is almost quadratic in both time and space.

7.5. Speedup ratio and parallel efficiency

In this section we clarify the timing, speedup ratio and parallel efficiency on the Beskow super computer at KTH (https://www.pdc.kth.se/resources/computers/beskow) by simulations on the three-layer torus described in Fig. 11a. The simulations were performed for $b = 1000\,\text{s/mm}^2$ and a PGSE with $\delta = \Delta = 10{,}000\,\mu\text{s}$. The linear system has 3,512,160 dofs for which the difference between the reference signal and simulated signal is about 0.4%. We varied the number of MPI processes between 2 and 256.

The speedup ratio is the ratio between timing for serial execution T_{serial} and timing for parallel execution $T_{parallel}$, i.e.

$$\mathcal{S}(p) = \frac{T_{serial}}{T_{parallel}(p)} \tag{33}$$

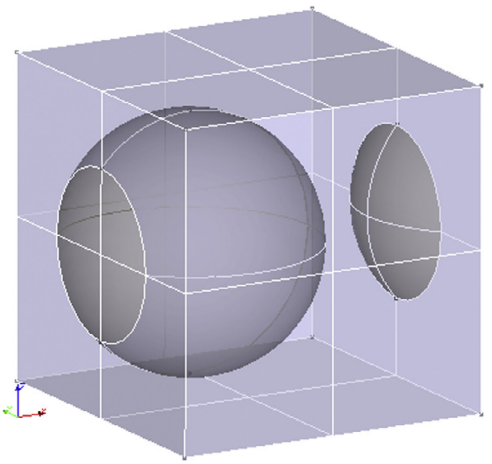
where p is the number of MPI processes used in the parallel execution.

The ideal speedup is $\mathcal{S}_{ideal}(p) = p$, i.e. when p MPI processes are used, the parallel execution will be p times faster than the serial execution.

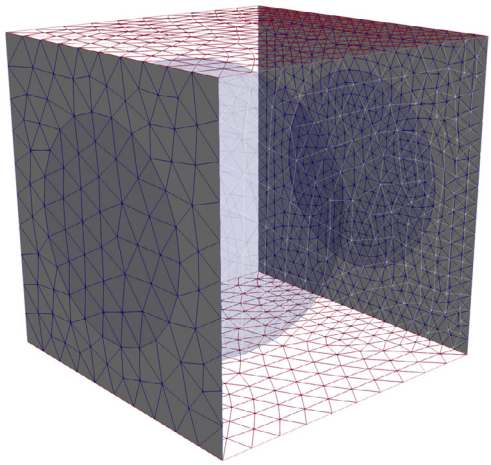
The parallel efficiency is computed by

$$\mathcal{E} = \frac{\mathcal{S}}{p} \tag{34}$$

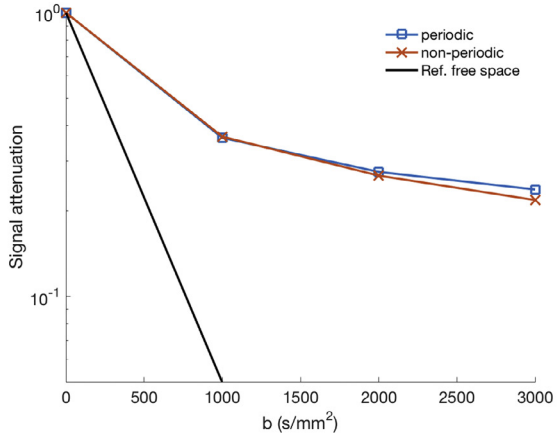
The serial computation costs about 40 hours and the parallel computation with 256 MPI processes only costs 17 minutes (Fig. 12a). The speedup ratio is nearly optimal (Fig. 12b) and the efficiency is more than 40% (Fig. 12c).



(a) Periodic cells in $x-$direction

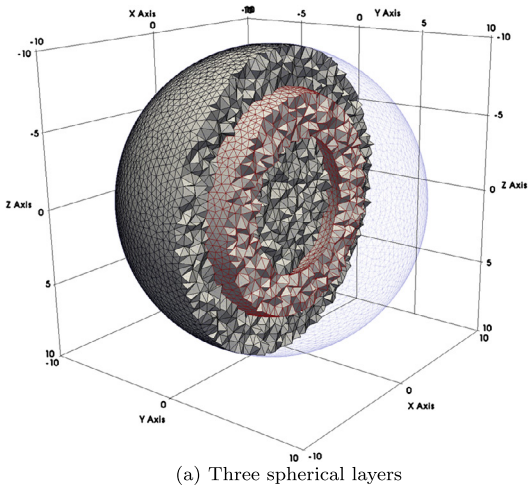


(b) A non-periodic mesh

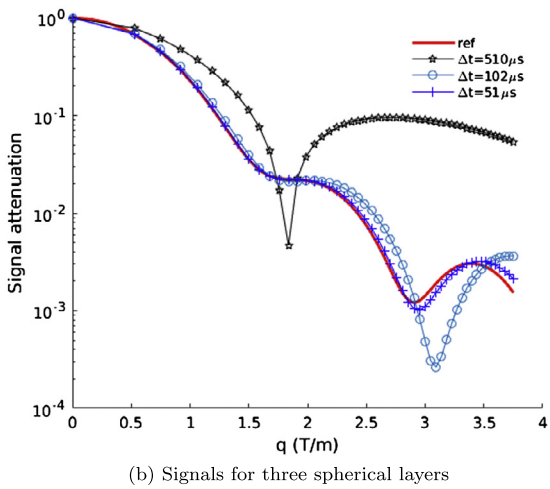


(c) A comparison in signals

Fig. 9. A 3D domain $\Omega = [-10\,\mu m, 10\,\mu m]^3$ consists of periodic cells of radius $R = 9\,\mu m$ in x-direction (a). The signals are computed on a non-periodic mesh (b) and compared with the standard master-slave approach on a periodic mesh. The difference between two signal curves for $\delta = \Delta = 50{,}000\,\mu s$ and b between 0 and $3000\,s/mm^2$ is less than 10% (c).



(a) Three spherical layers



(b) Signals for three spherical layers

Fig. 10. The signals for different time-step sizes, $\Delta t = 510, 102,$ and $51\,\mu s$, in comparison with the reference solution proposed in [24] (b) with $\Delta = 50\,\delta = 50{,}000\,\mu s$ for three-layer structures of radii $R = [5, 7.5, 10]\,\mu m$ (a).

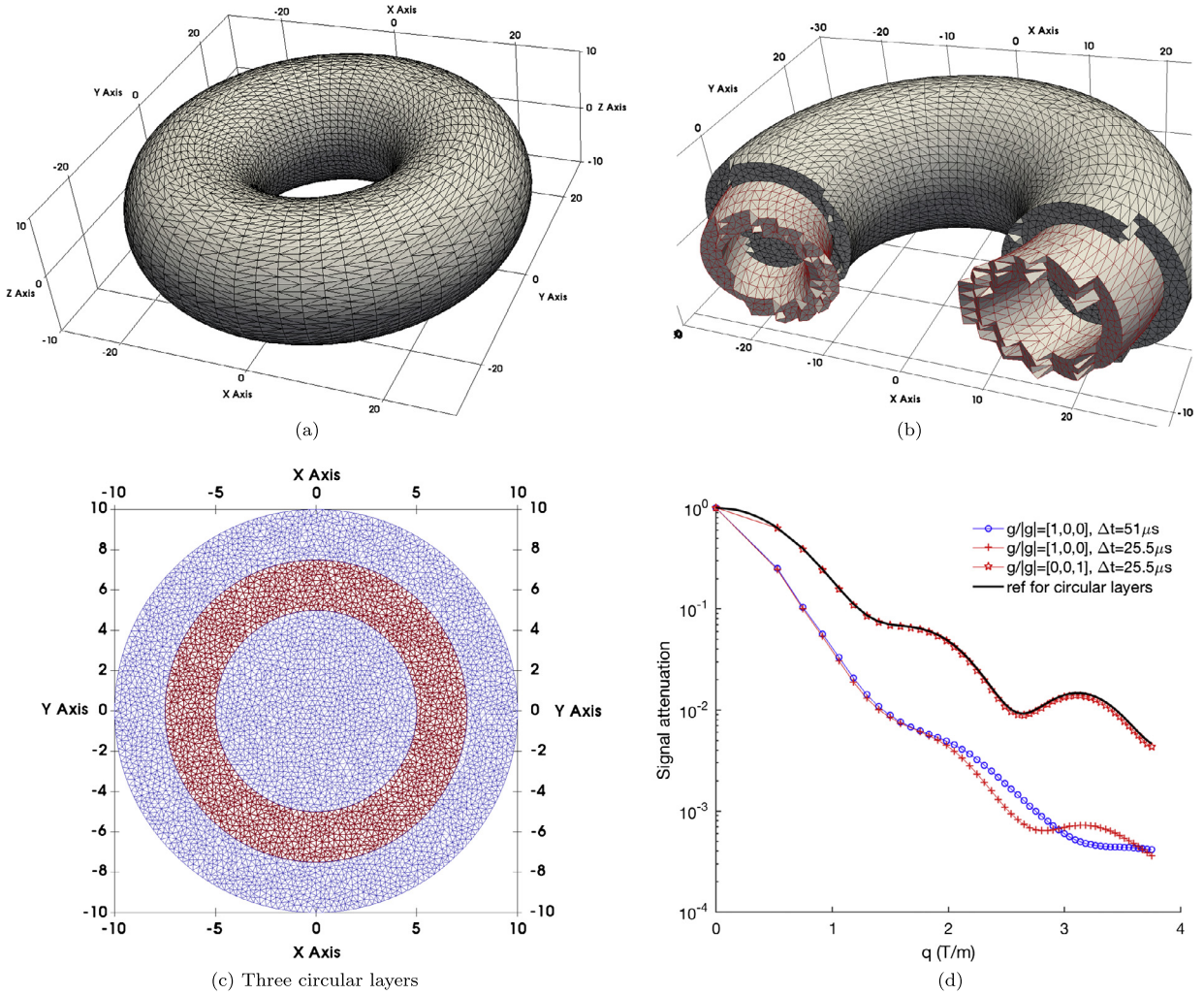


Fig. 11. Signals (d) computed in a torus (a) consisting of three layers (b). The radius from the center of the hole to the center of the torus tube is $R = 20\,\mu\text{m}$ and the radii of three layers are 5, 7.5 and 10 μm respectively. Two gradient directions $\frac{\mathbf{g}}{\|\mathbf{g}\|} = [1, 0, 0]$ and $[0, 0, 1]$ are considered. The second direction matches well the signal in three circular layers (c).

7.6. Simulations on a pyramidal neuron

In this section, we perform simulations on a pyramidal neuron of an adult female mouse (Fig. 13a) [38] to study how the signals decay under the effect of the permeable membrane. It also shows that the method can be used to simulate diffusion MRI for quite complex geometries.

The morphology file (.swc) corresponding to the neuron was downloaded from http://neuromorpho.org. The neuron is made of a soma with the surface area of 1117.83 μm^2 and small dendrites with the total length of 5953.41 μm and the average diameter of 1.2 μm. The neuron is embedded in the center of a computational domain $\Omega = [-250, 250] \times [-250, 250] \times [-100, 100]\,\mu\text{m}^3$ (Fig. 13b). The surface and volume meshes were generated with ANSA from Beta-CAE Systems https://www.beta-cae.com. The computational domain has approximately 8.5M tetrahedrons, 1.5M vertices. The neuron itself needs small elements to describe accurately small dendrites (see Fig. 13c) and its mesh consists of 131,996 vertices and 431,326 tetrahedrons.

The simulations were also carried out on the Beskow supercomputer for three principle gradient directions $\frac{\mathbf{g}}{\|\mathbf{g}\|} = [1, 0, 0], [0, 1, 0]$ and $[0, 0, 1]$ and six b-values between 0 and 5000 s/mm^2. The computational domain was filled everywhere with water by the use of uniform initial conditions. A PGSE sequence with $\delta = 10{,}000\,\mu\text{s}$ and $\Delta = 50{,}000\,\mu\text{s}$ was used with a time step size of $\Delta t = 60\,\mu\text{s}$ for the temporal discretization. The average timing is about 20 minutes per b-value with 320 MPI processes.

Fig. 13d shows the mean signals over the three gradient directions for different permeabilities. The vertical segment at each marker indicates the signal variation. The signals decay faster for higher membrane permeabilities. At $\kappa = 0.5\,\text{m/s}$, the signal is very close to that of the fully permeable membrane which is $S = \text{e}^{-b\,D}$. Here the diffusion coefficient of

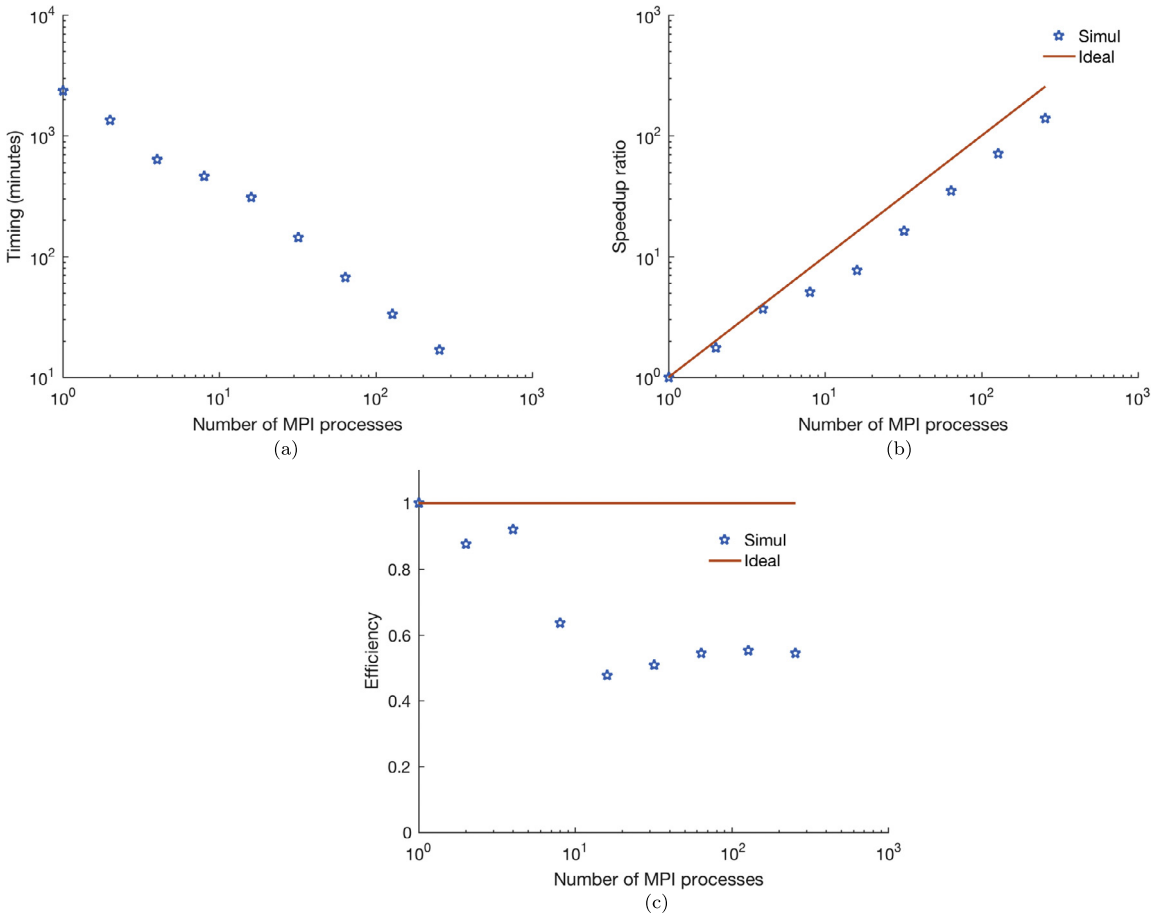


Fig. 12. Timing, speedup ratio and parallel efficiency on the Beskow. The simulations were performed for the three-layer torus described in Fig. 11a. The simulations were performed for $b = 1000\,\mathrm{s/mm^2}$ and the diffusion time $\delta = \Delta = 10000\,\mu\mathrm{s}$. The linear system has 3'512'160 dofs for which the difference between the reference signal and simulated signal is about 0.4%. We varied the number of MPI processes between 2 and 256.

water $D = 2.4 \times 10^{-3}\,\mathrm{mm^2/s}$ was used for both inside the neuron and the extra-cellular space. For the impermeable case ($\kappa = 0$), the signals decay nearly 80% at $b = 5000\,\mathrm{s/mm^2}$. For the commonly used permeability ($\kappa = 5 \times 10^{-5}\,\mathrm{m/s}$), the signals decay quite a lot even for small b-values. The signal decay is about 75% for $b = 1000\,\mathrm{s/mm^2}$ and 95% for $b = 5000\,\mathrm{s/mm^2}$.

8. Conclusions and future works

The paper presents a partition of unity finite element method for the two-compartment Bloch–Torrey equation that allows for imposing the permeability interface conditions in the weak form with basis functions of arbitrary order. The temporal scheme is unconditionally stable by the use of the Crank–Nicolson method. The accuracy of the method is validated for multilayered structures where the signals have a good agreement with the reference ones. We also show numerically that artificial jump conditions at the external boundaries can be used to approximate the pseudo-periodic boundary conditions. This technique allows the water exchange at the external boundaries for non-periodic meshes. The framework is of a high level simplicity and efficiency that facilitates parallelization. The proposed method can be straightforwardly implemented in different FEM software packages and it is implemented in FEniCS for moderate-scale simulations and in FEniCS-HPC for the large-scale simulations. The simulations on a pyramidal neuron show that the method can be used to simulate diffusion MRI on more complex geometries than have been done before. More realistic applications are under consideration to uncover microstructure information of biological tissues.

Acknowledgement

This research has been supported by the Swedish Energy Agency, Sweden with the project ID P40435-1; MSO4SC (Spain) with the grant number 731063; the Basque Excellence Research Center (BERC 2014-2017) program by the Basque Government; the Spanish Ministry of Economy and Competitiveness MINECO: BCAM Severo Ochoa accreditation SEV-2013-0323;

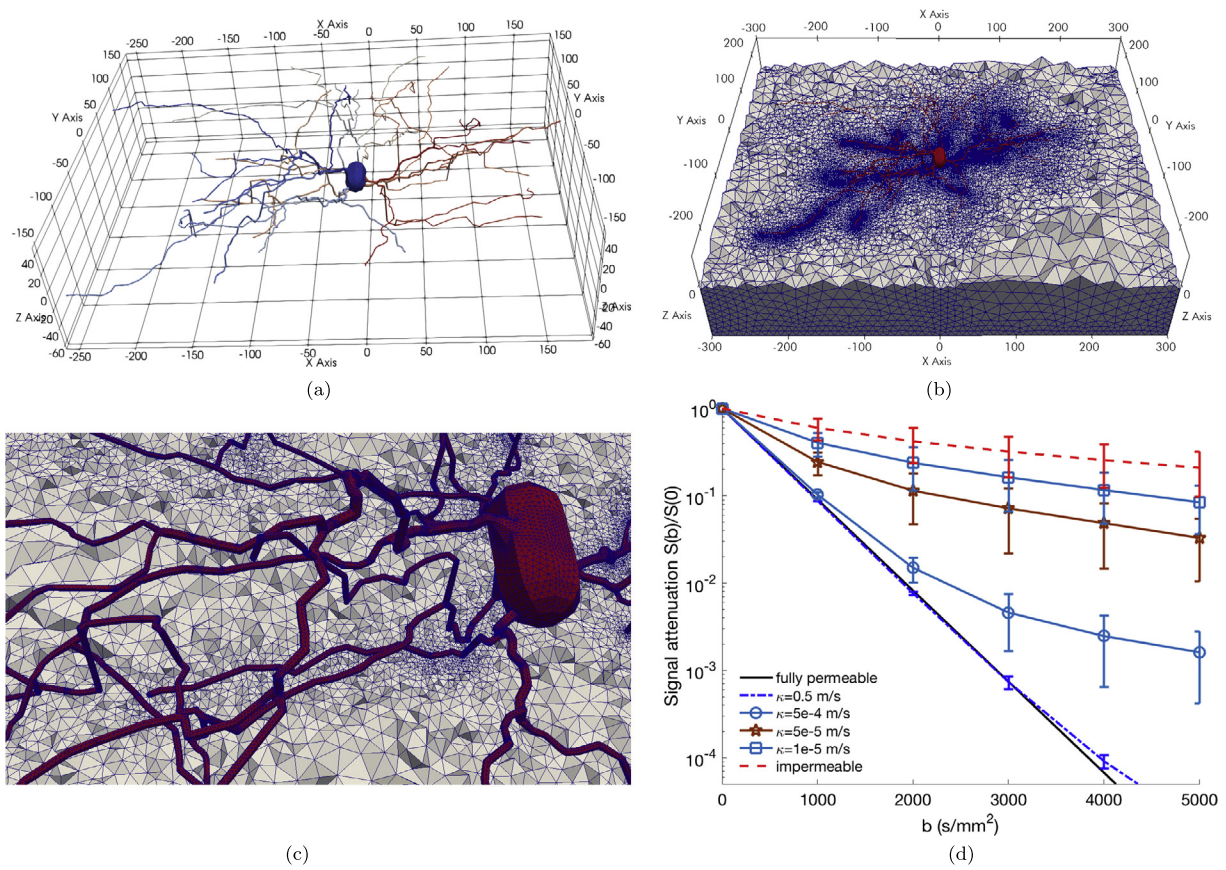


Fig. 13. A pyramidal neuron of an adult female mouse (a) [38] downloaded from http://neuromorpho.org in which the soma surface area is 1117.83 μm^2 and the dendrites have the total length of 5953.41 μm and the average diameter of 1.2 μm. The neuron is embedded in the center of a computational domain $\Omega = [-250, 250] \times [-250, 250] \times [-100, 100]\mu m^3$ (b) which was meshed with 1.5 M vertices and 8.5 M tetrahedrons. The neuron itself needs small elements to describe accurately small dendrites (a, c) and its mesh consists of 131,996 vertices and 431,326 tetrahedrons. The signals were averaged over the three principal directions for different membrane permeabilities (d). The vertical segment at each marker indicates the signal variation. The signals decay faster for higher membrane permeabilities.

the ICERMAR ELKARTEK project of the Basque Government; the projects of the Spanish Ministry of Economy and Competitiveness with reference MTM2013-40824-P and MTM2016-76016-R. The simulations were performed on resources provided by the SNIC. The first author would like to thank Niyazi Cem Degirmenci for his enthusiastic help with the FEniCS-HPC platform.

Appendix A. Projection technique versus `ident_zeros` technique

Here we numerically compare between the projection technique Eq. (13) and the `ident_zeros` technique in terms of convergence rates and the condition numbers.

Fig. A.14 shows the results for the L_2-projection problem of $p(x, y)$ on a continuous piecewise linear function space defined on $\Omega = [0, 4] \times [-0.4, 0.6]$. The domain is meshed with right-angled triangles. Here,

$$p(x, y) = \begin{cases} \sin(x)\, x + y & (x, y) \in [0, 4] \times [-0.4, 0] \\ -\cos(y)\, x - y & (x, y) \in [0, 4] \times [0, 0.6] \end{cases} \tag{A.1}$$

Although the condition numbers of the projection technique always reach $O(1)$ for all β (Fig. A.14b), only $\beta = 1e{-}3$ gives optimal convergence rate (Fig. A.14a). Its condition numbers are still larger than those of the `ident_zeros` technique which has $O(h^{-2})$ (Fig. A.14b).

Fig. A.15 shows the results for the problem presented in 7.1 with linear elements. Although both give optimal condition number estimates (Fig. A.15b), the `ident_zeros` technique gives much smaller condition numbers compared to those with $\beta = 1e{-}1$ to obtain the optimal convergence rate (Fig. A.15a).

In short, the above results suggest that the `ident_zeros` technique is more numerically efficient than the projection technique.

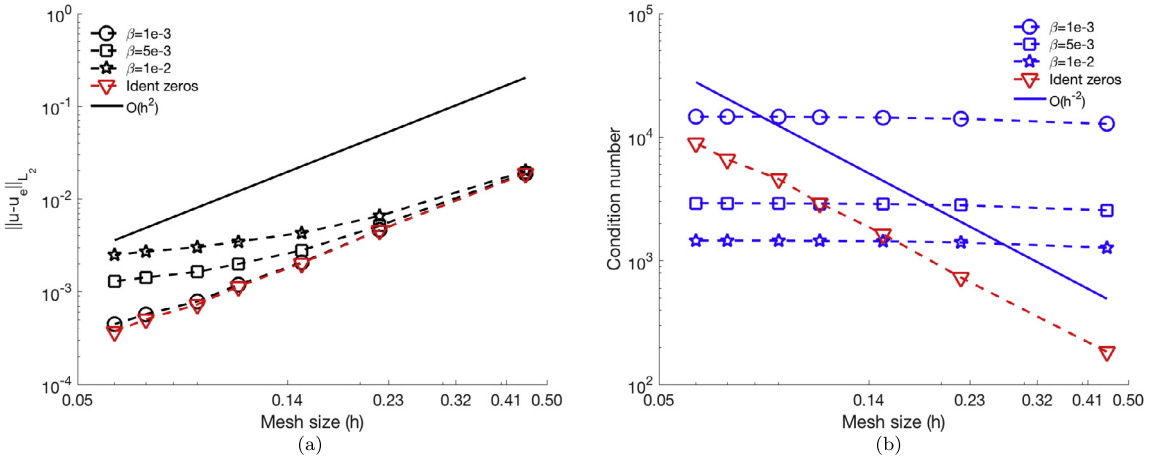


Fig. A.14. The convergence rate (a) and the condition number (b) for different values of the stabilization parameter β in Eq. (13) compared to the `ident_zeros` technique for the L_2-projection problem of $p(x, y)$ on a continuous piecewise linear function space defined on $\Omega = [0, 4] \times [-0.4, 0.6]$. The domain is meshed by right-angled triangles. Although the `ident_zeros` technique violates the optimal condition number estimates, its estimates are still smaller than those of $\beta = 1\text{e}{-3}$ to obtain the optimal convergence rate.

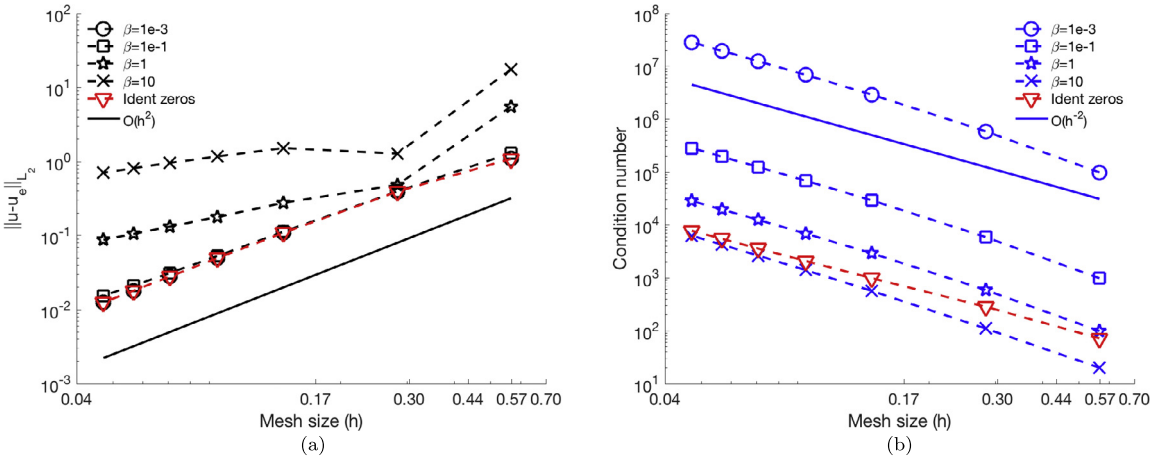


Fig. A.15. The convergence rate (a) and the condition number (b) for different values of the stabilization parameter β in Eq. (13) compared to the `ident_zeros` technique for the problem presented in 7.1. The `ident_zeros` technique maintains better the optimal convergence rate and the optimal condition number estimates.

Appendix B. Jump conditions at the external boundaries

The idea comes from a combination of the interface conditions between two compartments and the pseudo-periodic conditions for the exterior boundaries. Since the computational domain Ω is extended periodically, in some cases the cell interfaces touch the exterior boundary $\partial\Omega$. To simplify the explanation, we consider a computational domain $\Omega = [a_1, b_1] \times [a_2, b_2]$ in which a cell is placed touching $x = a_1$ (Fig. B.16). The boundary needs to be periodic at $x = a_1$ and $x = b_1$. The cell interior is characterized by a diffusion tensor $\boldsymbol{D}^c$. The extra-cellular space is the remaining part and is characterized by a diffusion tensor $\boldsymbol{D}^e$. The cell touches the boundary at the interface Σ, which is the intersection between the cell boundary and $\partial\Omega$. We recall the pseudo-periodic boundary conditions for this specific domain

$$\begin{aligned} u^e_{a_1} &= u^e_{b_1} e^{\mathcal{I}\theta}, \\ \boldsymbol{D}^e_{a_1} \nabla u^e_{a_1} \cdot \boldsymbol{n}_c &= -\boldsymbol{D}^e_{b_1} \nabla u^e_{b_1} \cdot \boldsymbol{n}_e e^{\mathcal{I}\theta}, \end{aligned} \tag{B.1}$$

where

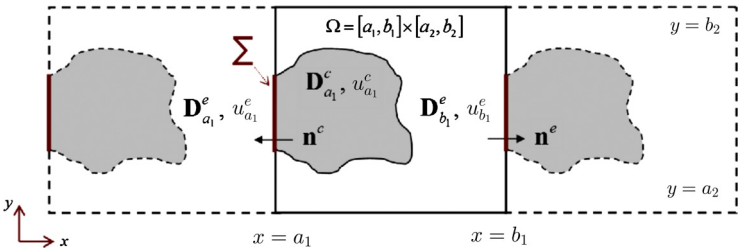


Fig. B.16. When the cell interface touches $\partial\Omega$, the interface conditions and periodic boundary conditions are combined.

$$\theta = \gamma\, g_1\, (b_1 - a_1)\mathcal{F}(t),$$

and the interface conditions at Σ are

$$\begin{aligned} \boldsymbol{D}^c_{a_1} \nabla u^c_{a_1} \cdot \boldsymbol{n}_c &= \kappa^e (u^e_{a_1} - u^c_{a_1}) \\ \boldsymbol{D}^e_{a_1} \nabla u^e_{a_1} \cdot \boldsymbol{n}_e &= \kappa^e (u^c_{a_1} - u^e_{a_1}) \end{aligned} \tag{B.2}$$

The combination of (B.1) and (B.2) with $\mathbf{n}^c = -\mathbf{n}^e$ gives

$$\begin{aligned} \boldsymbol{D}^c_{a_1} \nabla u^c_{a_1} \cdot \boldsymbol{n}_c &= \kappa^e \left(u^e_{b_1}\, e^{\mathcal{I}\theta} - u^c_{a_1} \right), \\ \boldsymbol{D}^e_{b_1} \nabla u^e_{b_1} \cdot \boldsymbol{n}_e &= \kappa^e \left(u^c_{a_1}\, e^{-\mathcal{I}\theta} - u^e_{b_1} \right). \end{aligned} \tag{B.3}$$

A general version of this equation using the master-salve notation is

$$\begin{aligned} \boldsymbol{D}_m \nabla u_m \cdot \boldsymbol{n}_m &= \kappa^e \left(u_s\, e^{\mathcal{I}\theta_{ms}} - u_m \right), \\ \boldsymbol{D}_s \nabla u_s \cdot \boldsymbol{n}_s &= \kappa^e \left(u_m\, e^{\mathcal{I}\theta_{sm}} - u_s \right), \end{aligned} \tag{B.4}$$

where κ^e reflects how much water can exchange at the boundaries and

$$\theta_{ms} = -\theta_{sm} = \gamma\, \boldsymbol{g} \cdot (\boldsymbol{x}_s - \boldsymbol{x}_m)\, \mathcal{F}(t).$$

Appendix C. FEniCS code for the space-convergence test problem

```
V = VectorFunctionSpace(mesh , "CG", porder);

u = TrialFunction(V);
v = TestFunction(V);

a = (inner(grad(u[0]),grad(v[0]))+u[0]*v[0])*(1-Phi)*dx \
  +(inner(grad(u[1]),grad(v[1]))+u[1]*v[1])*Phi*dx

# For the source p
L = p0*v[0]*(1-Phi)*dx + p1*v[1]*Phi*dx;

# For the flux
L += ( g_j*0.5*avg(v[0]+v[1])+g_a*avg(v[0]-v[1]) )*abs(jump(Phi))*dS
```

Appendix D. FEniCS code for the θ-method

```
GX=Expression("x[0]*g0+x[1]*g1", g0=g0, g1=g1,domain=mesh,degree=3);
def FuncF(ft, gnorm, GX, ur, ui, vr, vi, K):
   Fr = ft*gnorm*GX*ui*vr - K*inner(grad(ur), grad(vr))
   Fi = - ft*gnorm*GX*ur*vi - K*inner(grad(ui), grad(vi))
   return Fr + Fi

def interface_cond(kappa, u0rm, u1rm, v0r, v1r, u0im, u1im, v0i, v1i):
   F_bcr = kappa*(u0rm-u1rm)*(v0r-v1r)
   F_bci = kappa*(u0im-u1im)*(v0i-v1i)
   return F_bcr + F_bci

def ThetaMethod(ft, gnorm, GX, u0r, u0i, v0r, v0i, u1r, u1i, v1r, v1i, u0r_0, u0i_0, u1r_0,
    u1i_0,k, K, theta, Phi):
   a0 = (u0r/k*v0r + u0i/k*v0i -theta*FuncF(ft, gnorm, GX, u0r , u0i , v0r, v0i, K))*(1-Phi)*dx
   a1 = (u1r/k*v1r +u1i/k*v1i -theta*FuncF(ft, gnorm, GX, u1r , u1i , v1r, v1i, K))*Phi*dx
   L0 = (u0r_0/k*v0r + u0i_0/k*v0i+(1-theta)*FuncF(ft, gnorm, GX, u0r_0, u0i_0, v0r, v0i,
       K))*(1-Phi)*dx
   L1 = (u1r_0/k*v1r +u1i_0/k*v1i+(1-theta)*FuncF(ft, gnorm, GX, u1r_0, u1i_0, v1r, v1i,
       K))*Phi*dx

   a_bc = avg( (theta*icondition(kappa, u0r , u1r , v0r, v1r, u0i , u1i , v0i,
       v1i)))*abs(jump(Phi))*dS;
   L_bc = avg((1-theta)*icondition(kappa, u0r_0, u1r_0, v0r, v1r, u0i_0, u1i_0, v0i,
       v1i))*abs(jump(Phi))*dS;

   return a0+a1+a_bc, L0+L1-L_bc
```

Listing 1: Bilinear and linear forms.

```
def NoTimeMatrices(u0r, u0i, v0r, v0i, u1r, u1i, v1r, v1i, K, GX, kappa, theta, phase):
   m0 = (u0r*v0r + u0i*v0i)*(1-phase)*dx
   m1 = (u1r*v1r + u1i*v1i)*phase*dx
   M = assemble(m0+m1);

   j0 = -GX*(u0i*v0r - u0r*v0i)*(1-phase)*dx
   j1 = -GX*(u1i*v1r - u1r*v1i)*phase*dx
   J = assemble(j0+j1);
   s0 = K*( inner(grad(u0r), grad(v0r)) + inner(grad(u0i), grad(v0i)) )*(1-phase)*dx
   s1 = K*( inner(grad(u1r), grad(v1r)) + inner(grad(u1i), grad(v1i)) )*phase*dx
   S = assemble(s0+s1)

   im = avg( icondition(kappa, u0r , u1r , v0r, v1r, u0i , u1i , v0i, v1i) )*abs(jump(phase))*dS;
   I = assemble(im)

   M.ident_zeros();

   return M, J, S, I
```

Listing 2: Time independent matrices to avoid repeating assembling process.

References

[1] H.C. Torrey, Bloch equations with diffusion terms, Phys. Rev. 104 (1956) 563–565, https://doi.org/10.1103/PhysRev.104.563.

[2] D. Topgaard, Multidimensional diffusion MRI, J. Magn. Reson. 275 (2017) 98–113, https://doi.org/10.1016/j.jmr.2016.12.007, http://www.sciencedirect.com/science/article/pii/S1090780716302701.

[3] E.O. Stejskal, J.E. Tanner, Spin diffusion measurements: spin echoes in the presence of a time-dependent field gradient, J. Chem. Phys. 42 (1) (1965) 288–292, https://doi.org/10.1063/1.1695690.

[4] J.E. Tanner, Transient diffusion in a system partitioned by permeable barriers. Application to NMR measurements with a pulsed field gradient, J. Chem. Phys. 69 (4) (1978) 1748–1754, https://doi.org/10.1063/1.436751.

[5] J. Xu, M. Does, J. Gore, Numerical study of water diffusion in biological tissues using an improved finite difference method, Phys. Med. Biol. 52 (7) (2007), http://view.ncbi.nlm.nih.gov/pubmed/17374905.

[6] Z. Yuan, J. Fish, Toward realization of computational homogenization in practice, Int. J. Numer. Methods Eng. 73 (2008) 361–380, https://doi.org/10.1002/nme.2074.
[7] H. Hagslatt, B. Jonsson, M. Nyden, O. Soderman, Predictions of pulsed field gradient NMR echo-decays for molecules diffusing in various restrictive geometries. Simulations of diffusion propagators based on a finite element method, J. Magn. Reson. 161 (2) (2003) 138–147, http://www.sciencedirect.com/science/article/pii/S1090780702000393.
[8] N. Loren, H. Hagslatt, M. Nyden, A.-M. Hermansson, Water mobility in heterogeneous emulsions determined by a new combination of confocal laser scanning microscopy, image analysis, nuclear magnetic resonance diffusometry, and finite element method simulation, J. Chem. Phys. 122 (2) (2005) 024716, https://doi.org/10.1063/1.1830432.
[9] B.F. Moroney, T. Stait-Gardner, B. Ghadirian, N.N. Yadav, W.S. Price, Numerical analysis of NMR diffusion measurements in the short gradient pulse limit, J. Magn. Reson. 234 (2013) 165–175, http://www.sciencedirect.com/science/article/pii/S1090780713001572.
[10] J.-R. Li, D. Calhoun, C. Poupon, D.L. Bihan, Numerical simulation of diffusion MRI signals using an adaptive time-stepping method, Phys. Med. Biol. 59 (2) (2014) 441, http://stacks.iop.org/0031-9155/59/i=2/a=441.
[11] 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, J. Comput. Phys. 263 (Supplement C) (2014) 283–302, https://doi.org/10.1016/j.jcp.2014.01.009, http://www.sciencedirect.com/science/article/pii/S0021999114000308.
[12] L. Beltrachini, Z.A. Taylor, A.F. Frangi, A parametric finite element solution of the generalised Bloch–Torrey equation for arbitrary domains, J. Magn. Reson. 259 (Supplement C) (2015) 126–134, https://doi.org/10.1016/j.jmr.2015.08.008, http://www.sciencedirect.com/science/article/pii/S1090780715001743.
[13] G. Russell, K.D. Harkins, T.W. Secomb, J.-P. Galons, T.P. Trouard, A finite difference method with periodic boundary conditions for simulations of diffusion-weighted magnetic resonance experiments in tissue, Phys. Med. Biol. 57 (4) (2012) N35, http://stacks.iop.org/0031-9155/57/i=4/a=N35.
[14] J. Chung, G.M. Hulbert, A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method, J. Appl. Mech. 60 (2) (1993) 371–375, https://doi.org/10.1115/1.2900803.
[15] J.G. Verwer, W.H. Hundsdorfer, B.P. Sommeijer, Convergence properties of the Runge–Kutta–Chebyshev method, Numer. Math. 57 (1) (1990) 157–178, https://doi.org/10.1007/BF01386405.
[16] F. Larsson, K. Runesson, S. Saroukhani, R. Vafadari, Computational homogenization based on a weak format of micro-periodicity for rve-problems, Comput. Methods Appl. Mech. Eng. 200 (1) (2011) 11–26, https://doi.org/10.1016/j.cma.2010.06.023, http://www.sciencedirect.com/science/article/pii/S0045782510001908.
[17] V.-D. Nguyen, E. Béchet, C. Geuzaine, L. Noels, Imposing periodic boundary condition on arbitrary meshes by polynomial interpolation, Comput. Mater. Sci. 55 (2012) 390–406.
[18] C. Sandstöm, F. Larsson, K. Runesson, Weakly periodic boundary conditions for the homogenization of flow in porous media, Adv. Model. Simul. Eng. Sci. 1 (1) (2014) 12, https://doi.org/10.1186/s40323-014-0012-6.
[19] V.D. Nguyen, in: A FEniCS-HPC Framework for Multi-Compartment Bloch–Torrey Models, vol. 1, 2016, pp. 105–119, QC 20170509, https://www.eccomas2016.org/.
[20] J. Melenk, I. Babuška, The partition of unity finite element method: basic theory and applications, Comput. Methods Appl. Mech. Eng. 139 (1) (1996) 289–314, https://doi.org/10.1016/S0045-7825(96)01087-0.
[21] E. Wadbro, S. Zahedi, G. Kreiss, M. Berggren, A uniformly well-conditioned, unfitted Nitsche method for interface problems, BIT Numer. Math. 53 (3) (2013) 791–820, https://doi.org/10.1007/s10543-012-0417-x.
[22] P. Hansbo, M.G. Larson, S. Zahedi, A cut finite element method for a stokes interface problem, Appl. Numer. Math. 85 (Supplement C) (2014) 90–114, https://doi.org/10.1016/j.apnum.2014.06.009.
[23] B. Sommeijer, L. Shampine, J. Verwer, Rkc: an explicit solver for parabolic pdes, J. Comput. Appl. Math. 88 (2) (1998) 315–326, https://doi.org/10.1016/S0377-0427(97)00219-7.
[24] D.S. Grebenkov, Pulsed-gradient spin-echo monitoring of restricted diffusion in multilayered structures, J. Magn. Reson. 205 (2) (2010) 181–195, https://doi.org/10.1016/j.jmr.2010.04.017, http://www.sciencedirect.com/science/article/pii/S1090780710001199.
[25] A. Walter, J. Gutknecht, Permeability of small nonelectrolytes through lipid bilayer membranes, J. Membr. Biol. 90 (3) (1986) 207–217, https://doi.org/10.1007/BF01870127.
[26] J. Nitsche, Über ein variationsprinzip zur lösung von Dirichlet-problemen bei verwendung von teilräumen, die keinen randbedingungen unterworfen sind, Abh. Math. Semin. Univ. Hamb. 36 (1) (1971) 9–15, https://doi.org/10.1007/BF02995904.
[27] P. Hansbo, Nitsche's method for interface problems in computational mechanics, GAMM-Mitt. 28 (2) (2005) 183–206, https://doi.org/10.1002/gamm.201490018.
[28] A. Logg, K.-A. Mardal, G.N. Wells, Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book, Springer Verlag, 2012, xIII, 723, s.: ill.
[29] FEniCS, Fenics project, http://www.fenicsproject.org.
[30] Fenics-hpc, FEniCS-HPC, http://www.fenics-hpc.org.
[31] N. Jansson, J. Jansson, J. Hoffman, Framework for massively parallel adaptive finite element computational fluid dynamics on tetrahedral meshes, SIAM J. Sci. Comput. 34 (1) (2012) C24–C41.
[32] J. Hoffman, J. Jansson, R.V. de Abreu, N.C. Degirmenci, N. Jansson, K. Müller, M. Nazarov, J.H. Spühler, Unicorn: parallel adaptive finite element simulation of turbulent flow and fluid-structure interaction for deforming domains and complex geometry, Comput. Fluids 80 (10) (2013) 310–319.
[33] J. Hoffman, J. Jansson, C. Degirmenci, N. Jansson, M. Nazarov, Unicorn: A Unified Continuum Mechanics Solver, Springer, 2012, Ch. 18.
[34] N. Jansson, J. Hoffman, J. Jansson, Framework for massively parallel adaptive finite element computational fluid dynamics on tetrahedral meshes, SIAM J. Sci. Comput. 34 (1) (2012) C24–C41.
[35] R.C. Kirby, FIAT: Numerical Construction of Finite Element Basis Functions, Springer, 2012, Ch. 13.
[36] J. Hoffman, J. Jansson, N. Jansson, M. Nazarov, Unicorn: a unified continuum mechanics solver, in: Automated Solutions of Differential Equations by the Finite Element Method, Springer, 2011, http://www.fenicsproject.org/pub/documents/book/.
[37] J. Hoffman, J. Jansson, N. Jansson, C. Johnson, R.V. de Abreu, Turbulent flow and fluid-structure interaction, in: Automated Solutions of Differential Equations by the Finite Element Method, Springer, 2011, http://www.fenicsproject.org/pub/documents/book/.
[38] L. Carim-Todd, K.G. Bath, G. Fulgenzi, S. Yanpallewar, D. Jing, C.A. Barrick, J. Becker, H. Buckley, S.G. Dorsey, F.S. Lee, L. Tessarollo, Endogenous truncated TrkB.T1 receptor regulates neuronal complexity and TrkB kinase receptor function *in vivo*, J. Neurosci. 29 (3) (2009) 678–685, https://doi.org/10.1523/JNEUROSCI.5060-08.2009, http://www.jneurosci.org/content/29/3/678.full.