High Performance Finite Element Methods

with Application to Simulation of Diffusion MRI and Vertical Axis Wind Turbines

VAN DANG NGUYEN

Licentiate Thesis in Computer Science
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology
Stockholm, Sweden 2018
Abstract

The finite element methods (FEM) have been developed over decades, and together with the growth of computer engineering, they become more and more important in solving large-scale problems in science and industry. The objective of this thesis is to develop high-performance finite element methods (HP-FEM), with two main applications in mind: computational diffusion magnetic resonance imaging (MRI), and simulation of the turbulent flow past a vertical axis wind turbine (VAWT).

In the first application, we develop an efficient high-performance finite element framework HP-PUFEM based on a partition of unity finite element method to solve the Bloch-Torrey equation in heterogeneous domains. The proposed framework overcomes the difficulties that the standard approaches have when imposing the microscopic heterogeneity of the biological tissues. We also propose artificial jump conditions at the external boundaries to approximate the pseudo-periodic boundary conditions which allow for water exchange at the external boundaries for non-periodic meshes. The framework is of a high-level simplicity and efficiency that well facilitates parallelization. It can be straightforwardly implemented in different FEM software packages and it is implemented in FEniCS for moderate-scale simulations and in FEniCS-HPC for large-scale simulations. The framework is validated against reference solutions, and implementation shows a strong parallel scalability. Since such a high-performance simulation framework is still missing in the field, it can become a powerful tool to uncover diffusion in complex biological tissues.

In the second application, we develop an ALE-DFS method which combines advanced techniques developed in recent years to simulate turbulence. We apply a General Galerkin (G2) method which is continuous piecewise linear in both time and space, to solve the Navier-Stokes equations for a rotating turbine in an Arbitrary Lagrangian-Eulerian (ALE) framework. This method is enhanced with dual-based a posteriori error control and automated mesh adaptation. Turbulent boundary layers are modeled by a slip boundary condition to avoid a full resolution which is impossible even with the most powerful computers available today. The method is validated against experimental data of parked turbines with good agreements.

The thesis presents contributions in the form of both numerical methods for high-performance computing frameworks and efficient, tested software, published open source as part of the FEniCS-HPC platform.

Keywords: High performance finite element method, computational diffusion MRI, turbulent flow, vertical axis wind turbine.
Acknowledgements

First, I would like to thank my interim supervisor Johan Hoffman and de facto supervisor Johan Jansson for giving me a great chance to work with such an interesting topic.

I would like to thank all collaborators who give vital contributions to this work. I am also very thankful to Niyazi Cem Degirmenci and Jeannette Spühler for their enthusiastic help with the FEniCS-HPC platform.

This research has been supported by the European Research Council, the Swedish Energy Agency, 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, 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. I acknowledge the Swedish National Infrastructure for Computing (SNIC) at PDC – Center for High-Performance Computing for awarding me access to the supercomputer resource Beskow.

Finally, I would like to thank all my friends and my family, especially my wife Bich Loan for her encouragement to help me overcome some difficult periods.

Stockholm, April 23, 2018
Van Dang Nguyen
## Contents

1 Introduction 1
   1.1 Research objectives ............................................... 1
   1.2 Background and state of the art .................................... 2
      1.2.1 Computational diffusion MRI .................................. 2
      1.2.2 Simulations of VAWTs ......................................... 7
      1.2.3 High performance computing ................................... 9
   1.3 Thesis Organization ................................................ 10

2 Method 13
   2.1 The FEniCS-HPC framework ......................................... 13
   2.2 Speedup ratio and parallel efficiency ............................... 14
   2.3 An HP-PUFEM for interface conditions ............................ 14
   2.4 The ALE-DFS method for VAWT solution ........................... 15
      2.4.1 cG(1)cG(1) method ........................................... 15
      2.4.2 Automated adaptivity with do-nothing error control ......... 16

3 Results 19
   3.1 Simulation of diffusion MRI ....................................... 19
   3.2 Validation of parked VAWT ......................................... 21

4 Discussion and Future Works 23
   4.1 Computational diffusion MRI ....................................... 23
   4.2 Simulations of VAWT ................................................ 23

5 Abstracts of included papers 25
   5.1 Paper I ................................................................. 25
   5.2 Paper II ................................................................. 26
5.3 Paper III .............................................. 27

Bibliography ............................................. 29
Chapter 1

Introduction

1.1 Research objectives

We have witnessed a rapid growth of computer science in recent years. It has played a key role in solving a variety of problems ranging from science to industry. The physics behind these applications can often be mathematically described as partial differential equations (PDEs). The complexity of realistic applications means that one needs to solve these equations using numerical methods. Finding numerical solutions of PDEs is sometimes extremely difficult since a huge resolution of the physical phenomenon in both time and space is needed. Many numerical methods have been proposed to solve PDEs where the finite element methods (FEM) have appeared as a general method with solid mathematical principles. These methods have been implemented for standard serial computers since long. To save time and money, computer scientists have aggregated computer power to deliver much higher performance, in a computer system based on parallel programming models.

This thesis focuses on the development and implementation of finite element methods in high performance computing with two specific applications:

- First, we take into account a bio-medical application in the form of computational diffusion magnetic resonance imaging (MRI). Since the transverse magnetization can be modeled by the Bloch-Torrey equation, we can compute numerical approximations of its solution in heterogeneous computational domains from which one derives the MRI signal. The MRI signal can tell us the diffusion characteristics of the water protons in the computational domain. The diffusion characteristics can be used to detect the properties of biological tissue.
• Second, turbulent flow past a vertical axis wind turbine (VAWT) can be modeled by the Navier-Stokes equations with appropriate boundary conditions. We can compute numerical approximations of weak solutions of the equations for rotating VAWTs, from which we can calculate forces acting on the turbines used to determine the efficiency and the load on the turbine.

For computational diffusion MRI, the specific objective is to develop an efficient high performance framework which overcomes the drawbacks of the state of the art methods to become a powerful tool to uncover diffusion in complicated biological tissues. The following steps are taken:

1. Develop an efficient numerical method which adapts well to the HPC framework.
2. Implement the proposed method in the FEniCS-HPC software framework.
3. Verify the accuracy and efficiency of the code.

For simulations of VAWTs, the specific objective is to take advantage of previous developments of simulation methods for turbulent flow to adapt to this specific application. Validations against experimental data will be taken into account. More specifically,

1. Start with a fluid-rigid body interaction model coupled with the arbitrary Lagrangian-Eulerian method in order to compute weak solutions of the Navier-Stokes equations for high Reynolds numbers on moving meshes.
2. Validate computed forces acting on the turbine against experimental data.
3. Develop a full fluid-structure interaction model to model the bending and vibration of the turbine blades.

1.2 Background and state of the art

1.2.1 Computational diffusion MRI

Discovered in 1944 by Isidor Isaac Rabi, diffusion nuclear magnetic resonance (NMR) has rapidly become a powerful tool to probe the diffusion characteristics of molecules in micro-structures. Diffusion is encoded by the use
of external magnetic field gradients. Diffusion MRI is used to measure the average diffusion distance of the molecules in liquids non-invasively.

One of its applications is diffusion magnetic resonance imaging (MRI) which is used to study diffusion characteristics of water molecules inside biological tissues, notably in the brain. There are many medical applications such as to detect stroke in its acute phase, brain tumors, white matter disease, breast cancer and many other clinical applications (see a review in [1]).

Diffusion MRI is based on the fact that when a sample is magnetized by an external gradient field \( \mathbf{B}(t) = (B_x(t), B_y(t), B_z(t)) \), the nuclei (spins) will absorb and re-emit electromagnetic radiation. This response is expressed through the nuclear magnetization which is a vector field

\[
\mathbf{M}(t) = (M_x(t), M_y(t), M_z(t)).
\]

The change in nuclear magnetization over time obeys the Bloch equation [2]

\[
\begin{align*}
\frac{dM_x}{dt} &= \gamma \left( M_y(t) B_z(t) - M_z(t) B_y(t) \right) - \frac{M_x(t)}{T_2} \\
\frac{dM_y}{dt} &= \gamma \left( M_z(t) B_x(t) - M_x(t) B_z(t) \right) - \frac{M_y(t)}{T_2} \\
\frac{dM_z}{dt} &= \gamma \left( M_x(t) B_y(t) - M_y(t) B_x(t) \right) - \frac{M_z(t) - M_0}{T_1}
\end{align*}
\]

where \( \gamma \) is the gyromagnetic ratio with \( \gamma = 2.67513 \times 10^8 \text{ rad s}^{-1} \text{T}^{-1} \) for water protons, \( T_1, T_2 \) are longitudinal and transverse relaxation times respectively, and \( M_0 \) is the steady state nuclear magnetization as \( t \to \infty \). Here the relaxation time is the period of time needed for an excited magnetic state to return its equilibrium state.

The magnetization can be classified into transverse magnetization \( M_{xy} \) and longitudinal magnetization \( M_z \) by introducing complex numbers \( M_{xy} = M_x + i M_y \) and \( B_{xy} = B_x + i B_y \) from which one obtains

\[
\begin{align*}
\frac{dM_{xy}}{dt} &= i \gamma \left( M_{xy}(t) B_z(t) - M_z(t) B_{xy}(t) \right) - \frac{M_{xy}(t)}{T_2} \\
\frac{dM_z}{dt} &= i \gamma \left( M_{xy}(t) B_{xy}(t) - M_{xy}(t) B_{xy}(t) \right) - \frac{M_z(t) - M_0}{T_1}
\end{align*}
\]

where \( \overline{A_{xy}} \) indicates the complex conjugate of \( A_{xy} \), i.e \( \overline{A_{xy}} = A_x - i A_y \) and \( i^2 = -1 \) is the complex unit.
CHAPTER 1. INTRODUCTION

Assume that \( B_x(t) = B_y(t) = 0 \) and \( T_1, T_2 \) are infinitely long, the Bloch equation is simplified to

\[
\frac{dM_{xy}}{dt} = i \gamma M_{xy}(t) B_z(t), \quad \frac{dM_z}{dt} = 0
\]

(1.3)

In order to shorten the notations, we use \( u \) in diffusion MRI from now on to indicate the complex transverse magnetization, i.e \( u = M_{xy} \).

1.2.1.1 The Bloch-Torrey equation

In 1956, H. C. Torrey generalized the Bloch equation by adding a diffusion term that allows for the transfer of magnetization by diffusion. The generalized equation is called the Bloch-Torrey equation [3]. Assume that the gradient field changes linearly in space \( x \) and proportional to the temporal profile \( f(t) \), i.e \( B_z(x, t) = f(t)g \cdot x \). The Bloch-Torrey equation for the transverse magnetization \( u \) is

\[
\frac{\partial u(x, t)}{\partial t} = -i \gamma f(t) g \cdot u(x, t) + \nabla \cdot \left( D(x) \nabla u(x, t) \right)
\]

(1.4)

where \( D(x) \) is the diffusion tensor, and \( g \) is the diffusion gradient including gradient strength and gradient direction. 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 [4]. For this sequence, one can write \( f(t) \) in the following way:

\[
f(t) = \begin{cases} 
1, & 0 \leq t \leq \delta, \\
-1, & \Delta < t \leq \Delta + \delta, \\
0, & \text{otherwise.}
\end{cases}
\]

(1.5)

The quantity \( \delta \) is the duration of the diffusion-encoding gradient pulse and \( \Delta \) is the time delay between the start of the two pulses.

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.

On the interface \( \Gamma = \partial \Omega_0 \cap \partial \Omega_1 \) the magnetization is allowed to be discontinuous via the use of a permeability coefficient \( \kappa \):

\[
D_0(x) \nabla u^0(x, t) \cdot n_0 = -D_1(x) \nabla u^1(x, t) \cdot n_1 = \kappa \left( u^1(x, t) - u^0(x, t) \right)
\]

(1.6)
where \( x \in \Gamma \) and \( n_k \) is a normal vector pointing outward from \( \Omega_k \).

Let \([a] = a_0 - a_1\) and \( \{a\} = \frac{a_0 + a_1}{2} \), the jump conditions on \( \Gamma \) (Eq. 1.6) can be expressed as

\[
\begin{align*}
\{D \nabla u \cdot n_0\} &= 0 \\
\{D \nabla u \cdot n_0\} &= -\kappa \{u\}
\end{align*}
\] (1.7)

There are two commonly used techniques to employ the exterior boundaries \( \partial \Omega \). 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 a low probability of having arrived at \( \partial \Omega \) during the diffusion experiment. This option is straightforward to implement but it requires a lot of computational effort. Another option is to place the spins anywhere desired, but to assume that \( \Omega \) is repeated periodically in all space directions to fill \( \mathbb{R}^d \), for example, \( \Omega = \prod_{k=1}^{d} [a_k, b_k] \).

Under this assumption of the periodic continuation of the geometry, the magnetization satisfies pseudo-periodic boundary conditions on \( \partial \Omega \)

\[
\begin{align*}
u^m &= u^s e^{i \theta_k(t)}, \\
D \nabla u^m \cdot n &= D \nabla u^s \cdot n e^{i \theta_k(t)}
\end{align*}
\] (1.8)

where

\[
\begin{align*}
u^m &= u(x, t) |_{x_k = a_k}, \quad u^s = u(x, t) |_{x_k = b_k} \\
\nabla u^m \cdot n &= \nabla u(x, t) \cdot n |_{x_k = a_k}, \quad \nabla u^s \cdot n &= \nabla u(x, t) \cdot n |_{x_k = b_k}
\end{align*}
\]

and

\[
\theta_k(t) := \gamma g_k (b_k - a_k) F(t), \quad k = 1, \ldots, d, \quad F(t) = \int_0^t f(s) \, ds.
\]

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 standard implementation of the conditions.

From the transverse magnetization, one can calculate the MRI signal which is the total transverse magnetization \( u(x, t) \) over \( \Omega \). The MRI signal is measured at the echo time \( T \)

\[
S = \int_{x \in \Omega} u(x, T) \, dx
\] (1.9)
CHAPTER 1. INTRODUCTION

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

\[
b = \gamma^2 \|g\|^2 \int_0^T \left( \int_0^t f(s) \, ds \right) \, dt. \tag{1.10}
\]

For the PGSE, it is

\[
b = \gamma^2 \|g\|^2 \delta^2 \left( \Delta - \frac{\delta}{3} \right). \tag{1.11}
\]

1.2.1.2 Numerical methods

Solving the complete model, i.e., Eqs. (1.4, 1.7, 1.8), is challenging and many efforts have been made. In \cite{5,6,7} a simplifying assumption called the narrow pulse approximation was used, where the pulse duration \( \delta \) was assumed to be much smaller than the delay between pulses \( \Delta \). 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 \cite{8,9,10,11}. The computational domain is discretized either by a Cartesian grid \cite{8,12,9} or finite elements \cite{5,6,7,10,11}. The unstructured mesh of a finite element discretization appeared to be better than a Cartesian grid in both geometry description and signal approximation \cite{10}. For time discretization, both explicit and implicit methods have been used. In \cite{7} a second order implicit time-stepping method called the generalized \( \alpha \)-method \cite{13} was used to allow for high frequency energy dissipation. An adaptive explicit Runge-Kutta Chebyshev (RKC) method of second order was used in \cite{9,10}. It has been theoretically proven that the RKC allows for a much larger time-step compared to the standard explicit Euler method \cite{14}. There is an example showing that the RKC method is faster than the implicit Euler method in \cite{10}. Recently, the Crank-Nicolson method (CN) was used in \cite{11} to also allow for second-order convergence in time.

The jump conditions (Eq. 1.7) were treated differently in previous works. An average diffusion coefficient was introduced to approximate the permeability condition at the interfaces in \cite{9}. The matrices for jump conditions were explicitly calculated and imposed directly in the stiffness matrix for linear finite elements in \cite{10}. It was generalized to allow higher orders in \cite{11}. Although these techniques are efficient in serial computation, they are less interesting in terms of parallelization due to complicated communications needed.
Local treatments with the jump conditions at the interfaces make it hard to maintain and automate when being implemented.

To mimic the phenomenon where the water molecules can enter and exit the computational domain, the pseudo-periodic boundary conditions were implemented in [8, 9, 10]. The 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 [15, 16, 17] 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 arbitrary meshes. Although these methods are efficient, the finite element matrices need to be constructed in a special way in the implementation. Thus, they are less interesting in terms of parallelization.

1.2.2 Simulations of VAWTs

The interest in harvesting renewable energy, especially wind energy, in offshore as well as in the urban environment has increased significantly in recent years [18, 19, 20, 21]. Vertical axis wind turbines (VAWT) get more and more attention since they are able to capture the wind from any direction, they are easy to install, easy to transport, cheaper to build and maintain, and they are quite safe to humans and birds. They are especially suitable for urban areas or areas with extreme weather. However, they are less efficient than horizontal axis turbines because only one blade is active at a time, and VAWTs also generate a relatively high degree of vibration and noise pollution. These are some of the issues that researchers address to make VAWT more popular, see e.g. [22].

Detailed computational fluid dynamics (CFD) simulations of the flow around VAWTs may give new insights, and provide guidance for design improvements. Fluid dynamics is governed by the Navier-Stokes equations, and the balance of viscous and inertial effects is determined by the Reynolds number (Re),

\[
Re = \frac{\rho U L}{\mu} = \frac{U L}{\nu}, \quad (1.12)
\]
where $\rho$ is density, $\mu$ dynamic viscosity, $U$ a characteristic velocity scale, $L$ a characteristic length scale, and $\nu$ kinematic viscosity $\nu = \mu / \rho$. For high $Re$ the flow is turbulent, which corresponds to chaotic particle trajectories and vortices on a range of scales. The main challenge of CFD is to model turbulent flow, which typically is always present in the flow around a VAWT at operational conditions.

### 1.2.2.1 The Navier-Stokes equations

The Navier-Stokes equations (NSE) become the heart of fluid flow modeling and can be used to model turbulent flows. They are used to describe the motion of fluids with conservation principles of mass, momentum, and energy. For an incompressible flow, it reads

\[
\begin{align*}
\dot{u} + u \cdot \nabla u - \nu \Delta u + \nabla p &= f & \text{in } \Omega \times I \\
\nabla \cdot u &= 0 & \text{in } \Omega \times I \\
u(\cdot, 0) &= u_0 & \text{in } \Omega 
\end{align*}
\] (1.13)

where
- $u$ unknown velocity
- $p$ unknown pressure
- $f$ given body force
- $\Omega \in \mathbb{R}^3$ spatial domain
- $I = [0, T]$ time domain

In a moving domain framework with a mesh velocity $\beta$, the Lagrangian-Eulerian (ALE) method \cite{23} can be used. The convection term is modified and the Navier-Stokes equations become

\[
\begin{align*}
\dot{u} + \left( (u - \beta) \cdot \nabla \right) u - \nu \Delta u + \nabla p &= f & \text{in } \Omega \times I \\
\nabla \cdot u &= 0 & \text{in } \Omega \times I \\
u(\cdot, 0) &= u_0 & \text{in } \Omega 
\end{align*}
\] (1.14)

### 1.2.2.2 Boundary layer model

Since it is not feasible to compute a full resolution of the boundary layer at high Reynolds numbers with current computational resources, appropriate boundary conditions need to be chosen to model turbulence. The general form of the slip with friction and penetration with resistance boundary condition
for a boundary is \[24\]

\[
\begin{align*}
\mathbf{u} \cdot \mathbf{n} + \alpha \mathbf{n}^T \sigma \mathbf{n} &= 0 \\
\mathbf{u} \cdot \mathbf{\tau}_k + \beta^{-1} \mathbf{n}^T \sigma \mathbf{\tau}_k &= 0
\end{align*}
\] (1.15)

where \(\mathbf{n}\) and \(\mathbf{\tau}_k\) are normal and tangential vectors \((k = 1, 2)\) respectively, \(\alpha\) is penetration parameter, \(\beta\) is the skin friction and \(\sigma\) is the stress tensor.

The no-slip conditions \(\mathbf{u} = 0\) corresponds to \((\alpha, \beta) \to (0, \infty)\) and the slip conditions \(\mathbf{u} \cdot \mathbf{n} = 0\) corresponds to \((\alpha, \beta) \to (0, 0)\).

It has been shown in \[25\] that the slip boundary condition can be used to model the turbulent boundary layer.

### 1.2.2.3 Numerical methods for turbulence

Simulation of turbulence is challenging due to the complex mixing of spatial and temporal scales. The industry standard has long been RANS, where a statistical average of the flow is simulated, using turbulence models to model the effect of the fluctuating components of the flow. Large Eddy Simulation (LES) was developed based on the idea of approximating a filtered solution of the Navier-Stokes equations, with the effect of unresolved scales modeled in a subgrid model. LES is very expensive but is still a viable alternative to RANS in some applications. The main problem with both LES and RANS is that the subgrid and turbulence models may be problem-dependent, so that model parameters must be tuned to the particular problem at hand. Direct numerical simulation (DNS) is based on a full resolution of the Navier-Stokes equations, without any turbulence model or subgrid model, but with a computational cost so high that a VAWT simulation would exceed even the most powerful computers of today. We refer to the following VAWT studies for examples of LES \[26, 27, 28\], RANS \[29, 30\], and a coupled LES-RANS approach \[31\].

### 1.2.3 High performance computing

Finding the solution of a PDE in a general computational domain is challenging. One, with a pen, paper, and many assumptions, can only find out some solutions on very basic domains which are too far from the reality. So, one usually has to resort to numerical solutions with the aid of computers.

The discovery of electronic computers opens a new era of solving scientific problems. In the beginning, the computer was designed to work serially. A problem needs to be divided into small tasks and the computer will execute the tasks one by one sequentially.
CHAPTER 1. INTRODUCTION

High performance computing has appeared as one aggregated computing power to deliver much higher performance and the parallelism is the core part of it. "Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously" \cite{32}. In parallel computing, a large problem will be divided into smaller tasks and they will be solved simultaneously. Parallelization is a process to convert a serial code into a parallel code and it can be performed in different forms such as task parallelism and data parallelism. In contrast to data parallelism in which different data components of the same task will be executed at the same time, the task parallelism executes different tasks simultaneously.

Many parallel programming models have been proposed but the most commonly used are the shared memory model and the distributed memory model which can be represented by OpenMP and Message Passing Interface (MPI) respectively. Distributed memory is a memory of a multiprocessor computer system in which each processor has its own private memory. Tasks will be performed independently on processors and communications need to be set up between processors if they share the same data. On the contrary, shared memory can be simultaneously accessed by multiple programs.

One has been interested in high performance computing because of many reasons. The most obvious one is to significantly reduce computational time, and take advantage of the parallel architectures that are standard today. It can also deal with large-scale problems in climate simulations, computational biology, computational fluid dynamics, etc. and contribute to improvements in modern devices used in everyday life. The interest in parallel computing dates back to the late 1950’s when the first commercial machine with floating-point hardware capable of approximately 5 kFLOPS was introduced.

1.3 Thesis Organization

The thesis is organized as follows. We first introduce in Chapter 1 the research objectives which consist of development and implementation of HP-FEM with two concrete scientific applications: diffusion MRI and simulation of turbulent flow past a VAWT. We also recall the background and the state of the art of these two problems. In Chapter 2, the FEniCS-HPC platform, and methods used to address the specific objectives of the thesis outlined in Chapter 1 are introduced. The methods consist of an HP-PUFEM for the interface conditions in diffusion MRI application and the ALE-DFS method for VAWT solution. Some main results are shown in Chapter 3 to emphasize two issues: (1) the difficulties of imposing the discontinuity have been resolved by the use
of PUFEM which appears to be very suitable for automation and parallelization, and (2) the ALE-DFS method works quite well and a good validation has been shown. Chapter 4 consists of discussion and future works. Finally, we summarize some papers included in the thesis in Chapter 5.
Chapter 2

Method

We here describe the methods used in the thesis, to address the specific objectives of the thesis outlined in Section 1.1.

2.1 The FEniCS-HPC framework

FEniCS [33, 34] is a collection of open-source packages to enable automated solution of differential equations. It provides an automated evaluation of variational forms given a high-level description in mathematical notation. It is available in C++ and Python for different operating systems such as Linux, OS X, Unix, and Windows. The core components of FEniCS are:

- UFL allows for defining discrete variational forms and functionals which are close to mathematical notations.
- FIAT is a Python package for automated generation of finite element basis functions.
- FFC is a compiler converting UFL files to UFC files needed for the C++ interface.
- UFC is a C++ interface with low-level functions.
- INSTANT is a Python module for inclining C++ code in python
- DOLFIN manages mesh representation, automated assembly and interfaces with external libraries such as PETSc.
CHAPTER 2. METHOD

FEniCS-HPC\[^{[35, 36, 37]}\] is a C++ branch of FEniCS optimized for massively parallel architectures with the core components DOLFIN-HPC - a Problem Solving Environment (PSE) and Unicorn - an adaptive unified continuum modeling framework\[^{[38, 39]}\]. As in FEniCS, DOLFIN-HPC also handles mesh representation and automated assembly of weak forms but relies on external libraries for solving the linear systems, i.e PETSc. The mesh is fully decomposed by element-based decomposition. Each processing element (PE) executes a local mesh and the matrix assembly is done locally. The overlaps between two PEs are represented by ghosted entities. Therefore, the data dependency is minimized and good parallel scalability has been shown. The Unicorn solver framework was developed based on stabilized FEM, i.e cG(1)cG(1) with duality-based adaptive error control, parameter-free turbulence modeling for flow and fluid-structure interaction in an ALE moving mesh framework. It shows strong linear scaling up to thousands of cores\[^{[40, 41, 42, 43]}\].

2.2 Speedup ratio and parallel efficiency

The speedup ratio and parallel efficiency are used to measure the performance of the parallelization. The speedup ratio is the ratio between timing for serial execution \(T_{\text{serial}}\) and timing for parallel execution \(T_{\text{parallel}}\), i.e

\[
S(p) = \frac{T_{\text{serial}}}{T_{\text{parallel}}(p)}
\]

(2.1)

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

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

The parallel efficiency \(\mathcal{E}\) is computed by

\[
\mathcal{E} = \frac{S}{p}
\]

(2.2)

2.3 An HP-PUFEM for interface conditions

Based on a partition of unity finite element method (PUFEM)\[^{[44, 45]}\], developed for interface problems having interface curves arbitrarily located on a fixed background mesh, we develop a high performance computing framework for large-scale simulations of diffusion to allow for a weak imposition of the jump conditions Eq. (1.7) on confirming meshes.

Let \(Q \subset H^1(\Omega)\), \(Q_0 \subset H^1_0(\Omega)\) and \(V = [Q_0]^2\) denote the standard finite element subspaces of continuous piecewise linear polynomials, the PUFEM
corresponding to the $\theta$–method for Eq. (1.4) with jump conditions Eq. (1.7) is stated as: Find $(U_0^n, U_1^n) \in V$ such that

$$
\left( \frac{U_n^n - U_{n-1}^n}{k_n}, v_h \right) = \left( -i \gamma f(t) g \cdot x U_0^\theta, v_h \right) - \left( D \nabla U_0^\theta, \nabla v_h \right) - \kappa \left\langle [U_0^\theta], [v_h] \right\rangle
$$

(2.3)

for all $(v_0^h, v_1^h) \in V$. Here $k_n$ is the time-step size, $U_0^n = (1 - \theta) U_{n-1}^n + \theta U_n^n$, and

$$
\left( U_h, v_h \right) = \int_\Omega (1 - \Phi) U_0 v_0 \, dx + \int_\Omega \Phi U_1 v_1 \, dx
$$

$$
[U_h] = U_0 - U_1, \quad \{U_h\} = \frac{U_0 + U_1}{2}, \quad \left\langle U_h, v_h \right\rangle = \int_\Gamma U_h v_h \, dS,
$$

and the phase function $\Phi$ is defined as

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

The approximation we need to search for is $\bar{U}_h = (1 - \Phi) U_0 + \Phi U_1$. We note that the weak formulation Eq. (2.3) allows for imposing the interface conditions weakly. Thus, no explicit parallel implementation is needed to impose the permeability condition. It also allows for an easy generalization to arbitrary order finite elements and facilitates very well for automation and parallelization.

## 2.4 The ALE-DFS method for VAWT solution

In this thesis, we simulate turbulent flow around VAWTs with a cG(1)cG(1) method in a framework of high performance adaptive finite element methods.

### 2.4.1 cG(1)cG(1) method

The cG(1)cG(1) method is based on the General Galerkin method [24] with piecewise linear basis functions in both time and space. Let $0 = t_0 < t_1 < \cdots < t_N = T$ be a time partition associated with the time intervals $I_n = [t_n-1, t_n]$ of length $k_n = t_n - t_{n-1}$. Let $Q \subset H^1(\Omega)$, $Q_0 \subset H_0^1(\Omega)$ and $V = [Q_0]^3$ denote the standard finite element subspaces of continuous piecewise
CHAPTER 2. METHOD

linear polynomials. Then the cG(1)cG(1) method in the ALE framework with least-squares stabilization is stated as: For all time intervals \( I_n \), find \( (U^n_h, P^n_h) \in V \times Q \) such that

\[
\begin{align*}
\left( \frac{U^n_h - U^{n-1}_h}{k_n} \right) + ((\tilde{U}^n_h - \beta_h) \cdot \nabla) U^n_h, v_h \right) \
+ \left( \nu \nabla \tilde{U}^n_h, \nabla v_h \right) - \left( P^n_h, \nabla \cdot v_h \right) + \left( \nabla \cdot \tilde{U}^n_h, q_h \right) \
+ SD_\delta \left( U^n_h, P^n_h, v_h, q_h \right) = (f, v_h)
\end{align*}
\]

for all \((v_h, q_h) \in V \times Q\), where \( \tilde{U}^n_h = \frac{U^n_h + U^{n-1}_h}{2} \) and with stabilization term

\[
SD_\delta(\tilde{U}^n_h, P^n_h; v, q) := \left( \delta_1 \left( (\tilde{U}^n_h - \beta_h) \cdot \nabla \tilde{U}^n_h + \nabla P^n_h - f_n \right), \left( \tilde{U}^n_h - \beta_h \right) \cdot \nabla v + \nabla q \right) \
+ \left( \delta_2 \nabla \cdot \tilde{U}^n_h, \nabla \cdot v_h \right)
\]

Here \( \delta_1 \) and \( \delta_2 \) are given stabilization parameters:

- \( \delta_1 = C_1 \left( k_n^{-2} + |U^{n-1}_h - \beta_h|^2 h_n^{-2} \right)^{-1/2} \)
- \( \delta_2 = C_2 |U^{n-1}_h| h_n \)

We note that under CFL condition, i.e. \( k_n \approx \frac{h_n}{|U^{n-1}_h - \beta_h|} \), \( \delta_1 \) is simplified to \( \delta_1 = C_1 \frac{h_n}{|U^{n-1}_h - \beta_h|} \) and \( \beta_h \) is a discrete approximation of the mesh velocity \( \beta \).

2.4.2 Automated adaptivity with do-nothing error control

The adjoint problem of the Navier-Stokes equations is a linear convection-diffusion-reaction problem where the convection acts backward in time and in the opposite direction of the exact flow velocity \( u \) \[46, 47, 48\].

\[
\begin{align*}
- \varphi + ((u - \beta) \cdot \nabla) \varphi + \nabla U_T \cdot \varphi + \nabla \theta = \Psi & \quad \text{in } \Omega \times I \\
\nabla \cdot \varphi = 0 & \quad \text{in } \Omega \times I \\
\varphi(\cdot, T) = 0 & \quad \text{in } \Omega
\end{align*}
\]

(2.4)

where \( \nabla U_T \cdot \varphi \) is the dual solution corresponding to the primal solution \( \hat{u} = (u, p) \), and \( \Psi \in L_2(\Omega) \) is a given weight function.
used to define the quantity of interest of $\hat{u}$ for a choice of suitable boundary conditions

$$M(\hat{u}) = (\hat{u}, \Psi)$$

It follows that

$$|M(\hat{u}) - M(\hat{U})| = |(R(\hat{U}), \hat{\phi})| = \left| \sum_{K \in T_n} (R(\hat{U}), \hat{\phi})_K \right|$$  \hspace{1cm} (2.5)$$

Here $R(\hat{u})$ is the residual of the Navier-Stokes equations (Eq. 1.13):

$$R(\hat{u}) = \left( R_1(\hat{u}), R_2(\hat{u}) \right)$$

$$R_1(\hat{u}) = \dot{u} + \left( (u - \beta) \cdot \nabla \right) u - \nu \Delta u + \nabla p - f$$ \hspace{1cm} (2.6)$$

$$R_2(\hat{u}) = \nabla \cdot u$$

The do-nothing error indicator is defined element-wise as

$$e^K_N = (R(\hat{U}), \hat{\phi}_h)_K$$ \hspace{1cm} (2.7)$$

where $\hat{\phi}_h$ is a numerical approximation of the adjoint problem.
Chapter 3

Results

Through the thesis, we have developed two HP-FEM methods: (1) An HP-PUFEM for the interface problem in diffusion MRI, and (2) an ALE-DFS method for simulation of turbulent flow past a VAWT. Two corresponding software packages have been released with numerous validations. A good parallel scalability has been shown.

3.1 Simulation of diffusion MRI

An HP-PUFEM has been proposed to address the interface conditions with some interesting features. First, the interface conditions are imposed weakly. It does not only significantly simplify the implementation but also minimizes the communications in parallelization. Second, the method was straightforward to implement and maintain in the FEniCS/FEniCS-HPC platform with a few command lines and it can be straightforward for other FEM software packages. The method seems to be easy to extend for different interface problems beyond this application.

Fig. 3.1 shows timing, speedup ratio and parallel efficiency on the Beskow supercomputer for an MRI application implemented in FEniCS-HPC. The serial computation costs about 40 hours and the parallel computation with 256 MPI processes only costs 17 minutes (Fig. 3.1a). The speedup ratio is nearly optimal (Fig. 3.1b) and the efficiency is more than 40% (Fig. 3.1c).

Fig. 3.2b shows diffusion in a pyramidal neuron embedded in a large box which was meshed with 1.5M vertices and 8.5M tetrahedrons. The simulated results reflect correctly the physics for different permeabilities between the
neuron and the extra-cellular space. More details are presented in Paper 5.2 [49].

![Figure 3.1](image1.png)

**Figure 3.1:** Timing, speedup ratio and parallel efficiency on the Beskow for an MRI application implemented on FEniCS-HPC.

![Figure 3.2](image2.png)

**Figure 3.2:** A pyramidal neuron of an adult female mouse downloaded from [http://neuromorpho.org](http://neuromorpho.org) and embedded in the center of a computational box (a) which was meshed with 1.5M vertices and 8.5M tetrahedrons. The neuron itself needs small elements to describe accurately small dendrites 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 (b). The vertical segment at each marker indicates the signal variation. The signals decay faster for higher membrane permeabilities.
3.2 Validation of parked VAWT

The numerical method has been implemented in the Unicorn solver \cite{38, 50} in the FEniCS-HPC platform \cite{35, 36} which is a high performance computing branch of FEniCS \cite{33, 34}.

Fig. 3.3 shows a comparison of drag coefficients between simulation and experiment. The method works well and we capture well the shape of the force coefficient curve. More details are presented in Paper 5.1 and Paper 5.3.

![Figure 3.3: A comparison of drag coefficients between simulation and experiment versus wind directions for constant wind profile (a) and logarithmic wind profile (b).](image)

\cite{5.3}
Chapter 4

Discussion and Future Works

4.1 Computational diffusion MRI

We proposed an efficient HP-PUFEM framework to solve the Bloch-Torrey equation in heterogeneous domains. The method would be straightforward to implement in different FEM software packages and it has been implemented in FEniCS for moderate-scale problems and FEniCS-HPC for large-scale problems. The MPI communication is minimized since the boundary conditions are weakly imposed in the variational forms. The accuracy of the method has been verified by numerous examples and the implementation has been shown to have a good parallel scalability.

Simulations on a realistic neuron were performed to show that the framework can be used to uncover diffusion characteristics in complicated geometries. The framework can become a powerful tool in computational diffusion MRI and realistic applications are under consideration.

4.2 Simulations of VAWT

We proposed an efficient DFS-ALE method to solve the Navier-Stokes equation for VAWTs. The method combines powerful techniques to simulate turbulent flow such as the least-squares stabilized finite element method, the ALE method for a rotating frame of reference and the dual-based error control for automated mesh adaptivity. The method was validated against experimental data in parked conditions with a good agreement in drag coefficients.

The next step is to validate the method in rotating conditions. A comparison with actuator line methods is under consideration. More work needs to
be done with automated error control on a rotating frame of reference. The method can be used to study the flow around rotating objects in general.

Another goal is to implement the slip velocity boundary conditions to the unified continuum fluid-structure interaction model proposed in [51, 41] to model the bending and vibration of the turbine blades. These boundary conditions have been used to simulate the effect of turbulent boundary layers in fluid-rigid body interaction models [25] and have been applied successfully in many applications in aerodynamics. The implementation can be done by using the PUFEM approach. Fig. 4.1 shows the velocity approximation of a 2D Euler flow in a fluid-structure interaction model by a stabilized finite element method with slip velocity boundary conditions imposed at the internal interface. The result gives a qualitative indication that the PUFEM correctly implements the slip boundary conditions for the Euler flow.

Figure 4.1: The velocity approximation of a 2D Euler flow in a fluid-structure interaction model by a stabilized finite element method with slip velocity boundary conditions imposed at the internal interface. It gives a qualitative indication that the PUFEM correctly implements the slip boundary conditions for the Euler flow.
Chapter 5

Abstracts of included papers

5.1 Paper I


- The author of this thesis implemented the method, performed simulations and prepared the manuscript.

- Abstract: We combine the unified continuum fluid-structure interaction method with a multiphase flow model to simulate turbulent flow and fluid-structure interaction of rotating vertical axis turbines in offshore environments. This work is part of a project funded by the Swedish Energy Agency, which focuses on energy systems combining ecological sustainability, competitiveness, and reliability of supply. The numerical methods used comprise the Galerkin least-squares finite element method, coupled with the arbitrary Lagrangian-Eulerian method, in order to compute weak solutions of the Navier-Stokes equations for high Reynolds numbers on moving meshes. Mesh smoothing methods help to improve the mesh quality when the mesh undergoes large deformations. The simulations have been performed using the Unicorn solver in the FEniCS-HPC framework, which runs on supercomputers with near-optimal weak and strong scaling up to thousands of cores.
5.2 Paper II


- The author of this thesis developed ideas, implemented the proposed method, performed simulations and prepared the manuscript.

- 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 base 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.

Highlights:

– The discontinuity in the magnetization across the interior interfaces of the medium is weakly imposed, allowing generalization to arbitrary order finite elements.
– Spin exchange across the external boundaries is implemented by weakly imposing an artificial, high permeability, condition, allowing generalization to non-matching meshes.
– Optimal convergence with respect to the space discretization is achieved.
– The second-order Crank-Nicolson method is chosen for time discretization to reduce oscillations at high gradient strengths and allows for larger time-step sizes.
– The method is of a high level of simplicity and suitable for parallelization.
– An efficient open-source software framework is implemented in the FEniCS and FEniCS-HPC platforms.

5.3 Paper III


• The author of this thesis implemented the methods, performed simulations and prepared the manuscript.

• Abstract: There is today a significant interest in harvesting renewable energy, specifically wind energy, in offshore and urban environments. Vertical axis wind turbines get increasing attention since they are able to capture the wind from any direction, they are relatively easy to install and to transport, cheaper to build and maintain, and they are quite safe for humans and birds. Detailed computer simulations of the fluid dynamics of wind turbines provide an enhanced understanding of the technology and may guide design improvements. In this paper, we simulate the turbulent flow past a parked vertical axis wind turbine for a
range of rotation angles. The force on a parked turbine is one of the most important design cases for the survival of the turbine. We use the method of Direct Finite Element Simulation, and the simulation results are validated against experimental data in the form of force measurements. First, simulations are performed for a set of sampled rotation angles, and then a simulation is performed where the turbine is slowly rotated to cover all the rotation angles continuously. We find that the simulation results are stable with respect to mesh refinement and that we capture the general shape of the variation of force measurements over the rotation angles.
Bibliography


MODELLING OF ROTATING VERTICAL AXIS TURBINES
USING A MULTIPHASE FINITE ELEMENT METHOD

Van Dang Nguyen*, Johan Jansson†‡, Massimiliano Leoni†, Bärbel Janssen*,
Anders Goude‡, and Johan Hoffman*

*Department of Computational Science and Technology, School of Computer Science and
Communication, KTH Royal Institute of Technology, Stockholm, Sweden.
e-mail: {vdnguyen, jjan, barbel, jhoffman}@kth.se

† Basque Center for Applied Mathematics, Bilbao, Spain.
e-mail: mleoni@bcamath.org.

‡ Uppsala University, Uppsala, Sweden.
e-mail: anders.goude@angstrom.uu.se

Key words: Vertical axis turbines, fluid-structure interaction, fluid-rigid body interaction,
Unicorn solver, FEniCS-HPC, Navier-Stokes equations, multiphase finite element method.

Abstract. We combine the unified continuum fluid-structure interaction method with a mul-
tiphase flow model to simulate turbulent flow and fluid-structure interaction of rotating ver-
tical axis turbines in offshore environments. This work is part of a project funded by the
Swedish Energy Agency, which focuses on energy systems combining ecological sustainability,
competitiveness and reliability of supply. The numerical methods used comprise the Galerkin
least-squares finite element method, coupled with the arbitrary Lagrangian-Eulerian method,
in order to compute weak solutions of the Navier-Stokes equations for high Reynolds numbers
on moving meshes. Mesh smoothing methods help to improve the mesh quality when the mesh
undergoes large deformations. The simulations have been performed using the Unicorn solver in
the FEniCS-HPC framework, which runs on supercomputers with near optimal weak and strong
scaling up to thousands of cores.

1 INTRODUCTION

Wind and water have been sources of energy for thousands of years, and one of the most
utilized during the 17-th and 18-th centuries [10]. Nowadays, the increased interest in renewable
energy resources has caused an increase in wind and water power utilization, and improvement of
wind and hydraulic turbine technology. The function of a wind turbine is the result of a complex
interaction of its components and subsystems such as the blades, rotor, tower, foundation, power
train, control system, etc. [1]. Vertical axis turbines have long been used to convert kinetic
energy from wind/water into electrical power since they are easy to install with low cost and
Modelling of rotating vertical axis turbines

high safety. Simulated results of wind turbines may explain some physical phenomena and give some guidance for improvement. However, simulation of rotating turbines is a challenging problem, considered an open problem in all its complexity, even though some impressive results have been achieved using stabilized finite element methods [2, 3].

In this paper, we combine the unified continuum fluid-structure interaction method [7] with a multiphase flow model [9] to simulate turbulent flow and fluid-structure interaction of rotating turbines in offshore environments. This work is part of a project funded by the Swedish Energy Agency, which focuses on energy systems combining ecological sustainability, competitiveness and reliability of supply. The numerical method used comprises the Galerkin least-squares finite element method, coupled with the arbitrary Lagrangian-Eulerian method in order to allow us to compute weak solutions of the Navier-Stokes equations for high Reynolds numbers on moving meshes. Mesh smoothing methods help to improve the mesh quality when the mesh undergoes large deformations. The simulations have been performed using the Unicorn solver in the FEniCS-HPC framework, which runs on supercomputers with near optimal weak and strong scaling up to thousands of cores.

The paper is organized as follows. We first introduce in Section 2 the vertical axis turbine used for all simulations in the paper. In Section 3 we consider a simplified model based on fluid-rigid body interaction in which the turbine rotates with a given rotational speed applied to the computation mesh. In Section 4 we study a full fluid-structure interaction model. The computer implementation is discussed in Section 5. In Section 6 we present some preliminary results and suggest future directions.

2 A VERTICAL AXIS TURBINE

The turbine under consideration is a single 3-bladed H-rotor turbine, with a radius of 3.24 m and a blade length of 5 m (Fig. 1a is reproduced from [5]). The blades are pitched outwards with a chord length of 0.25 m at the middle of the blade. For simplification, we assume that the turbine axis is coincident with the $z$-axis and that the turbine $Ω^T$ is placed in a cylinder $Ω^C$ (Fig. 1b)

$$Ω^C = \{(x, y, z) \in \mathbb{R}^3 | x^2 + y^2 \leq R^2, z \in [-L/2; L/2]\}. \quad (1)$$

Here we choose $R = 30$ m and $L = 20$ m, with the turbine axis placed in the center-line of the cylinder domain. Let $f$ and $s$ indicate fluid and structure parts respectively and for fluid-structure interaction we denote $Ω^f = Ω^C \setminus Ω^T$ and $Ω^s = Ω^T$. 

2
3 FLUID-RIGID BODY INTERACTION MODEL

We start with a simplified model where the turbine is assumed to be a rigid body, so that we remove $\Omega^s$ from the domain and solve only for the fluid domain $\Omega = \Omega^f$ (Fig. 2) where the turbine is forced to rotate with a given rotational velocity.
Modelling of rotating vertical axis turbines

The incompressible Navier-Stokes equations described in the following need to be solved in the fluid part, $\Omega = \Omega_f$:

\[
\begin{align*}
\dot{u} + (u \cdot \nabla)u - \nu \Delta u + \nabla p &= f \quad \text{in } \Omega \times I \\
\nabla \cdot u &= 0 \quad \text{in } \Omega \times I \\
u(u(\cdot, 0) - u_0) &= 0 \quad \text{in } \Omega
\end{align*}
\]  

(2)

Here, $u$ is the unknown velocity, $p$ is the unknown pressure, $\nu$ is the kinematic viscosity computed as the ratio between the dynamic viscosity $\mu$ and the density $\rho$, i.e. $\nu = \mu/\rho$, and $f$ is a given body force. Eq. (2) is subjected to the slip velocity boundary condition

\[
\mathbf{u} \cdot \mathbf{n} = 0
\]  

(3)

or the no-slip velocity boundary condition

\[
\mathbf{u} = 0
\]  

(4)

on the boundaries. Here $\mathbf{n}$ is the outward unit normal of the fluid boundary and $\sigma$ denotes the stress tensor. The inflow velocity boundary condition $\mathbf{u} = (1, 0, 0)$ and the outflow condition $p = 0$ are imposed on $\Gamma^i$ and $\Gamma^o$ respectively. These boundaries are defined as follows

\[
\begin{align*}
\Gamma^T &= \partial \Omega \cap \Omega^T \\
\Gamma^i &= \{ x \in \partial \Omega \mid x \leq 0 \} \setminus \Gamma^T \\
\Gamma^o &= \{ x \in \partial \Omega \mid x > 0 \} \setminus \Gamma^T
\end{align*}
\]  

(5) (6) (7)

The computational domain rotates under the operational mesh velocity $\mathbf{\beta} = (\dot{x}, \dot{y}, \dot{z})$ which can be explicitly computed for a given angular velocity $\omega$ and time-step size $k$:

\[
\begin{align*}
\dot{x} &= \frac{1}{k} \left( x (\cos(\omega) - 1) - y \sin(\omega) \right) \\
\dot{y} &= \frac{1}{k} \left( y (\cos(\omega) - 1) + x \sin(\omega) \right) \\
\dot{z} &= 0
\end{align*}
\]  

(8)

The mesh and its corresponding finite element function spaces need to be updated at each time step. However, the solution from the previous time step needs to be interpolated to the new function spaces. This process is expensive and accumulates interpolation errors. To overcome the problem, the arbitrary Lagrangian-Eulerian (ALE) method \cite{4} is applied. This method automatically projects the solution from the previous time step to the current function space without using interpolation. The Navier-Stokes’ momentum Eq. (2) formulated in ALE coordinates with a mesh velocity $\mathbf{\beta}$ reads

\[
\dot{u} + \left( (u - \mathbf{\beta}) \cdot \nabla \right) u - \nu \Delta u + \nabla p = f
\]  

(9)
The Galerkin least-squares space-time FEM [6] is used to obtain an efficient method for the discretization of a rotating turbine problems. Let 0 = t_0 < t_1 < \cdots < t_N = T be a time partition associated with the time intervals I_n = (t^{n-1}, t^n] of length k_n = t^n - t^{n-1}. Let \mathbf{Q} \subset H^1(\Omega), \mathbf{Q}_0 \subset H^1_0(\Omega) and V = [\mathbf{Q}_0]^3$. The ALE finite element method with least-squares stabilization is stated as the following. For all time intervals I_n, find \((U^n, P^n) ∈ V_h × Q_h\) such that

\[
\begin{aligned}
&\left((U^n - U^{n-1})k_n^{-1} + (\bar{U}_h^n - \beta_h) \cdot \nabla\right)\bar{U}_h^n, v_h + \left(\nu \nabla \bar{U}_h^n, \nabla v_h\right) - \left(P_h^n, \nabla \cdot v\right) \\
&\quad + \left(\nabla \cdot \bar{U}_h^n, q_h\right) + SD_1 \left((\bar{U}_h^n - \beta_h) \cdot \nabla \bar{U}_h^n + \nabla P_h^n - f_h\right) + \left((\bar{U}_h^n - \beta_h) \cdot \nabla v_h + \nabla q_h\right) \\
&\quad + \left(\delta_2 \nabla \cdot \bar{U}_h^n, \nabla \cdot v_h\right) = \left(f_h, v_h\right)
\end{aligned}
\] (10)

for all \((v_h, q_h) ∈ V_h × Q_h\), where

\[
SD_1^n (\bar{U}_h^n, P_h^n; v_h, q_h) := \left(\delta_1 \left((\bar{U}_h^n - \beta_h) \cdot \nabla \bar{U}_h^n + \nabla P_h^n - f_h\right) + (\bar{U}_h^n - \beta_h) \cdot \nabla v_h + \nabla q_h\right)
\]

Here the midpoint method is used for the time stepping, i.e.

\[
\bar{U}_h^n = \frac{U_h^n + U_h^{n-1}}{2},
\]

\(\delta_1\) and \(\delta_2\) are stabilization parameters given by positive constants \(C_1, C_2\), the time step size \(k_n\)

\[
k_n \sim \min\left(\frac{h_n}{U_h^{n-1}}\right)
\]

and the mesh size \(h_n\)

\[
\delta_1 = C_1 \left(1 + \frac{U_h^{n-1} - \beta_h}{k_n^2} \right)^{-1/2}
\]

\[
\delta_2 = C_2 \frac{|U_h^{n-1}|}{h_n}
\] (12)

Since the inflow and outflow boundary conditions are not moving with the mesh, they need to be updated for each new time step.

### 4 FLUID-STRUCTURE INTERACTION MODEL

Including structure deformation, the rotating turbine can be described using a unified continuum fluid-structure interaction model [7]. In ALE coordinates, it reads:

\[
\begin{aligned}
\rho \left(\dot{\mathbf{u}} + \left((\mathbf{u} - \mathbf{\beta}) \cdot \nabla\right) \mathbf{u}\right) + \nabla \cdot \mathbf{\sigma} = \mathbf{f} \quad \text{in } \Omega × I \\
\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega × I \\
\dot{\Theta} + \left((\mathbf{u} - \mathbf{\beta}) \cdot \nabla\right) \Theta = 0 \quad \text{in } \Omega × I \\
\mathbf{u}(\cdot, 0) = \mathbf{u}_0 \quad \text{in } \Omega
\end{aligned}
\] (13)
The no-slip velocity boundary condition in this setting means that \( \sigma \) on \( \Gamma \) and no-slip velocity Eq. (16) are implicitly imposed. Emphasizing that in the unified continuum fluid-structure interaction model [7], this condition \( \sigma \) also needs to assure the force balance condition at the interface, i.e.

\[
\sigma = -\sigma_D + \rho \mathbf{T}
\]

For a Newtonian fluid and an incompressible Neo-Hookean solid, the stress \( \sigma \) is computed as the following:

\[
\begin{align*}
\sigma &= -\sigma_D + \rho \mathbf{T} \\
\sigma_D &= \theta \sigma_f + (1 - \theta) \sigma_s \\
\sigma_f &= 2\mu_f \varepsilon(u) \\
\sigma_s &= 2\mu_s \varepsilon(u) + \nabla u \sigma_s + \sigma_s \nabla u^T
\end{align*}
\]

The no-slip velocity boundary condition in this setting means that

\[
u_f = u_s
\]

on \( \Gamma^{fs} \) and the slip condition is

\[
u_f \cdot n = u_s \cdot n
\]

We also need to assure the force balance condition at the interface, i.e. \( \sigma_f \cdot n = \sigma_s \cdot n \). It is worth emphasizing that in the unified continuum fluid-structure interaction model [7], this condition and no-slip velocity Eq. (16) are implicitly imposed.

The turbine is pointwise fixed along the turbine axis by an additional condition, i.e. \( u = 0 \). This condition allows the turbine rotating along its axis and will not be blown away.

The least-square finite element method for the FSI model is stated as the following. Find \((U^n_{h}, P^n_{h})\) with \( U^n \in V_h \equiv [Q_h]_3^n \) and \( P^n_{h} \in Q_h \) such that

\[
\left( \rho \left( (U^n_{h} - U^{n-1}_{h}) \kappa_n^{-1} + \left( \bar{U}_{h} - \beta_h \right)^n \cdot \nabla \bar{U}^n_{h} \right), v_h \right) + \theta \left( 2\mu_f \varepsilon(\bar{U}^n_{h}), \varepsilon(v_h) \right) + \left( 1 - \theta \right) \left( S_s, \nabla v_h \right)
\]

\[
- (P^n, \nabla \cdot v) + (\nabla \cdot \bar{U}^n, q) + SD^n_{\delta}(\bar{U}^n_{h}, P^n_{h}; v_h, q_h) = (f_h, v_h)
\]

\[\forall (v_h, q_h) \in V_h \times Q_h\]

with stabilisation term

\[
SD^n_{\delta}(\bar{U}^n_{h}, P^n_{h}; v_h, q_h) := \left( \delta_1 \rho \left( (\bar{U}^n_{h} - \beta_h) \cdot \nabla \bar{U}^n_{h} + \nabla P^n_{h} - f_h \right), \rho (\bar{U}^n_{h} - \beta_h) \cdot \nabla v_h + \nabla q_h \right)
\]

\[
+ \left( \delta_2 \nabla \cdot \bar{U}^n_{h}, \nabla \cdot v_h \right)
\]

where \( \delta_1 = C_1 \rho^{-1} \left( \kappa_n^{-2} + |U_h^{n-1} - \beta_h|^2 h^{-2}_n \right)^{-1/2}, \delta_2 = C_2 \rho |U_h^{n-1}| h_n \) and \( S_s \) is the numerical solution of Neo-Hookean solid equation Eq. [15].

Since the solid part of the mesh moves, the mesh at the interface between the solid and the fluid becomes ill conditioned. A Laplacian mesh smoothing technique has been used to redistribute the vertices of the mesh in the fluid domain. The technique works by moving all the internal vertices to the center of all its neighbors. The mesh motion \( \beta \) is found by solving the following Laplace equation

\[
\begin{align*}
- \Delta \beta &= 0 \text{ in } \Omega_f \\
\beta &= u \text{ in } \Omega_s \\
\beta \cdot n &= 0 \text{ on } \partial \Omega
\end{align*}
\]

The computational domain is \( \Omega = \Omega^f \cup \Omega^s , \Gamma^{fs} = \overline{\Omega^f} \cap \overline{\Omega^s} \), the phase function \( \theta \) defines the structure and fluid domains

\[
\theta(x, t) = \begin{cases} 
1 & \text{if } x \in \Omega^f \\
0 & \text{if } x \in \Omega^s 
\end{cases}
\]

(14)
5 IMPLEMENTATIONS

The two numerical models described above have been implemented in Unicorn and the high performance branch of the finite element problem solving environment DOLFIN-HPC which is optimized for distributed memory architectures using a hybrid MPI+OpenMP approach with efficient parallel I/O. Both Unicorn and DOLFIN-HPC are written in C++ and the implementation has proven to be portable across several different architectures, such as ordinary Unix/Linux workstations. The first model has been implemented with both the slip condition (Eq. 3) and the no-slip condition (Eq. 4), whereas the second model has only been implemented with the no-slip condition in Eq. (17).

6 PRELIMINARY RESULTS AND DISCUSSION

The proposed methods were implemented in the Unicorn solver for parallel adaptive finite element simulation of turbulent flow and fluid-structure interaction. The solver showed near optimal weak and strong scaling results when simulating turbulent flow on massively parallel machines [8]. The adaptive mesh refinement in the solver, however, is disabled for this case since without coarsening the $h$–adaptivity is not suitable for rotating computational domains. The use of $r$–adaptivity may be a better choice for the ALE case of rotating domain and we leave it for future works. The simulations were carried out on the Beskow supercomputer at the KTH Royal Institute of Technology, that is a Cray XC40 system, based on Intel Haswell processors and Cray Aries interconnect network. Each Beskow node has 32 cores divided into two sockets, with 16 cores each, and thus the total number of cores is 53632. The available RAM for each node is 64 GB DDR3 SDRAM.

The volume meshes were generated with ANSA from Beta-CAE Systems with 556789 vertices for the fluid part and 116622 vertices for the structure part. We note that in the fluid-rigid body interaction model, only the fluid part of the mesh is used whereas in the fluid-structure interaction model, the computational mesh unifies the fluid mesh and the structure mesh.

Numerical simulations show that the fluid-rigid body interaction model works efficiently. We show in Fig. 3 drag forces for three rotational velocities: fixed turbine, $\pi/200$ rad/s and $\pi/20$ rad/s. A larger rotational velocity gives larger oscillations. Some snapshots in time for the two rotational velocities are visualized in Fig. 4 and Fig. 5 respectively.

Fig. 6a shows a vertical cut of the computed velocity for the fluid-structure interaction model where the no-slip velocity Eq. (16) is applied. Under the effect of the inflow velocity, the turbine rotates for a while (Fig. 6b) before stopping at the position shown in Fig. 6c. One of the reasons may be because the no-slip boundary condition (16) is not suitable. A Nitsche method for the slip condition [17] is under consideration and it is expected to solve the problem.

ACKNOWLEDGEMENTS

This research has been supported by the European Research Council, the Swedish Energy Agency, 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, 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. We acknowledge the Swedish National Infrastruc-
Figure 3: Drag forces for three rotational velocities: fixed turbine, $\pi/200$ rad/s and $\pi/20$ rad/s.

The initial volume mesh was generated with ANSA from Beta-CAE Systems S. A., who generously provided an academic license for this project.

REFERENCES


Modelling of rotating vertical axis turbines

Figure 4: Rotating turbine at rotational speed $\pi/200$ rad/s, for 4 snapshots in time: horizontal cuts (left) and vertical cuts (right).


[7] Johan Hoffman, Johan Jansson, Rodrigo V. de Abreu, Cem Degirmenci, Niclas Jansson,
Figure 5: Rotating turbine at rotational speed $\pi/20$ rad/s, for 4 snapshots in time: horizontal cuts (left) and vertical cuts (right).


Figure 6: A vertical cut of the computed velocity (a). Under the effect of the inflow velocity, the turbine only rotates for a while (b) before stopping in the position shown in (c).


A partition of unity finite element method for computational diffusion MRI

Van-Dang Nguyen\textsuperscript{a,*}, Johan Jansson\textsuperscript{a}, Johan Hoffman\textsuperscript{a}, Jing-Rebecca Li\textsuperscript{b}

\textsuperscript{a}Department of Computer Science, KTH Royal Institute of Technology, Sweden
\textsuperscript{b}INRIA Saclay-Equipe DEFI, CMAP, Ecole Polytechnique Route de Saclay, 91128, Palaiseau Cedex, France

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

\textsuperscript{*}I am corresponding author

Email addresses: vdnguyen@kth.se (Van-Dang Nguyen), jjjan@kth.se (Johan Jansson), jhoffman@kth.se (Johan Hoffman), jingrebecca.li@inria.fr (Jing-Rebecca Li)
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.

Highlights:

- The discontinuity in the magnetization across the interior interfaces of the medium is weakly imposed, allowing generalization to arbitrary order finite elements.
- Spin exchange across the external boundaries is implemented by weakly imposing an artificial, high permeability condition allowing generalization to non-matching meshes.
- Optimal convergence with respect to the space discretization is achieved.
- The second-order Crank-Nicolson method is chosen for time discretization to reduce oscillations at high gradient strengths and allows for larger time-step sizes.
- The method is of a high level of simplicity and suitable for parallelization.
- An efficient open-source software framework is implemented in the FEniCS and FEniCS-HPC platforms.

Keywords: computational diffusion MRI, Bloch-Torrey equation, partition of unity finite element method, interface conditions, weak pseudo-periodic
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 \( u(x,t) \) is

\[
\frac{\partial u(x,t)}{\partial t} = -I \gamma f(t) g \cdot x u(x,t) + \nabla \cdot \left( D(x) \nabla u(x,t) \right)
\]

where \( I \) is the complex unit (\( I^2 = -1 \)), \( \gamma = 2.67513 \times 10^8 \text{ rad s}^{-1} \text{T}^{-1} \) is the gyromagnetic ratio of the water proton, \( D(x) \) is the diffusion tensor, and \( g = (g_1, \cdots, g_d) \) is the diffusion gradient including gradient strength and gradient direction, \( d \) is the space dimension. 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 [2]. For this sequence,
one can write $f(t)$ in the following way (see also Fig. 1b):

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

The quantity $\delta$ is the duration of the diffusion-encoding gradient pulse and $\Delta$ 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 $\kappa$:

$$D_0(x)\nabla u_0(x, t) \cdot n_0 = -D_1(x)\nabla u_1(x, t) \cdot n_1 = \kappa(u_1(x, t) - u_0(x, t)), \quad (3)$$

where $x \in \Gamma = \partial \Omega_0 \cap \partial \Omega_1$ and $n^k$ is a normal vector pointing outward $\Omega_k$.

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 peri-
odic continuation of the geometry, the magnetization satisfies pseudo-periodic
boundary conditions on \( \partial \Omega \):

\[
\mathbf{u}_m = \mathbf{u}_s e^{i \theta_k(t)},
\]

\[
D \nabla \mathbf{u}_m \cdot \mathbf{n} = D \nabla \mathbf{u}_s \cdot \mathbf{n} e^{i \theta_k(t)},
\]

(4)

where

\[
\mathbf{u}_m = \mathbf{u}(\mathbf{x}, t)|_{x_k = a_k}, \quad \mathbf{u}_s = \mathbf{u}(\mathbf{x}, t)|_{x_k = b_k},
\]

\[
\nabla \mathbf{u}_m \cdot \mathbf{n} = \nabla \mathbf{u}(\mathbf{x}, t) \cdot \mathbf{n} \bigg|_{x_k = a_k}, \quad \nabla \mathbf{u}_s \cdot \mathbf{n} = \nabla \mathbf{u}(\mathbf{x}, t) \cdot \mathbf{n} \bigg|_{x_k = b_k}
\]

and

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

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 [5].

The MRI signal \( S \) is the total transverse magnetization \( u(\mathbf{x}, t) \) over \( \Omega \) mea-
sured at the echo time \( T \)

\[
S = \int_{\mathbf{x} \in \Omega} u(\mathbf{x}, T) \, d\mathbf{x}
\]

(5)

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

\[
b = \gamma^2 \|g\|^2 \int_0^T \left( \int_0^t f(s) \, ds \right) \, dt.
\]

(6)

For the PGSE, it is

\[
b = \gamma^2 \|g\|^2 \delta^2 \left( \Delta - \frac{\delta}{3} \right).
\]

(7)

Solving the complete model, i.e., Eqs. [3, 4], is challenging and many
efforts have been made. In [6, 7, 8] a simplifying assumption called the narrow
pulse approximation was used, where the pulse duration $\delta$ was assumed to be much smaller than the delay between pulses $\Delta$. 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 [4, 9, 10, 11]. The computational domain is discretized either by a Cartesian grid [4, 12, 9] or finite elements [6, 7, 8, 10, 11]. The unstructured mesh of a finite element discretization appeared to be better than a Cartesian grid in both geometry description and signal approximation [10]. For time discretization, both explicit and implicit methods have been used. In [8] a second order implicit time-stepping method called the generalized $\alpha-$method [13] was used to allow for high frequency energy dissipation. An adaptive explicit Runge-Kutta Chebyshev (RKC) method of second order was used in [9, 10]. It has been theoretically proven that the RKC allows for a much larger time-step compared to the standard explicit Euler method [14] although the RKC requires a time-step size proportional to $h^2$. In [10], 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 [11] 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 [9]. The matrices for jump conditions were explicitly calculated and imposed directly in the stiffness matrix for linear finite elements in [10]. It was generalized to allow higher orders in [11]. The pseudo-periodic boundary conditions were implemented in a finite difference method [4] and a finite volume method [9] with Cartesian grids. Its direct implementation is inefficient on a triangulation with a finite element method as discussed in [10] 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 [15, 16, 17] 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 [18] 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 the partition of unity finite element method (PUFEM) proposed in [19, 20] 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 [10, 11]. Second, we use the CN method as the time stepping method (as in [11]) 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 $\theta-$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 a 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 $Q^h$ of a Sobolev space $Q = H^1(\Omega)$.

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

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

where $(\cdot, \cdot)_{\Omega^h}$ is the standard $L_2-$inner product in $Q$, and $\Omega^h$ is the discreization
of $\Omega$ 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 $\Omega$ can be expressed as $p = (1 - \Phi) p_0 + \Phi p_1$ where $p_k \in Q$ and $p_k = \epsilon |_{\Omega_k}$ where the piecewise constant phase function $\Phi$ is defined as

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

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

Let us consider the bilinear form

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

One can, therefore, define $U_k$ on the function space $Q_k^h$ corresponding to $\Omega_k^h$ to couple the system but here we extend the solution $U_i$ to the whole domain following the PUFEM [19]. The extension facilitates better for automation and parallelization as will be clear later. So, we write

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

here $\Phi^h$ is a discretized function of $\Phi$ and it is element-wise constant.

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

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

where

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

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

In practice, the coupling Eq. (11) is singular since it contains two zero blocks. One can, therefore, add to Eq. (11) the following stabilization term [21]

$$a_{mstab}(U, v^h) = \beta \left(\Phi^h U_0, v_0^h\right)_{\Omega^h} + \beta \left((1 - \Phi^h) U_1, v_1^h\right)_{\Omega^h} \quad (13)$$
where $\beta$ is a parameter which is small enough to avoid accumulating computational errors. 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 $C$ corresponding to Eq. (13) that does not generate a large error with respect to the original matrix (right figure).

We note that although Eq. (13) is straightforward to implement as a bilinear form, $\beta$ is chosen differently for different problems [21]. The effect of $\beta$ for the convergence rate and the condition number of the derived matrix is shown in Appendix A. In our application, we choose $C$ to be the identity matrix. So,
it will take only some more memory, compared to the standard FEM, but no extra work is needed to solve the system.

3. A PUFEM for the Bloch-Torrey equation

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

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

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

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

where \( n_k \) is the normal vector pointing outward \( \Omega_k \). Now we couple Eq. (15) for the two domains to obtain

\[
\sum_{k=0}^1 \left( \frac{\partial u_k}{\partial t}, v_k \right)_{\Omega_k} = \sum_{k=0}^1 \left( -\mathcal{I} \gamma f(t) g \cdot x u_k, v_k \right)_{\Omega_k} - \sum_{k=0}^1 \left( D \nabla u_k, \nabla v_k \right)_{\Omega_k} + \sum_{k=0}^1 \int_\Gamma D \nabla u_k \cdot n_k v_k dS \tag{16}
\]

Let \( \langle a \rangle := \int_\Gamma a dS, \llbracket a \rrbracket := a_0 - a_1 \), and \( \{a\} = \frac{a_0 + a_1}{2} \), Eq. (16) becomes

\[
\sum_{k=0}^1 \left( \frac{\partial u_k}{\partial t}, v_k \right)_{\Omega_k} = \sum_{k=0}^1 \left( -\mathcal{I} \gamma f(t) g \cdot x u_k, v_k \right)_{\Omega_k} - \sum_{k=0}^1 \left( D \nabla u_k, \nabla v_k \right)_{\Omega_k} + \left\{ \left\langle D \nabla u \cdot n_0, v \right\rangle \right\} \tag{17}
\]

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

\[
\llbracket D \nabla u \cdot n^0 \rrbracket = 0
\]

\[
\{ D \nabla u \cdot n^0 \} = -\kappa [u] \tag{18}
\]
Using the fact that
\[
\llbracket a b \rrbracket = \{a\} \llbracket b \rrbracket + \llbracket a \rrbracket \{b\}
\]
we have
\[
\llbracket \langle D \nabla u \cdot n^0, v \rangle \rrbracket = \kappa \langle \llbracket u \rrbracket, \llbracket v \rrbracket \rangle
\]
(19)

Finally, substituting Eq. (19) to Eq. (17) we obtain a compact variational form for the two-compartment model as
\[
\sum_{k=0}^{1} \left( \frac{\partial u_k}{\partial t}, v_k \right)_{\Omega_k} = \sum_{k=0}^{1} \left( -I \gamma f(t) g \cdot x u_k, v_k \right)_{\Omega_k} - \sum_{k=0}^{1} \left( D \nabla u_k, \nabla v_k \right)_{\Omega_k} - \kappa \langle \llbracket u \rrbracket, \llbracket v \rrbracket \rangle
\]
(20)

4. Space-time discretization

The explicit RKC method [14] [22] is suitable for moderately stiff ordinary differential equations and was used in [9] [10] 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 [23]. As in [11], we will discretize the time domain by the \( \theta \)-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 \( u(x, t) \) for a given triangulation \( T^h \) 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 $\Phi^h$ to extend functions to the whole domain. The PUFEM corresponding to Eq. (20) with the time-stepping $\theta-$method is stated as: Find $U^n = (U^n_0, U^n_1) \in V^h$ such that

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

for all $v^h = (v^h_0, v^h_1) \in V^h$. Here $\left( a, b \right)_{\Omega_0 \cup \Omega_1} = \left( (1-\Phi^h) a_0, b_0 \right)_{\Omega_1} + \left( \Phi^h a_1, b_1 \right)_{\Omega_1}$.

The bilinear and linear forms are defined by

$$
a(U^n, v^h) = \left( \frac{U^n - U^{n-1}}{k^n}, v^h \right)_{\Omega_0 \cup \Omega_1} - \theta F(U^n, v^h, t^n)
$$

$$
L(v^h) = \left( \frac{U^{n-1} - U^n}{k^n}, v^h \right)_{\Omega_0 \cup \Omega_1} + (1 - \theta) F\left( U^{n-1}, v^h, t^{n-1} \right)
$$

where

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

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

$$
AU^n = F
$$

where

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

Here $M$ and $S$ are referred to as the mass and stiffness matrices respectively, $J$ and $I$ are corresponding to the first and third terms on the right-hand side of $F$ (Eq. 23), i.e $(g \cdot x u, v)$ and $\kappa \left\langle \left[ u \right], [v] \right\rangle$. The approximate solution we search for is discontinuous and has the form $U^n = (1 - \Phi) U^n_0 + \Phi U^n_1$. 

13
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, $\kappa^e$, similarly to [18] (see also in Appendix B). The artificial permeability condition at the external boundaries is of the form:

$$D_m \nabla u_m \cdot n_m = \kappa^e \left( u_s e^{T \theta_{ms}} - u_m \right),$$
$$D_s \nabla u_s \cdot n_s = \kappa^e \left( u_m e^{T \theta_{sm}} - u_s \right),$$

where $\kappa^e$ reflects how much the water can exchange at the boundaries and

$$\theta_{ms} = -\theta_{sm} = \gamma g \cdot (x_s - x_m) F(t).$$

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 $\kappa^e$ to approximate the pseudo-periodic boundary conditions. Based on the fact that the transverse diffusion coefficient inside the membrane layer (which is perpendicular to $\Gamma$) $D^n$ is related to the permeability $\kappa$ and the thickness $\eta$ of $\Gamma$ in the following way (see [24]):

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

we propose to use $\kappa^e = \frac{D^n}{h}$ 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 [25] (see also a review in [26] and references therein).

The stability is constrained by the CFL condition and the time-step size is inversely proportional to $\kappa^e$ when we chose to approximate $u_s(t^n) \approx U_s^{n-1}$ and $u_m(t^n) \approx U_m^{n-1}$ [18]. 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

$$D_m \nabla u_m \cdot n_m \approx \kappa^e \left( U_s^{n-1} e^{T \theta_{ms}} - U_m^n \right),$$
$$D_s \nabla u_s \cdot n_s \approx \kappa^e \left( U_m^{n-1} e^{T \theta_{sm}} - U_s^n \right).$$
This approach is more suitable for parallelization than those in [15, 16, 17] since no matrix modification is needed to impose the permeability $\kappa^e$ allowing a high-level, maintainable implementation. It also allows for non-matching meshes.

6. Implementation in FEniCS and FEniCS-HPC

FEniCS [27, 28] 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 [29, 30] 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 [31, 32, 33, 34, 35, 36].

The proposed method has been implemented both in FEniCS for moderate-scale simulations and in FEniCS-HPC for large-scale simulations. The bilinear and linear forms for the $\theta$-method (Eq. 21) can be implemented 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 $U^n = U^n_r + jU^n_i$, the real and imaginary parts are solved in a system of equations

$$
\begin{align*}
\left( \frac{U^n_r - U^{n-1}_r}{k^n}, v^n_r \right)_{\Omega_0 \cup \Omega_1} &= \left( \gamma f(t) g \cdot x U^n_{r, I}, v^n_r \right)_{\Omega_0 \cup \Omega_1} - \left( D \nabla U^n_{r, I}, \nabla v^n_r \right)_{\Omega_0 \cup \Omega_1} \\
&- \kappa \left\langle \| \nabla U^n_{r, I} \|, \| v^n_r \| \right\rangle
\end{align*}
$$

$$
\begin{align*}
\left( \frac{U^n_i - U^{n-1}_i}{k^n}, v^n_i \right)_{\Omega_0 \cup \Omega_1} &= \left( -\gamma f(t) g \cdot x U^n_{i, I}, v^n_i \right)_{\Omega_0 \cup \Omega_1} - \left( D \nabla U^n_{i, I}, \nabla v^n_i \right)_{\Omega_0 \cup \Omega_1} \\
&- \kappa \left\langle \| \nabla U^n_{i, I} \|, \| v^n_i \| \right\rangle
\end{align*}
$$

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 $\mathcal{C}$ 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 $\Omega = \Omega_0 \cup \Omega_1$

$$-\Delta u + u = p$$

(28)

where $\Omega_0$ is either a circle of radius $r = 0.5$, i.e $\Omega_0 = \{(x, y) \mid x^2 + y^2 \leq r^2\}$ 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

$$\begin{align*}
\sqrt{\nabla u \cdot n} &= g_j \\
\{\nabla u \cdot n\} &= g_a
\end{align*}$$

(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)

$$u_e = \begin{cases} 
16 \left(\frac{1}{2} - x^2\right) (4 - y^2) - 20 & \text{in } \Omega_0 \\
1 + \frac{-x^2 + y^2}{(x^2 + y^2)^{\frac{3}{2}}} & \text{in } \Omega_1
\end{cases}$$

(30)

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

$$A(U, v^h) = L(v^h), \quad \forall v^h \in V^h$$

(31)

where the bilinear and linear forms are defined as

$$A(u, v) = (1 - \Phi)\nabla u_0, \nabla v_0) + (1 - \Phi)u_0, v_0) + (\Phi \nabla u_1, \nabla v_1) + (\Phi u_1, v_1)$$

$$L(v) = (1 - \Phi)p, v_0) + (\Phi p, v_1) + \langle g_j, \{v\} \rangle + \langle g_a, [v]\rangle$$

(32)
Figure 4: The reference solutions for the test problem (Eq. 28) on \( \Omega = \Omega_0 \cup \Omega_1 \) with the flux conditions (Eq. 29). \( \Omega_0 \) is either a square \( \Omega_0 = [-0.4, 0.4]^2 \) (a) or a circle of radius \( r = 0.5 \), i.e. \( \Omega_0 = \{(x, y) | x^2 + y^2 \leq r^2 \} \) (b) which is embedded in a square \( \Omega = [-1, 1]^2 \)

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 for both cases when the linear (P1) and quadratic (P2) basis functions are considered (see 5c).

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

We perform simulations on a two-layer cylinder of radius \( R_0 = 10 \mu m \) and \( R_1 = 20 \mu m \) (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 \( \delta = 5000 \mu s \) and \( \Delta = 10,000 \mu s \) 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 \( t = 3000 \mu s \) for \( q = 0.82 \text{T/m} \).

The RKC controls the time-step sizes by a given error control \( \texttt{rtol} \) (see [22] for more details). In this simulation, we considered \( \texttt{rtol}=1e-4 \) and \( \texttt{rtol}=1e-6 \)
which correspond to the average time-step size of 104µs and 23µs respectively. The number of function evaluations varies between 225 and 2358 for \( \text{rtol=1e-4} \) and between 445 and 5718 for \( \text{rtol=1e-6} \). Three time-step sizes were considered for the Crank-Nicolson method, \( \Delta t = 30, 150, 300\mu s \) 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 \( \text{rtol=1e-4} \) and the approximation is still good up to \( q = 1.83 \text{T/m} \) for \( \text{rtol=1e-6} \) although some os-
cillations 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 $\Delta t = 30 \mu s$, 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 $\Omega = [-10 \mu m, 10 \mu m]^2$. 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 $\kappa^e = \frac{D}{\delta}$ to mimic diffusion in the free space. The reference signals are available for two extreme cases: full permeability and impermeability. The exact signal for the first case is well known as diffusion in the free space with $S = e^{-bD}$. In the latter, the reference signal is taken from the matrix formalism [23]. 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 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
Figure 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 ($q < 1 \text{T/m}$), both methods give good approximation. The RKC start oscillating with $q > 0.5 \text{T/m}$ for $\text{rtol}=1e-4$ and the approximation is still good up to $q = 1.83 \text{T/m}$ for $\text{rtol}=1e-6$ although some oscillations start appearing when $q > 1 \text{T/m}$ (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).

boundary conditions are 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 $b = 3000 \text{s/mm}^2$ (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
Figure 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 3000s/mm$^2$. The convergence is almost quadratic in both time and space.

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 [10] 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 each other for quite large time-step size $\Delta t = 500\mu$s.
Figure 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 $3000 \text{s/mm}^2$ is less than 10% (c).

### 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 [23]. 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, $\Delta t = 510, 102, \text{ and } 51 \mu$s, in comparison with the reference solution proposed in [23] (b) with $\Delta = 50 \delta = 50,000 \mu$s for three-layer structures of radii $R = [5, 7.5, 10] \mu$m (a).

Figure 10: The signals for different time-step sizes, $\Delta t = 510, 102, \text{ and } 51 \mu$s, in comparison with the reference solution proposed in [23]. 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.
Figure 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 $R = 20\mu m$ and the radii of three layers are 5, 7.5 and 10$\mu m$ respectively. Two gradient directions $\mathbf{g} = [1, 0, 0]$ and $[0, 0, 1]$ are considered. The second direction matches well the signal in three circular layers (c).

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.

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/).
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 $T_{\text{serial}}$ and timing for parallel execution $T_{\text{parallel}}$, i.e.

$$S(p) = \frac{T_{\text{serial}}}{T_{\text{parallel}}(p)}$$

(33)

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

The ideal speedup is $S_{\text{ideal}}(p) = p$, 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

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

(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).

7.6. Simulations on a pyramidal neuron

In this section, we perform simulations on a pyramidal neuron of an adult female mouse (Fig. 13a) 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 $\mu\text{m}^2$ and small dendrites with the total length of 5953.41 $\mu\text{m}$ and the average diameter of 1.2 $\mu\text{m}$. 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
Figure 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 $b = 1000 \text{s/mm}^2$ and the diffusion time $\delta = \Delta = 10000 \mu\text{s}$. 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.

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
Figure 13: A pyramidal neuron of an adult female mouse (a) downloaded from http://neuromorpho.org in which the soma surface area is 1117.83 µm² 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 \( \Omega = [-250, 250] \times [-250, 250] \times [-100, 100] \) µm³ (b) which was meshed with 1.5M vertices and 8.5M 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 principle directions for different membrane permeabilities (d). The vertical segment at each marker indicates the signal variation. The signals decay faster for higher membrane permeabilities.

\[ \delta = 10,000 \mu s \text{ and } \Delta = 50,000 \mu s \] was used with a time step size of \( \Delta t = 60 \mu s \) 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
κ = 0.5 m/s, the signal is very close to that of the fully permeable membrane which is \( S = e^{-bD} \). Here the diffusion coefficient of water \( D = 2.4 \times 10^{-3} \text{mm}^2/\text{s} \) was used for both inside the neuron and the extra-cellular space. For the impermeable case (κ = 0), the signals decay nearly 80% at \( b = 5000 \text{s/mm}^2 \). For the commonly used permeability (κ = 5 × 10^{-5} m/s), the signals decay quite a lot even for small \( b \) values. The signal decay is about 75% for \( b = 1000 \text{s/mm}^2 \) and 95% for \( b = 5000 \text{s/mm}^2 \).

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 work has been funded by the Swedish Foundation for Strategic Research, the European Research Council, and the Swedish Research Council.
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.

Appendices

Appendix A. The effect of the stabilization parameter

Fig. A.14 shows the convergence rate and the condition number for different values of the stabilization parameter $\beta$ in Eq. (13) 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,

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

As $\beta$ decreases, the convergence rate gets optimal but the condition number increases that makes the derived linear system hard to solve. So, one needs to trade-off between the convergence rate and the condition number when choosing $\beta$. 
Figure A.14: The convergence rate (a) and the condition number (b) for different values of the stabilization parameter $\beta$ in Eq. 13 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. The convergence rate gets optimal as $\beta$ gets small but the condition number becomes large. So, one needs to trade-off when choosing $\beta$.

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 $\Omega$ 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.15). The boundary needs to be periodic at $x = a_1$ and $x = b_1$. The cell interior is characterized by a diffusion tensor $D^c$. The extracellular space is the remaining part and is characterized by a diffusion tensor $D^e$. The cell touches the boundary at the interface $\Sigma$, which is the intersection between the cell boundary and $\partial \Omega$. We recall the pseudo-periodic boundary conditions for this specific domain

$$u_{a_1}^e = u_{b_1}^e e^{I \theta},$$

$$D_{a_1}^e \nabla u_{a_1}^e \cdot n_c = -D_{b_1}^e \nabla u_{b_1}^e \cdot n_c e^{I \theta}. \quad (B.1)$$
Figure B.15: When the cell interface touches $\partial \Omega$, the interface conditions and periodic boundary conditions are combined.

where

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

and the interface conditions at $\Sigma$ are

$$D_{a_1}^c \nabla u_{a_1} \cdot n_c = \kappa^c (u_{a_1}^c - u_{a_1}^e)$$

(B.2)

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

$$D_{a_1}^c \nabla u_{a_1} \cdot n_c = \kappa^e (u_{a_1}^e e^{-T \theta} - u_{a_1}^e)$$

$$D_{b_1}^e \nabla u_{b_1} \cdot n_e = \kappa^e (u_{b_1}^e e^{-T \theta} - u_{b_1}^e)$$

(B.3)

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

$$D_m \nabla u_m \cdot n_m = \kappa^e (u_s e^{T \theta_m} - u_m),$$

$$D_s \nabla u_s \cdot n_s = \kappa^e (u_m e^{T \theta_s} - u_s),$$

(B.4)

where $\kappa^e$ reflects how much water can exchange at the boundaries and

$$\theta_{m,s} = -\theta_{s,m} = \gamma g \cdot (x_s - x_m) \mathcal{F}(t).$$

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

```python
V = VectorFunctionSpace(mesh, "CG", porder);
```
\begin{verbatim}
350 u = TrialFunction(V);
v = TestFunction(V);

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

# For the source p
L = p0*v[0]*(1-Phi)*\text{dx} + p1*v[1]*Phi*\text{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
\end{verbatim}

Appendix D. FEniCS code for the $\theta$-method

Listing 1: Bilinear and linear forms

\begin{verbatim}
365 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))*\text{(1-Phi)*dx}
\end{verbatim}
\[ a_1 = \frac{u_1 r}{k} v_1 r + u_1 i / k v_1 i \ - \theta * \text{FuncF}(f_t, g_{norm}, G_X, u_1 r, u_1 i, v_1 r, v_1 i, K) * \Phi \ dx \]

\[ L_0 = \left( \frac{u_{0r 0}}{k} v_{0r} + \frac{u_{0i 0}}{k} v_{0i} + (1 - \theta) * \text{FuncF}(f_t, g_{norm}, G_X, u_{0r 0}, u_{0i 0}, v_{0r}, v_{0i}, K) \right) * (1 - \Phi) \ dx \]

\[ L_1 = \left( \frac{u_{1r 0}}{k} v_{1r} + u_{1i 0} / k v_{1i} + (1 - \theta) * \text{FuncF}(f_t, g_{norm}, G_X, u_{1r 0}, u_{1i 0}, v_{1r}, v_{1i}, K) \right) \Phi \ dx \]

\[ a_{bc} = \text{avg}( \theta \ \text{icondition}(kappa, u_{0r}, u_{1r}, v_{0r}, v_{1r}, u_{0i}, u_{1i}, v_{0i}, v_{1i})) * \text{abs}(\text{jump}(\Phi)) * dS; \]

\[ L_{bc} = \text{avg}( (1 - \theta) \ \text{icondition}(kappa, u_{0r 0}, u_{1r 0}, v_{0r}, v_{1r}, u_{0i 0}, u_{1i 0}, v_{0i}, v_{1i})) * \text{abs}(\text{jump}(\Phi)) * dS; \]

\[ \text{return } a_0 + a_1 + a_{bc}, L_0 + L_1 - L_{bc} \]

---

Listing 2: Time independent matrices to avoid repeating assembling process

```python
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;
```

---

33
I = assemble(im)
M.ident_zeros();
return M, J, S, I

References


URL https://doi.org/10.1063/1.436751


[6] H. Hagslatt, B. Jonsson, M. Nyden, O. Soderman, Predictions of pulsed field gradient NMR echo-decays for molecules diffusing in various restric-

URL http://scitation.aip.org/content/aip/journal/jcp/122/2/10.1063/1.1830432


URL http://stacks.iop.org/0031-9155/59/i=2/a=441


URL http://stacks.iop.org/0031-9155/57/i=4/a=N35

URL http://dx.doi.org/10.1115/1.2900803

URL https://doi.org/10.1007/BF01386405


URL https://www.eccomas2016.org/


URL https://doi.org/10.1016/j.apnum.2014.06.009

URL https://doi.org/10.1007/s10543-012-0417-x


URL https://doi.org/10.1007/BF01870127

URL https://doi.org/10.1007/BF02995904

URL http://dx.doi.org/10.1002/gamm.201490018


URL http://www.jneurosci.org/content/29/3/678
Direc t Finite Element Simulat ion of the Turbulent Flow Past a Parked Vertical Axis Wind Turbine

Van-Dang Nguyen\textsuperscript{a}, Johan Jansson\textsuperscript{a}, Anders Goude\textsuperscript{b}, Johan Hoffman\textsuperscript{a}

\textsuperscript{a}Department of Computational Science and Technology, KTH Royal Institute of Technology, Stockholm, Sweden
\textsuperscript{b}Ångström Laboratory, Uppsala University, Uppsala, Sweden

Abstract

There is today a significant interest in harvesting renewable energy, specifically wind energy, in offshore and urban environments. Vertical axis wind turbines get increasing attention since they are able to capture the wind from any direction, they are relatively easy to install and to transport, cheaper to build and maintain, and they are quite safe for humans and birds. Detailed computer simulations of the fluid dynamics of wind turbines provide an enhanced understanding of the technology, and may guide design improvements. In this paper we simulate the turbulent flow past a parked vertical axis wind turbine for a range of rotation angles. The forces on a parked turbine is one of the most important design cases for the survival of the turbine. We use the method of Direct Finite Element Simulation, and the simulation results are validated against experimental data in the form of force measurements. First, simulations are performed for a set of sampled rotation angles, and then a simulation is performed where the turbine is slowly rotated to cover all the rotation angles continuously. We find that the simulation results are stable with respect to mesh refinement, and that we capture the general shape of the variation of force measurements over the rotation angles.

Keywords: VAWT, Direct FEM simulation, ALE.

Email addresses: vdnguyen@kth.se (Van-Dang Nguyen), jjjan@kth.se (Johan Jansson), anders.goude@angstrom.uu.se (Anders Goude), jhoffman@kth.se (Johan Hoffman)
1. Introduction

1.1. Background

The interest in harvesting renewable energy, especially wind energy, in off-shore as well as in the urban environment has increased significantly in recent years [1, 2, 3, 4]. Vertical axis wind turbines (VAWT) get more and more attention since they are able to capture the wind from any direction, they are easy to install, easy to transport, cheaper to build and maintain, and they are quite safe to humans and birds. They are especially suitable for urban areas or areas with extreme weather. However, they are less efficient than horizontal axis turbines because only one blade is active at a time, and VAWTs also generate a relatively high degree of vibration and noise pollution. These are some of the issues that researchers address to make VAWT more popular, see e.g. [5].

Detailed computational fluid dynamics (CFD) simulations of the flow around VAWTs may give new insights, and provide guidance for design improvements. Fluid dynamics is governed by the Navier-Stokes equations, and the balance of viscous and inertial effects is determined by the Reynolds number (Re),

\[ Re = \frac{\rho U \mathcal{L}}{\mu} = \frac{U \mathcal{L}}{\nu}, \]  

where \( \rho \) is density, \( \mu \) dynamic viscosity, \( U \) a characteristic velocity scale, \( \mathcal{L} \) a characteristic length scale, and \( \nu \) kinematic viscosity \( \nu = \mu/\rho \). For high Re the flow is turbulent, which corresponds to chaotic particle trajectories and vortices on a range of scales. The main challenge of CFD is to model turbulent flow, which typically is always present in the flow around a VAWT at operational conditions.

Simulation of turbulence is difficult due to the complex mixing of spatial and temporal scales. The industry standard has long been RANS, where a statistical average of the flow is simulated, using turbulence models to model the effect of the fluctuating components of the flow. Large Eddy Simulation (LES) was developed based on the idea of approximating a filtered solution of the Navier-Stokes equations, with the effect of unresolved scales modelled in a subgrid
model. LES is very expensive, but is still a viable option to RANS in some applications. The main problem with both LES and RANS is that the subgrid and turbulence models may be problem dependent, so that model parameters must be tuned to the particular problem at hand. Direct numerical simulation (DNS) is based on full resolution of the Navier-Stokes equations, without any turbulence model or subgrid model, but with a computational cost so high that a VAWT simulation would exceed even the most powerful computers of today. We refer to the following VAWT studies for examples of LES [6, 7, 8], RANS [9, 10], and a coupled LES-RANS approach [11].

1.2. Problem formulation

In this paper we will investigate an alternative to RANS, LES and DNS, for simulation of VAWT fluid dynamics. With a focus on aerodynamics, we have developed an approach to turbulence simulation without any turbulence or subgrid model, as in LES or RANS, but avoiding the computational cost of DNS. In short, it is a Galerkin finite element method, with least-squares stabilization of the residual of the Navier-Stokes equations. Further, adaptive mesh refinement is used to minimize the computational cost of the method, which we refer to as Direct Finite Element Simulation (DFS) [12, 13, 14, 15]. The stabilization works like an automatic turbulence model which is based on the equations themselves, through the residual, without any explicit model terms, and thus do not require tuning to specific problems. Residual based turbulence simulation is becoming a popular methodology, also for turbine simulations, e.g. in the form of Variational Multiscale (VMS) methods [16].

The numerical method is implemented in the FEniCS-HPC framework [17, 18], which shows near optimal weak and strong scaling up to thousands of cores on supercomputers. To model turbulent boundary layers, we use a simple wall shear stress model where we assume the skin friction to be negligible for high Re, which corresponds to a slip velocity boundary condition. The method is combined with an Arbitrary Lagrangian-Eulerian (ALE) discretization to model a rotating turbine.
To test the DFS method for VAWT simulations, we consider a VAWT problem for which we also have experimental data in terms of force measurements \[19\]. We focus on the case of a parked turbine, which is an important design case with respect to the survival of the turbine over time, and for which force measurements are available for different angles relative to the inflow velocity. First, simulations are performed for a set of fixed sampled angles using adaptive mesh refinement, and then a simulation is performed where the turbine is slowly rotated to cover all the rotation angles continuously.

2. Method

2.1. Test case: vertical axis wind turbine

To test the DFS simulation method, we consider a 3-bladed H-rotor turbine used in \[20, 21, 19\], with a radius of 3.24m and a blade length of 5m (Fig. 1b). The blades are pitched outwards with a chord length of 0.25m at the middle of the blade. For simplification, we assume that the turbine axis is coincident with the \(z\)-axis and that the turbine \(\Omega^T\) is placed in a cylinder \(\Omega^C\) (Fig. 1a):

\[
\Omega^C = \left\{(x, y, z) \in \mathbb{R}^3 \mid x^2 + y^2 \leq R^2, z \in [0; L] \right\}. \tag{2}
\]

Figure 1: A vertical axis turbine reproduced from [21] placed in a cylinder. It is 1m high from the cylinder bottom to model the ground effect.
We set $R = 100\text{m}$ and $L = 100\text{m}$, which is large compared to the turbine size to avoid artificial blockage effects, following the recommendations in [10]:

- Minimum ratio between the distance from the turbine center to the domain inlet and outlet and the turbine diameter is 10. In our setting, the ratio is about 15.4.

- Minimum ratio between the domain width and the turbine diameter is 20. Since the ground is attached to the setting, it is still reasonable to reduce the ratio by half. In the current setting, it is about 15.4.

- The blockage ratio in this setting is about 0.16% which is smaller than 5% as recommended.

More detailed study of the blockage effect can be found in [22] [10].

The turbine axis is placed in the center-line of the cylinder domain, and it is 1m high from the bottom of the cylinder to model the ground effect (Fig. 1b).

2.2. Navier-Stokes equations

The airflow around the VAWT is modelled by the Navier-Stokes equations. For incompressible flow, the equations read

\[
\begin{align*}
\dot{u} + (u \cdot \nabla)u - \nu \Delta u + \nabla p &= f, \quad \text{in } \Omega \times I, \\
\nabla \cdot u &= 0, \quad \text{in } \Omega \times I, \\
u(t,0) &= u_0, \quad \text{in } \Omega,
\end{align*}
\]

where $u$ is the velocity, $p$ pressure and $f$ a given body force. $\Omega \subset \mathbb{R}^3$ is a spatial domain with boundary $\Gamma$, and $I = [0,T]$ a time interval.

For a moving or deforming domain, we may use an Arbitrary Lagrangian-Eulerian (ALE) method [23], which is based on the introduction of a separate set of reference coordinates. Often we let these reference coordinates trace the deformation of the finite element mesh, described by the mesh velocity $\beta$. In an ALE method, the convection term is modified to take the mesh velocity into
account, which gives the modified the Navier-Stokes equations on ALE form,

\[
\begin{aligned}
    u + \left( (u - \beta) \cdot \nabla \right) u - \nu \Delta u + \nabla p &= f, \quad \text{in } \Omega \times I, \\
    \nabla \cdot u &= 0, \quad \text{in } \Omega \times I, \\
    u(\cdot , 0) &= u_0, \quad \text{in } \Omega.
\end{aligned}
\] (4)

2.3. Finite element discretization

A Galerkin least-squares space-time finite element method (GLS) [24] is used to discretize the flow around the VAWT. Let \( 0 = t_0 < t_1 < \cdots < t_N = T \) be a time partition associated with the time intervals \( I_n = (t_{n-1}, t_n] \) of length \( k_n = t_n - t_{n-1} \). We denote the finite element space of continuous piecewise linear functions by \( Q_h \), with the derived spaces \( Q_{h,0} = \{ q \in Q_h : q(x) = 0, \ x \in \Gamma \} \) and \( V_h = [Q_{h,0}]^3 \). The ALE finite element method with least-squares stabilization is stated as: For all time intervals \( I_n \), find \((U^n_h, P^n_h) \in V_h \times Q_h\) such that

\[
\begin{aligned}
    \left( \frac{U^n_h - U_{n-1}^h}{k_n} + \left( (\bar{U}_h^n - \beta_h) \cdot \nabla \right) \bar{U}_h^n, v_h \right) \\
    + \left( \nu \nabla \bar{U}_h^n, \nabla v_h \right) - \left( P^n_h, \nabla \cdot v_h \right) + \left( \nabla \cdot \bar{U}_h^n, q_h \right) \\
    + SD_\delta \left( \bar{U}_h^n, P^n_h ; v, q \right) &= \left( f, v_h \right)
\end{aligned}
\]

for all test functions \((v_h, q_h) \in V_h \times Q_h\), where \( \bar{U}_h^n = \frac{U^n_h + U_{n-1}^h}{2} \), and \((U^n_h, P^n_h)\) is a numerical approximation of \((u, p)\) at \( t = t_n \), and with stabilization term

\[
SD_\delta(\bar{U}_h^n, P^n_h ; v, q) := \left( \delta_1 \left( (\bar{U}_h^n - \beta_h) \cdot \nabla \bar{U}_h^n + \nabla P^n_h - f_n \right), (\bar{U}_h^n - \beta_h) \cdot \nabla v_h + \nabla q_h \right) \\
+ \left( \delta_2 \nabla \cdot \bar{U}_h^n, \nabla \cdot v_h \right).
\]

Here \( \delta_1 \) and \( \delta_2 \) are given stabilization parameters:

\[
\delta_1 = C_1 \left( k_n^{-2} + |U_{n-1}^h - \beta_h|^2 h_n^{-2} \right)^{-1/2}, \quad \delta_2 = C_2 |U_{n-1}^h|^1 h_n.
\]

We note that under a CFL condition, i.e \( k_n = \frac{h_n}{|U_{n-1}^h - \beta_h|} \), \( \delta_1 \) is simplified to

\[
\delta_1 = C_1 \frac{h_n}{|U_{n-1}^h - \beta_h|}.
\]
2.4. Operational mesh velocity

We model a rotating VAWT by rotating the complete computational domain $\Omega^C$ with an operational mesh velocity $\beta_h = (\dot{x}, \dot{y}, \dot{z})$, which can be explicitly computed for a given angular velocity $\omega$, and time-step size $k$:

$$
\begin{align*}
\dot{x} &= \frac{1}{k} \left( x (\cos(\omega) - 1) - y \sin(\omega) \right), \\
\dot{y} &= \frac{1}{k} \left( y (\cos(\omega) - 1) + x \sin(\omega) \right), \\
\dot{z} &= 0.
\end{align*}
$$

(5)

2.5. Boundary layer model

Since it is not feasible to resolve a turbulent boundary layer with current computational resources, appropriate boundary conditions need to be chosen to model the effect of turbulent boundary layers without full resolution. Here we use a slip with friction and penetration with resistance boundary condition [25]:

$$
\begin{align*}
u \cdot n + \alpha n^T \sigma n &= 0, \\
u \cdot \tau_k + \beta^{-1} n^T n \tau_k &= 0,
\end{align*}
$$

(6)

where $n$ and $\tau_k$ are normal and tangential vectors ($k = 1, 2$) respectively, $\alpha$ is a penetration parameter, $\beta$ is a skin friction parameter, and $\sigma$ is the stress tensor. The no-slip condition $u = 0$ corresponds to $(\alpha, \beta) \rightarrow (0, \infty)$, and the free slip condition $u \cdot n = 0$ corresponds to $(\alpha, \beta) \rightarrow (0, 0)$.

In this paper we will use the approximation of a free slip boundary condition, which is shown in [26] to be a good approximation at high Re.

2.6. Adaptive mesh refinement

Adaptive mesh refinement algorithms provide a method for efficient use of computational resources, based on a posteriori error estimates that approximate the local contribution to the global error of all the cells in the mesh. The localization of the global error is based on the solution of an adjoint problem to the Navier-Stokes equations. The adjoint problem takes the form of a system of
convection-diffusion-reaction equations that runs backward in time, linearized at the exact velocity \( u \) and its numerical approximation \( U_h \) [27]:

\[
\begin{aligned}
-\dot{\varphi} + \left( (u - \beta) \cdot \nabla \right) \varphi + \nabla U_h^T \cdot \varphi + \nabla \theta &= \Psi, \quad \text{in } \Omega \times I, \\
\nabla \cdot \varphi &= 0, \quad \text{in } \Omega \times I, \\
\varphi(\cdot, T) &= 0, \quad \text{in } \Omega,
\end{aligned}
\]

where \( \nabla U_h^T \cdot \varphi \rangle_j = (U_h)_{j} \cdot \varphi \), and \( \Psi \) is a given weight function used to define a quantity of interest,

\[
M(\tilde{u}) = (\tilde{u}, \Psi),
\]

with of \( \tilde{u} = (u, p) \).

Let \( \tilde{U} = (U_h, P_h) \), from a standard analysis [14], it follows that

\[
|M(\tilde{u}) - M(\tilde{U})| = |(R(\tilde{U}), \tilde{\varphi})| = \sum_{K \in T_h} (R(\tilde{U}), \tilde{\varphi})_K,
\]

where \( R(\tilde{U}) \) is the residual of the Navier-Stokes equations (Eq. 3):

\[
R(\tilde{U}) = \left( R_1(\tilde{U}), R_2(\tilde{U}) \right),
\]

\[
R_1(\tilde{U}) = \dot{U}_h + \left( (U_h - \beta) \cdot \nabla \right) U_h - \nu \Delta U_h + \nabla P_h - f,
\]

\[
R_2(\tilde{U}) = \nabla \cdot U_h,
\]

\( \tilde{\varphi} = (\varphi, \theta) \) is the solution to the adjoint problem, and \( (\cdot, \cdot)_K \) is the local \( L_2(K) \) inner product,

\[
(v, w)_K = \int_K v \cdot w \, dx.
\]

The local error indicator is then defined for each element \( K \), as

\[
e^K = (R(\tilde{U}), \hat{\varphi}_h)_K,
\]

where \( \hat{\varphi}_h \) is the numerical solution of the adjoint problem.

### 2.7. Implementation

The numerical method has been implemented in the Unicorn solver [28, 29] in the FEniCS-HPC platform [17, 18] which is a high performance computing branch of FEniCS [30, 31]. This branch is optimized for massively parallel architectures, and implements duality-based adaptive error control, implicit
parameter-free turbulence modeling by the use of stabilized FEM and shows strong linear scaling up to thousands of cores [12, 32, 33, 34, 28, 35].

3. Results

3.1. Wind profiles and force coefficient

We consider two wind profiles: a constant profile \( U_x = -1 \), and a logarithmic profile (Fig. 2) defined as follows,

\[
U_x = \begin{cases} 
-\frac{1}{M} \log \left( \frac{(z - z_{min} + z_0)}{z_0} \right) & \text{if } z - z_{min} < d \\
-1 & \text{otherwise}
\end{cases}
\]

with \( M = \log \left( \frac{(d + z_0)}{z_0} \right) \) (12)

Figure 2: Logarithmic wind profile with \( z_0 = 0.025 \text{m}, z_{min} = 0, d = 16.5 \text{m (double the turbine height)}, L = 100 \text{m} \).

The force coefficient \( C_{tot} \) is computed as,

\[
C_{tot} = \frac{2 F_{tot}}{\rho U^2 \infty A}
\]

where \( \rho = 1.25 \text{kg/m}^3 \), and \( A = 1.15 \text{m}^2 \). For the constant profile \( U_{\infty} = U_x = -1 \), and for the logarithmic profile, we compute \( U_{\infty} \) to be the average of the
wind speed acting on the blades

\[ U_\infty = \frac{1}{5} \int_{3.25}^{8.25} U_x \, dz \approx -0.83 \, \text{m/s} \]

We can see that this value is quite close to the inflow velocity at the middle of the blades, i.e. \( U_x(z = 5.75\,\text{m}) \approx -0.84\,\text{m/s} \).

\( F_{\text{tot}} \) is the total force defined as

\[ F_{\text{tot}} = \sqrt{F_R^2 + F_T^2} \quad (14) \]

where \( F_R, F_T \) are the estimated radial and tangential forces from the measurements (see Fig. 3).

The drag forces are measured on one blade with its two supporting arms.

### 3.2. Boundary conditions

We choose the boundary conditions as follows:

- **Inflow**: \( u = (U_x, 0, 0) \) on \( \Gamma_{\text{in}} = \{(x, y, z) \mid x > 0 \wedge x^2 + y^2 = R^2\} \)
- **Outflow**: \( p = 0 \) on \( \Gamma_{\text{out}} = \{(x, y, z) \mid x < 0 \wedge x^2 + y^2 = R^2\} \)
- **No-slip boundary condition**: \( u = 0 \) to model the ground at the bottom of the computational domain \( z = z_{\text{min}} \).
- **The free slip boundary condition**: \( u \cdot n = 0 \) is prescribed for the turbine surface and on top of the computational domain \( z = z_{\text{min}} + L \).

### 3.3. Angle-wise validation - automated mesh adaptivity

Simulations are performed for a set of angles between 0 and 360 degrees, and Fig. 3 shows the wind directions. The blade to be measured is located at 41° from North. For each angle, the automated mesh adaptivity is used. The error indicator (Eq. 11) is computed element-wise and the mesh is refined locally by choosing 10% of the elements for refinement based on the absolute values of the error indicator.
Figure 3: Wind directions definition with the drawing of the turbine from above. The blade to be measured is located at 41° from North in the image.

Fig. 4 shows numerical approximations of the primal problem by solving Eq. (3) and the dual problem by solving Eq. (7). The effect of the automated mesh adaptivity is shown in Fig. 6 for the initial mesh with 524,686 vertices and 2,887,293 tetrahedrons, and after three iterations with 1,751,236 vertices and 9,520,104 tetrahedrons. Fig. 5 shows the wake formed downstream the turbine.

The comparison against experimental data is shown in Fig. 7 for the constant and logarithmic wind profiles.

The markers represent the simulated drag coefficients for different levels of mesh refinement, the colored curves represent the interpolations of simulated
drag coefficients, and the black curve represents the experimental data. The two different wind profiles generate slightly different drag coefficients. For example, near 0°, the log profile appears to approximate the experiment somewhat better than the constant profile does, but for angles near 180° and 270°, the situation is the opposite. For both inflow profiles, the simulations capture the general shape.
Figure 6: Automated mesh adaptivity: initial mesh with 524,686 vertices and 2,887,293 tetrahedrons (a), and after three iterations with 1,751,236 vertices and 9,520,104 tetrahedrons (b).

of the experimental drag curve, although the oscillations of the experimental drag between $120^0$ and $200^0$ are not seen clearly which could be an effect of the interpolation. We also find that the results are quite stable under different levels of mesh refinement.

Figure 7: Angle-wise drag coefficients and their interpolations for three iterations against experimental data: (a) constant wind profile, (b) logarithmic wind profile.
3.4. Validation by slow rotation - local refinement

Next, we will slowly rotate the computational domain around the $z$–axis. The mesh is locally refined in a region containing the blade with its two supporting arms (Fig. 8). We start with an initial mesh of 524,686 vertices and 2,887,293 tetrahedrons. After 4 iterations the mesh has 885,842 vertices and 4,863,350 tetrahedrons. After 8 iterations of refinement, the mesh has 1,136,739 vertices and 6,254,266 tetrahedrons. The local refinement seems to work well for this purpose and we capture quite well the shape of the force coefficient curve.

Different colored curves here represent different levels of mesh refinement, and we find that the curves are almost on top of each other that shows that the results are stable under mesh refinement. This method gives a better approximation to the experiment than the angle-wise validation does. Especially, the oscillations between $120^0$ and $200^0$ are stronger and the peak between $300^0$ and $360^0$ is more clearly seen which could be since now drag is computed for all angles, without any interpolation.

4. Conclusions and future works

The paper presents a DFS method for simulation of the turbulent flow past a VAWT, which is validated by experimental measurements. The DFS method consists of a Galerkin least-squares finite element method, coupled with an arbitrary Lagrangian-Eulerian method in order to allow us to compute high Reynolds number turbulent flow on moving meshes. The results are stable with respect to the mesh refinement and we capture well the general shape of the drag curve.

The slow-rotation strategy seems to be more efficient than the angle-wise strategy in terms of computational time, since the drag forces are valid only when the flow is fully developed. The waiting time to reach developed flow in the second case needs to be represented for each angle, whereas in the first case it is done only once. The manual local refinement is, however, less reliable than the automated mesh adaptation used in the angle-wise strategy. An efficient
Figure 8: Local mesh refinement: initial mesh with 524,686 vertices and 2,887,293 tetrahedrons (a), after 4 iterations with 885,842 vertices and 4,863,350 tetrahedrons (b), and after 8 iterations with 1,136,739 vertices and 6,254,266 tetrahedrons (c) and drag coefficient curve versus wind directions for a constant wind profile (d) and a logarithmic wind profile (e).

Automated mesh adaptation is worth considering for the rotating case. We also note that in the angle-wise strategy we may miss important local extrema since we only sample a subset of angles and interpolate in between.

In future work we will extend our study to rotating turbines. Specifically,
we will both investigate turbines with a given operational rotation speed, but also turbines that passively rotate due to the incoming wind. The bending and vibration in the blades are important phenomena, and may be predicted with a fluid-structure interaction model of the VAWT. A full fluid-structure interaction model with slip boundary conditions at the internal interfaces is under development.

**Acknowledgement**

This research has been supported by the European Research Council, the Swedish Energy Agency, Standup for Energy, 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, 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. We acknowledge the Swedish National Infrastructure for Computing (SNIC) at PDC – Center for High-Performance Computing for awarding us access to the supercomputer resource Beskow. The initial volume mesh was generated with ANSA from Beta-CAE Systems S. A., who generously provided an academic license for this project.

**References**


2. A. Tummala, R. K. Velamati, D. K. Sinha, V. Indraja, V. H. Krishna, A review on small scale wind turbines, Renewable and


doi:http://dx.doi.org/10.1016/j.cam.2014.02.020


URL http://dx.doi.org/10.1002/we.2056


doi:http://dx.doi.org/10.1016/j.compfluid.2012.02.003


URL http://cds.cern.ch/record/1644577


60th Birthday. doi:http://dx.doi.org/10.1016/j.cma.2005.05.034


