Modeling Neurodegenerative Protein Dynamics: A Numerical Approach Using the Fisher-Kolmogorov Equation

Numerical Methods for Partial Differential Equations
Academic Year 2023/2024

Project by:

Simon Hugot
Luigi Pagani
Samuele Vignolo

M.Sc. High-Performance Computing Engineering
Politecnico di Milano - Milan, Italy

May 15, 2024

Contents

Modeling Neurodegenerative Protein Dynamics: A Numerical Approach Using the Fisher-Kolmogorov Equation

Numerical Methods for Partial Differential Equations - A.Y. 2023/2024

Simon Hugot*, Luigi Pagani† and Samuele Vignolo‡
M.Sc. High-Performance Computing Engineering Politecnico di Milano - Milan, Italy
*Email: *simon.hugot@mail.polimi.it, †luigi2.pagani@mail.polimi.it, ‡samuele.vignolo@mail.polimi.it*
*Student ID: *10941153, †10677832, ‡10770345*

Abstract

This project report details the development of a numerical solver for the Fisher-Kolmogorov equation, aimed at modeling the dynamics of protein spreading in the context of neurodegenerative diseases such as Alzheimer's and Parkinson's. By employing the finite element method, the project seeks to capture the complex biological processes underlying protein aggregation and its implications on disease progression. The report is structured to first introduce the mathematical model, followed by an in-depth discussion on the numerical methods applied, including the semidiscrete and fully discrete Galerkin formulations. Subsequent sections are dedicated to the solver's implementation, stability and error analysis, and a comprehensive evaluation through a series of test cases. These cases illustrate the solver's capability to model both isotropic and anisotropic diffusion phenomena in simplified and anatomically inspired brain geometries.

1. Introduction and project goal

Neurodegenerative diseases represent a significant challenge in biomedical research, with Alzheimer's and Parkinson's among the most prevalent. These conditions are characterized by the progressive degeneration and death of nerve cells, leading to symptoms such as memory loss, cognitive decline, and impaired movement. A key feature of these diseases is the accumulation of misfolded proteins, which aggregate and spread within the brain, disrupting normal function. The Fisher-Kolmogorov equation, a nonlinear partial differential equation, provides a mathematical framework for modeling these complex biological processes. This project focuses on developing a numerical solver for the Fisher-Kolmogorov(FK) equation using the finite element method, aiming to reproduce and analyze the patterns of protein spread observed in neurodegenerative diseases by applying this model to both idealized and anatomically realistic brain geometries.

2. The Mathematical Model

In this section, we consider the FK equation to describe the reaction and diffusion of misfolded proteins. For a final time $T > 0$, the problem is dependent on time $t \in (0,T]$ and space $x \in \Omega \subset \mathbb{R}^d (d = 2,3)$. The solution to our problem represents the relative concentration of the misfolded protein $c = c(x,t)$. Indeed, under the assumption of constant baseline concentration of healthy state protein, the variable c is rescaled in the interval $[0,1]$, where 0 means the absence of misfolded proteins and 1 is a high prevalence of them. A detailed derivation of this model can be found in this paper [2]. The problem in its strong formulation reads as follows:

$$\begin{cases} \frac{\partial c}{\partial t} - \nabla \cdot (D\nabla c) - \alpha c(1-c) = 0, & \text{in } \Omega, \\ D\nabla c \cdot n = 0, & \text{on } \partial\Omega, \\ c(t=0) = c_0, & \text{in } \Omega. \end{cases} \tag{1}$$

In Eq. (1), the reaction parameter $\alpha = \alpha(x)$ represents the local conversion rate of the proteins from healthy to misfolded state. Moreover, the diffusion tensor $\mathbf{D} = \mathbf{D}(x)$ denotes the spreading of misfolded protein inside the domain (the whole brain parenchymal tissue in our case). Concerning the boundary condition, we impose only Neumann condition equal to 0 on the entire boundary $\partial\Omega$.

In the prions' spreading applications, the diffusion tensor is typically modeled as the superimposition of an extracellular diffusion effect with magnitude d_{ext} and an axonal diffusion with magnitude d_{axn}; for this reason, we assume that $\mathbf{D}$ has the following structure:

$$\mathbf{D} = d_{\text{ext}}\mathbf{I} + d_{\text{axn}}(n \otimes n), \tag{2}$$

where $n = n(x)$ is the axonal fibers direction at the point $x \in \Omega$ and $d_{\text{ext}}, d_{\text{axn}} \geq 0$. The axonal direction is the principal orientation of the connections between the neurons (axons), which can be derived from Diffusion Tensor Imaging (DTI).[1] [2]

The eigenvectors of the diffusion tensor matrix $\mathbf{D}$, denoted as v_i for $i = 1,2,3$ in a three-dimensional domain, are fundamental in elucidating the directions of protein spreading within the brain tissue. Each eigenvector corresponds to a principal axis of diffusion, with the eigenvector associated with the largest eigenvalue pointing in the direction of greatest diffusion. This direction often aligns with the axonal fibers in neural tissue, indicating that the spread of misfolded proteins is most pronounced along the paths of neuronal connections. The other eigenvectors represent directions orthogonal to this principal diffusion pathway, typically corresponding to slower diffusion rates due to the restrictive nature of the brain's microstructure.

Adopting standard notation for Sobolev spaces, we make the following assumption on the coefficients' regularity.

We assume the following regularities for the coefficients and the forcing terms appearing in (1):

- $\alpha \in L^{\infty}(\Omega)$.
- $\mathbf{D} \in L^{\infty}\left(\Omega, \mathbb{R}^{d\times d}\right)$ and there exists $d_0 > 0$ such that $\forall \xi \in \mathbb{R}^d$, we have $d_0|\xi|^2 \leq \xi^{\top}\mathbf{D}\xi$.
- $c_0 \in L^2(\Omega)$.

It can be proved that under these assumptions: $c(x,t) \in [0,1]$ for each $x \in \Omega$ and $t > 0$. In this specific setting, the equations admit two steady-state solutions: an unstable equilibrium at $c = 0$ and a stable one at $c = 1$.[1] This implies:

$$\lim_{t\to+\infty} c(x,t) = 1 \quad \text{if} \quad \exists x \in \Omega \quad \text{such that} \quad c_0(x) > 0.$$

3. Weak Formuation

The Fisher-Kolmogorov equation is used to model the diffusion and reaction of misfolded proteins within the brain, described by:

$$\begin{cases} \frac{\partial c}{\partial t} - \nabla\cdot(D\nabla c) - \alpha c(1-c) = 0, & \text{in } \Omega\times(0,T] \\ D\nabla c\cdot n = 0, & \text{on } \partial\Omega\times(0,T] \\ c(t=0) = c_0, & \text{in } \Omega\times\{0\}. \end{cases} \tag{3}$$

Multiplying both sides of Equation (3) by a test function $v \in H^1(\Omega)$ (where $H^1(\Omega)$ is the Sobolev space of functions with square-integrable derivatives on Ω) and integrating over the domain Ω, we get:

$$\int_{\Omega} v\frac{\partial c}{\partial t}\,d\Omega - \int_{\Omega} v\nabla\cdot(D\nabla c)\,d\Omega - \int_{\Omega} \alpha v c(1-c)\,d\Omega = 0. \tag{4}$$

Applying integration by parts to the diffusion term and considering the Neumann boundary condition, we have:

$$-\int_{\Omega} v\nabla\cdot(D\nabla c)\,d\Omega = \int_{\Omega} D\nabla v\cdot\nabla c\,d\Omega - \cancel{\int_{\partial\Omega} vD\nabla c\cdot n\,d\Gamma} = \int_{\Omega} D\nabla v\cdot\nabla c\,d\Omega, \tag{5}$$

where n is the outward unit normal on $\partial\Omega$ represents integration over the boundary, which goes to 0 due to the Neumann condition of our equation. By defining the residual

$$R(c)(v) = \int_\Omega v\frac{\partial c}{\partial t}\,d\Omega + \int_\Omega D\nabla v\cdot\nabla c\,d\Omega - \int_\Omega \alpha vc(1-c)\,d\Omega \quad \forall v\in H^1(\Omega) \quad \forall t\in(0,T]. \tag{6}$$

therefore, the weak form of the Fisher-Kolmogorov equation is:

find $c\in H^1(\Omega)$ such that $u(0)=u_0$ and

$$R(c)(v) = 0 \quad \forall v\in H^1(\Omega) \quad \forall t\in(0,T]. \tag{7}$$

This formulation requires the solution c to satisfy the equation in an integrated sense for all test functions v. Notice that $R(c)(v)$ is non linear in c.

4. Numerical Modeling

4.1. Semidiscrete Galerkin Formulation

To discretize the weak formulation, we begin by employing finite elements for spatial discretization and the implicit Euler method for temporal discretization. Let consider a mesh defined over the domain Ω, and let $V_h = H^1(\Omega)\cap X_h^r(\Omega)$ be the finite element space, with $N_h = dimV_h$ and $X_h^r(\Omega)$ defined as:

$$X_h^r(\Omega) = \{v_h\in C^0(\bar{\Omega}) : v_h|_{K_i}\in\mathbb{P}_r, \forall K_i\subset\Omega\}, \tag{8}$$

where $\mathbb{P}_r$ denotes the space of polynomials of degree r or less, $C^0(\bar{\Omega})$ represents continuous functions over Ω and $K_i, \quad i=1,...N$ is a partition of Ω into N subintervals. The semi-discrete formulation can be stated as follows: find $c_h\in V_h(\Omega)$ such that $u_h(0)=u_{0,h}$ and

$$R(c_h)(v_h) = 0 \quad \forall v\in V_h(\Omega) \quad \forall t\in(0,T]. \tag{9}$$

Let $\{\varphi_i\}_{i=1}^{N_h}$ be a properly basis function of the space V_h, so, instead of evaluating the Galerkin formulation for all v_h in V_h, we can just check it for each basis function φ_i, for $i=1,2,\ldots,N_h$ and the formulation reads:

find $c_h\in V_h(\Omega)$ such that $u_h(0)=u_{0,h}$ and

$$R(c_h)(\varphi_j) = 0 \quad for \quad j=1,\ldots,N_h \quad \forall t\in(0,T]. \tag{10}$$

4.2. Fully Discrete Galerkin Formulation

Now we introduce the temporal discretization. Let us partition the interval $(0,T]$ into the sub-intervals $(t_n,t_{n+1}]$, with $n=0,1,\ldots,N_T-1$, $t_0=0$, $t_{NT}=T$, and $t_{n+1}-t_n=\Delta t$. We denote with a superscript n the approximate solution at time t_n, i.e., $u_h^n\approx u_h(t_n)$.

Considering the Fisher-Kolmogorov equation with a nonlinear term, the fully discrete formulation at time step $n+1$ is given by:

$$\int_\Omega \frac{c_h^{n+1}-c_h^n}{\Delta t}\varphi_j\,dx + \int_\Omega D\nabla c_h^{n+1}\cdot\nabla\varphi_j\,dx - \int_\Omega \alpha c_h^{n+1}(1-c_h^{n+1})\varphi_j\,dx = 0, \quad for \quad j=1,\ldots,N_h, \tag{11}$$

where c_h^n is the approximation of c at time step n, Δt is the time step, and $\varphi_j \quad j=1,\ldots,N_h$ are the basis functions of the finite-dimensional subspace. The ODE system is solved over time to obtain the solution's evolution, typically using time-stepping schemes.

Newton's Method. Due to the nonlinearity, we employ Newton's method to solve the fully discrete problem. The Frechet derivative of the discrete residual $R^{n+1}(c_h^{n+1})(v_h)$ is given by:

$$a(c_h^{n+1})(\delta_h, v_h) = \int_\Omega \frac{\delta_h}{\Delta t} v_h \, dx + \int_\Omega D\nabla\delta_h \cdot \nabla v_h \, dx - \int_\Omega \left(\alpha(1-2c_h^{n+1})\delta_h\right) v_h \, dx. \tag{12}$$

The Newton iteration process at time step $n+1$ involves:

1) Given an initial guess $c_h^{n+1,(0)} = c_h^n$, iterate for $k = 0, 1, 2, \ldots$ until convergence:
2) Assemble and solve the linear problem:

$$a(c_h^{n+1,(k)})(\delta_h^{(k)}, v_h) = -R(c_h^{n+1,(k)})(v_h) \quad \forall v_h \in V_h, \tag{13}$$

where $R(c_h^{n+1,(k)})(v_h)$ represents the residual evaluated at $c_h^{n+1,(k)}$.

3) Update the solution:

$$c_h^{n+1,(k+1)} = c_h^{n+1,(k)} + \delta_h^{(k)}. \tag{14}$$

This iterative process is repeated until a specified convergence criterion is satisfied, effectively advancing the solution from time n to $n+1$.

4.3. Software implementation

Our C++ implementation leverages the deal.II library for efficient finite element computations in the context of solving nonlinear partial differential equations. The solver class has several main methods:

- **Solver Setup**: The `setup` method initializes key components for the finite element solver, including mesh creation, finite element space setup, degree of freedom (DoF) handling, and linear system initialization. It establishes the foundation for solving the finite problem by configuring mesh, finite element space, DoF handler, and linear system components.
- **Assembly of the System Matrices**: The `assemble_system` method constructs the system matrices for the finite element solver. It iterates over the active cells, computing the element-wise contributions to the matrix and the residual vector (`residual_vector`). We have to assemble:
  - $A(c_h^{k+1})_{ij} = a(c_h^{k+1})(\varphi_j, \varphi_i) \quad \text{for} \quad i, j = 1 \ldots, N_h$
  - $(r(u))_i = -R(c_h^{k+1})(\varphi_i) \quad \text{for} \quad i = 1 \ldots, N_h$

  The matrices are formed by evaluating the solution, its gradient, and other terms at quadrature points within each cell and the integrals are approximated by means of a quadrature formula. The resulting contributions are accumulated to form the global matrix and residual vector. The final matrices are compressed for efficient storage and computation. The overall effect of this process is to build the matrices necessary for solving the finite element problem, incorporating the effects of time discretization, diffusion, and non-linear terms in the governing equations.
- **Solver for Linear System**: It is trivial to show that the A matrix assembled is symmetric, this is because

$$A(c_h^{k+1})_{ij} = a(c_h^{k+1})(\varphi_j, \varphi_i) = a(c_h^{k+1})(\varphi_i, \varphi_j) = A(c_h^{k+1})_{ji} \quad for \quad i, j = 1 \ldots, N_h, \tag{15}$$

  so the `solve_linear_system` method is responsible for solving the linear system using the Conjugate Gradient (CG) method with a Symmetric Successive Overrelaxation (SSOR) preconditioner. It employs a solver control mechanism to control the maximum number of iterations and the residual tolerance. The key steps of the method include:
  - Initialization of solver control with a specified maximum number of iterations and residual tolerance.
  - Creation of a CG solver and an SSOR preconditioner.
  - Solution of the linear system using the CG solver with preconditioning.
  - Output of the number of CG iterations performed during the solution process.

This method ensures the convergence of the linear system solution within the specified tolerance, providing insights into the efficiency of the iterative solver.

- **Newton's Method**: The `solve_newton` method applies Newton's iterative approach for solving nonlinear systems (see 4.2). It refines the solution iteratively until a specified tolerance is achieved or a maximum iteration limit is reached. Key steps involve setting iteration parameters, assembling the nonlinear system, calculating the residual norm, and updating the solution. The stopping condition is based on the modulus of the residual, which must be below a predefined threshold. This method offers an effective and efficient solution for nonlinear problems.
- **Output**: The `output` method generates visualization data for the solution at a given time step. It includes the solution vector "u" and information about the partitioning of the mesh. The data is written in VTU format with a PVTU record for parallel visualization.
- **Time Stepping**: The `solve` method implements the time-stepping algorithm for solving the time-dependent problem. It starts by applying the initial condition, then iteratively advances in time using the Newton's method (4.3) to solve the nonlinear problem at each time step. The solution is stored and visualized, providing a comprehensive approach for solving time-dependent nonlinear systems.
- **Error Computation**: The `compute_error` method calculates the error of the numerical solution by comparing it with the exact solution. It utilizes finite element spaces, mappings, and quadrature rules to perform the integration of the difference between the numerical and exact solutions. The computed error is then evaluated globally using the specified norm type, providing a quantitative measure of the accuracy of the numerical solution.

4.3.1. Parallelization Strategy. To enable efficient solution of problems on large meshes, the code is parallelized using domain decomposition techniques from the `deal.II` library. Specifically, the mesh and associated degree of freedom handlers are distributed across multiple processes using `GridTools::partition_triangulation`.

Each process operates on a subset of elements and owns only the degrees of freedom associated with its local mesh partition. Ghost cells are introduced on partition boundaries to correctly account for inter-process communication requirements. The linear system assemble routines utilize threading for on-node parallelism and MPI for inter-node data exchanges.

The global residual vector and solution vector are distributed, with each process holding only its local contributions in `TrilinosWrappers::MPI::Vector` objects. similarly, the global sparse matrix is stored block-wise across processes using the `TrilinosWrappers::SparseMatrix` class. This facilitates seamless integration with Trilinos' parallel linear solvers and preconditioners.

Mesh partitioning and DoF distribution occurs only once during code initialization. In each Newton iteration, assembly is performed locally on each partition, followed by global communications to sum contributions across processes. The resulting linear system is then solved in parallel using a preconditioned conjugate gradient method from Trilinos. Finally, the converged update is added to the local solution to conclude the nonlinear step.

Overall, this strategy allows large problems to be solved on computer clusters by distributing work evenly while minimizing communication overheads. The implementation demonstrates good weak scaling, with runtimes remaining fairly constant as more processes are introduced to solve larger meshes. The shared and distributed memory parallelism makes the code well-suited for modern high performance computing architectures.

5. Analysis of Polynomial Degree Effects on Solution Quality with near singular matrices

Our examination into the stability and accuracy of numerical solutions for partial differential equations revealed nuanced interactions between the polynomial degree of finite elements and the characteristics of the diffusion tensor. Specifically, we employed a diagonal diffusion tensor:

$$\mathrm{D} = \begin{pmatrix} 0.001 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 0.001 \end{pmatrix},$$

and conducted analyses over various time steps, specifically $\Delta t = 0.05$ and $\Delta t = 0.2$. Our findings indicate that for complex diffusion tensors like the one above, finite elements with a lower polynomial degree, specifically 1, tend to yield solutions that adhere more closely to the expected physical and theoretical constraints, particularly keeping the solution values within the $[0,1]$ range.

In contrast, when applying finite elements to a scenario with an identity diffusion matrix on a $2D$ unitary square mesh, where the exact solution is known, the use of higher-degree polynomials enhances the convergence rate of the numerical method. This suggests a contrasting effect: while higher polynomial degrees can improve convergence in more straightforward or well-understood problems, they may not be as effective or might even introduce inaccuracies in scenarios involving more complex diffusion behaviors.

Thus, the choice of polynomial degree in finite elements is not one-size-fits-all but should be informed by the specific characteristics of the diffusion tensor and the underlying physical phenomena being modeled. For complex diffusion scenarios, lower polynomial degrees may provide more reliable and physically consistent solutions, whereas for simpler cases or when the exact solution is known, higher degrees can offer computational advantages through improved convergence rates.

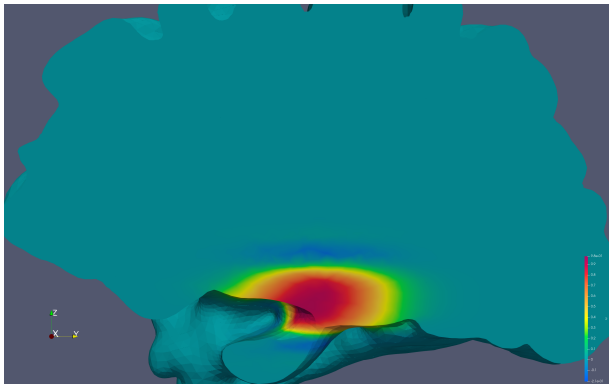

Figure 1. Solution viewed at $T = 2.6$, with polynomials of degree 1 and time step $\Delta t = 0.2$

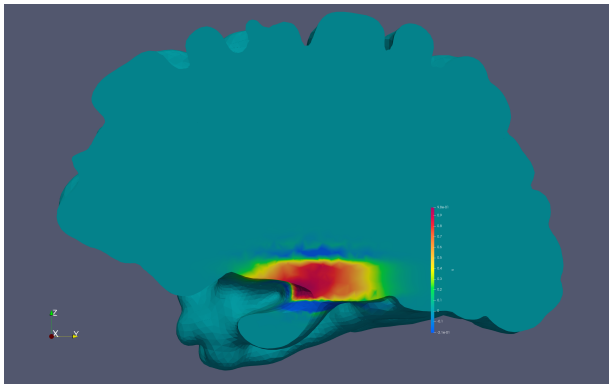

Figure 2. Solution viewed at $T = 2.6$ years, with polynomials of degree 2 and time step $\Delta t = 0.2$

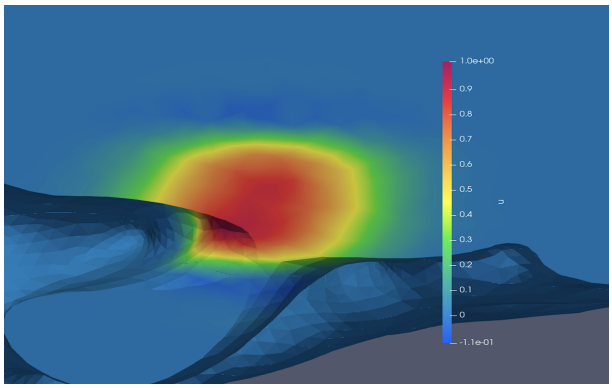

Figure 3. Solution viewed at $T = 1$, with polynomials of degree 1 and time step $\Delta t = 0.05$

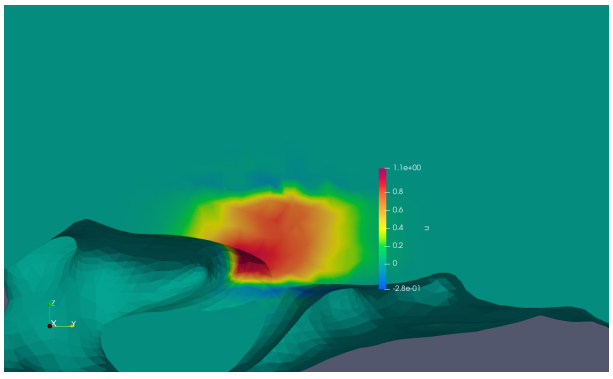

Figure 4. Solution viewed at $T = 1$, with polynomials of degree 2 and time step $\Delta t = 0.05$

6. Analysis of Polynomial Degree Effects on Newton Convergence

Our numerical experiments, conducted on both a 3D triangular mesh of a brain hemisphere and a unitary cubic mesh, have highlighted the impact of the polynomial degree on the efficiency and robustness of the Newton method in solving the discretized Fisher-Kolmogorov equation for modeling neurodegenerative protein dynamics. In particular, we have observed that using a polynomial degree of $p = 1$ leads to better convergence properties of the Newton method compared to degree $p = 2$, especially for large time steps Δt.

7. Numerical Results

7.1. Test Case 1: Convergence Analysis in a 2D Case

We conduct a convergence analysis in a two-dimensional setting to ascertain the method's accuracy. The domain Ω is taken as $(0,1)^2$, and we discretize it using meshes with different resolutions. The time discretization is performed with a timestep $\Delta t = 10^{-9}$, considering a maximum time $T = 10^{-8}$. The exact solution for this test case is designed as:

$$c(x,y,t) = (\cos(\pi x)\cos(\pi y) + 2)e^{-t}. \tag{16}$$

We assume $\alpha = 0.1$.

Figures 5 and 6 display the errors in the L^2 norm and H^1 norm, at the final time $T = 10^{-8}$. We observe that the convergence rates in the H^1 norm aligns with the polynomial degrees of the approximation space.

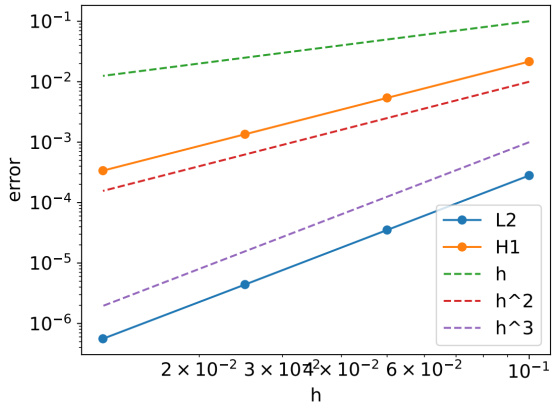

Figure 5. Error convergence with respect to the polynomial order $p = 1$

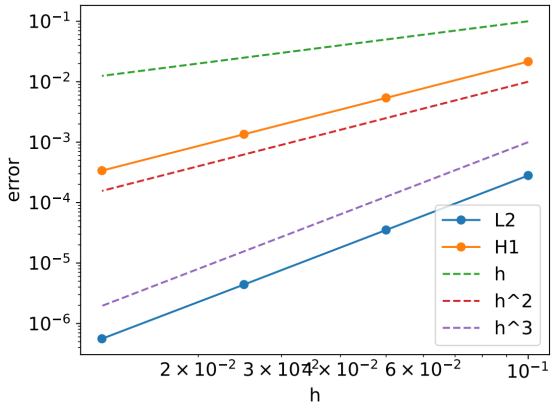

Figure 6. Error convergence with respect to the polynomial order $p = 2$

7.2. Test Case 2: Isotropic Diffusion on a 3D Triangular Mesh of Brain Hemisphere

This study investigates the isotropic diffusion process in a 3D triangular mesh that represents a brain hemisphere. The Fisher-Kolmogorov equation is applied, setting the diffusion parameter α to 0.9/year and utilizing an identity matrix for isotropic diffusion. The simulation ran over a period of one year (T = 1 year) with a time increment (Δt) of 0.01 year. The initial condition simulated a concentrated area of misfolded proteins within the brain mesh.

Isotropic diffusion implies that the spread of misfolded proteins occurs uniformly in all directions, suggesting a homogenous and ideal tissue environment for diffusion. The Fisher-Kolmogorov equation's weak form was

discretized with linear finite elements, and protein concentrations were calculated at each time increment. The visualization showcases the protein distribution initially and after one year to illustrate the dispersion of the aggregates.

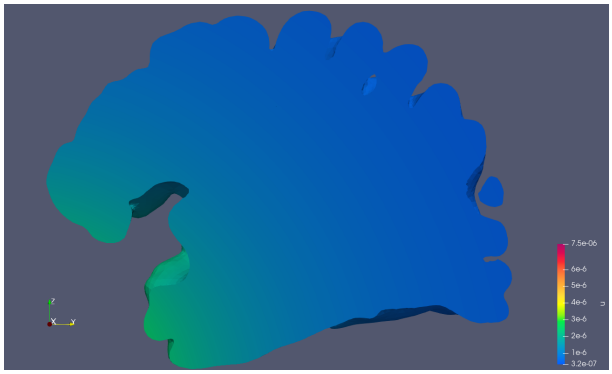

Figure 7. Initial concentration of misfolded proteins in brain hemisphere model, demonstrating localized distribution

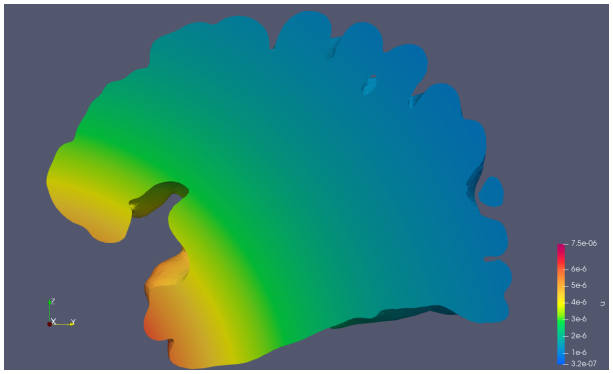

Figure 8. Concentration of misfolded proteins after one year of isotropic diffusion, illustrating uniform dispersion

7.3. Test Case 3: Non Isotropic Diffusion on a 3D Triangular Mesh of Brain Hemishpere

In this test case, we examine the effects of anisotropic diffusion characterized by the diffusion matrix:

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

which signifies predominant diffusion along the y-axis, while diffusion along the x and z axes is negligible.

This configuration leads us to anticipate a more rapid dispersion of misfolded proteins along the y-direction, resulting in a distinctively elongated concentration profile over time, due to the directional bias in diffusion.

The simulation is configured with parameters such as $\alpha = 0.8$/year, a time step of $\Delta t = 0.002$ year, and a total simulation duration of $T = 2$ years. Initially, the proteins are distributed within a cubic region, illustrating that diffusion predominantly occurs along the y-axis.

The anisotropic nature of diffusion in this scenario causes a directional and non-uniform spread of the protein concentration, highlighting the model's ability to simulate complex diffusion dynamics within biological tissues. This effect is particularly evident when observing the dispersion from a centrally located cubic region within a simulated brain structure.

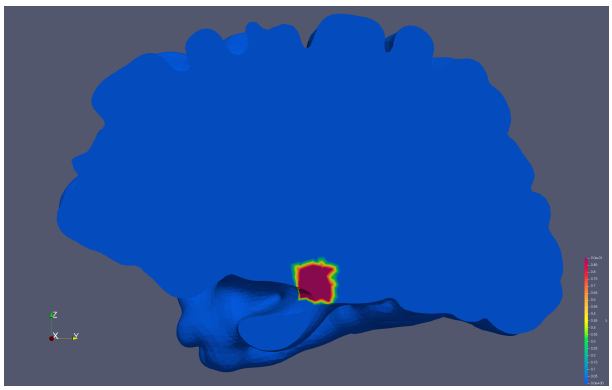

Figure 9. X-axis view at $T = 0$

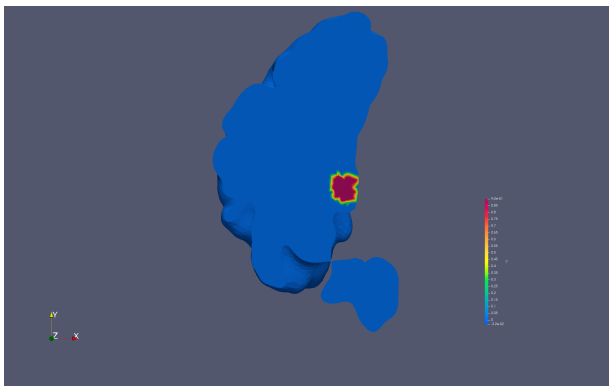

Figure 10. Z-axis view at $T = 0$

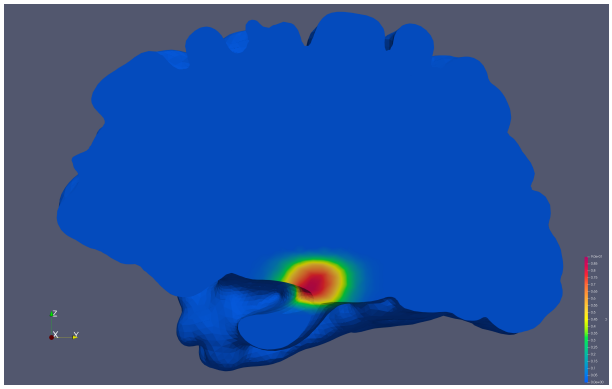

Figure 11. X-axis view at $T = 1$

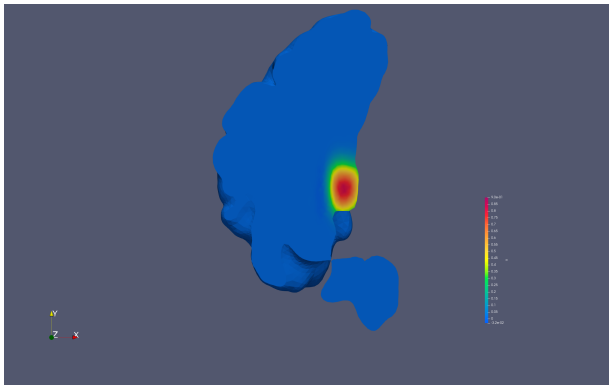

Figure 12. Z-axis view at $T = 1$

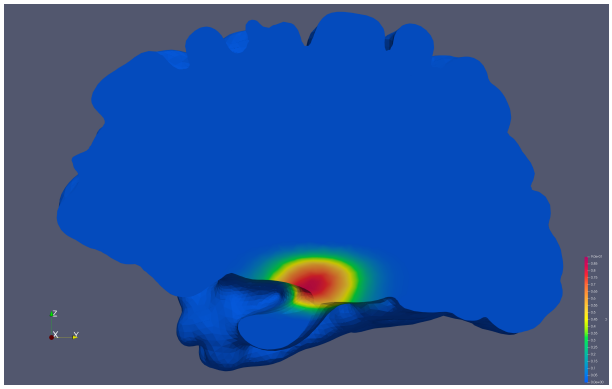

Figure 13. X-axis view at $T = 2$

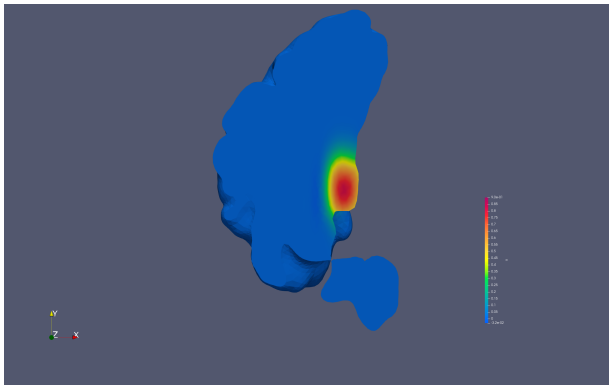

Figure 14. Z-axis view at $T = 2$

References

[1] Mattia Corti et al. “Discontinuous Galerkin methods for Fisher–Kolmogorov equation with application to -synuclein spreading in Parkinson’s disease”. In: *Computer Methods in Applied Mechanics and Engineering* 417 (2023), p. 116450. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2023.116450. URL: https://www.sciencedirect.com/science/article/pii/S0045782523005741.

[2] Johannes Weickenmeier et al. “A physics-based model explains the prion-like features of neurodegeneration in Alzheimer’s disease, Parkinson’s disease, and amyotrophic lateral sclerosis”. In: *Journal of the Mechanics and Physics of Solids* 124 (2019), pp. 264–281. ISSN: 0022-5096. DOI: https://doi.org/10.1016/j.jmps.2018.10.013. URL: https://www.sciencedirect.com/science/article/pii/S0022509618307063.