Fisher-Kolmogorov equations for neurodegenerative diseases

Andrea Boin, Giacomo Pauletti, Lorenzo Pettenuzzo

Contents

1 Introduction

The objective of this project is to apply various numerical methods to solve the Fisher-Kolmogorov equations to reproduce the results of the paper [1]. The Fisher-Kolmogorov equations can be used to effectively model the spread of misfolded proteins in the brain, a process associated with numerous neurodegenerative diseases.

1.1 Fisher-Kolmogorov equation

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

- c: concentration of the misfolded protein in a region of the brain ($0 \leq c \leq 1$)
- α: constant of concentration growth
- D: diffusion coefficient of the misfolded protein.
  It can be isotropic (a scalar) or anisotropic (a square matrix).
  In case of anisotropic coefficient the term can be computed as:

  $$\underline{\mathrm{D}} = d^{\mathrm{ext}}\underline{\mathrm{I}} + d^{\mathrm{axn}}(\mathbf{n} \otimes \mathbf{n})$$

  where d^{ext} is the extracellular diffusion term, d^{axn} is the axonal diffusion term and $\mathbf{n}$ the direction of axonal diffusion.

  Usually extracellular diffusion is slower than axonal diffusion: $d^{\mathrm{ext}} < d^{\mathrm{axn}}$.

  The Fisher-Kolmogorov equation is a **diffusion-reaction** equation with a nonlinear forcing term that can be used to model population growth. In this case it is used to model the spreading of proteins in the brain.

  The interested problem is a **nonlinear parabolic PDE** with **Neumann boundary conditions**.

1.2 Mesh

The mesh we used for the simulation is a 3D representation of a hemisphere of the human brain with 21211 points and 42450 cells.
To process the mesh with our software, we did convert the format from *.stl* to *.msh* using **GMSH** with the following procedure:

1. Import the mesh (*.stl*) in GMSH
2. From the left menu, select "geometry → add → volume"
3. Save the new generated *.geo* file
4. Define the 3D mesh: "mesh → define → 3D"
5. Export the file as *.msh*: "file → export → msh"

The mesh isn't associated to any function that could provide information about axonal orientation so an anisotropic model can't be used to accurately simulate the evolution of the system. Also there is no distinction between white and grey matter in the mesh. Different kinds of matter have different reaction coefficients so we used an average of the two for our simulations. We used an initial seed based on the function

$$c(x,y,z,t=0) = \begin{cases} 0.1 & z < 15 \\ -0.02z + 0.4 & 15 < z < 20 \\ 0 & \text{otherwise} \end{cases}$$

The value 0.1 is the same value used in [1]. We used a decreasing ramp to avoid a step function as initial condition and obtain a continuous function.

1.3 Weak formulation and semi-discretized formulation

By choosing a domain $V = H^1 = \{v \in L^2 | \nabla v \in L^2\}$ and considering a time domain $(0,T)$, the weak formulation of the problem is:

Find $c(t) \in V$ such that $\forall v \in V$ and $\forall t \in (0,T)$:

$$\begin{cases} \int_\Omega \frac{\delta c}{\delta t} v d\Omega + \int_\Omega D \nabla c \nabla v d\Omega - \int_\Omega \alpha c(1-c) v d\Omega = 0 \\ c(t=0) = c_0 \end{cases}$$

By renaming:

- $a(c, v) = \int_\Omega D\nabla c \nabla v d\Omega$
- $n(c, v) = -\int_\Omega \alpha c(1-c)v d\Omega$

By introducing a triangulation $T_h = \{K|\Omega = \bigcup K\}$ of the domain Ω and defining with it a polynomial space

$$X_h = \{v_h \in C^0(\bar{\Omega})|v_{h|k} \in \mathbb{P}^r(K), \forall K \in T_h\}$$

we can obtain the discrete space $V_h = V \cap X_h$ for our discrete formulation. The semi-discrete formulation can then be written as:

Find $c_h \in V_h$ such that, $\forall v_h \in V_h$ and $\forall t \in (0, T)$:

$$\int_\Omega \frac{\delta c_h}{\delta t} v_h d\Omega + a(c_h, v_h) + n(c_h, v_h) = 0$$

$$c_h(t = 0) = c_{h,0}$$

2 Methods

We studied the problem with 3 methods and implemented 2 of them algorithmically:

- An **explicit** scheme in which all terms have been treated explicitly to handle the nonlinear part of the model.
- A **mixed explicit/implicit** scheme in which the linear terms have been treated implicitly while the nonlinear terms explicitly to get rid of nonlinarities.
- An **implicit** scheme in which all terms in the equation have been treated implicitly, and then the nonlinear parts have been solved with the Newton method.

To obtain a full discretization of the problem we need to partition the time domain in N partitions of size Δt, obtaining $(0, T) = (0, N\Delta t) = \bigcup_{n=1}^{N}(t^n, t^{n+1}]$ where $t^{n+1} - t^n = \Delta t$, $t^0 = 0$ and $t^N = T$. We can then use an upper-index notation to identify time dependent elements: $c^n = c(t^n)$.

2.1 Explicit scheme

The fully discrete formulation for the **explicit** scheme becomes:

Find $c_h(t) \in V_h$ such that, $\forall v_h \in V_h$, $c_h^0 = c_{h,0}$ and $\forall n \in \{0, N\}$:

$$\int_\Omega \frac{c_h^{n+1} - c_h^n}{\Delta t} v_h d\Omega + a(c_h^n, v_h) + n(c_h^n, v_h) = 0$$

By introducing a basis $\{\phi_i\}$ for the space V_h the problem can be written as:

Find $c_h(t) \in V_h$ such that $c_h^0 = c_{h,0}$ and $\forall n \in \{0, N\}$:

$$Mc^{n+1} = F^n$$

where the **mass matrix** can be computed as:

$$M_{ij} = \frac{1}{\Delta t}\langle \phi_j, \phi_i \rangle$$

and the **forcing term** is:

$$F_i^n = \frac{1}{\Delta t}\langle c_h^n, \phi_i \rangle - a(c_h^n, \phi_i) - n(c_h^n, \phi_i)$$

2.1.1 Stability and Accuracy

The accuracy is $O(\Delta t)$ for time and $O(h^2)$ for space.
The stability condition of the explicit scheme is: $\Delta t \leq \min(\frac{h^2}{2D}, \frac{2}{\alpha})$. The first term acts as a bottleneck for the method. With our values for example $(h = 1[cm], D = 1.5[cm/year], \alpha = 0.5[1/year])$, $\Delta t \leq \min(\frac{1}{3}, 4) = \frac{1}{3}$. The following methods allow for a larger choice of Δt and are generally faster so we decided to implement them and not implement the explicit version.

2.2 Mixed explicit/implicit scheme

The full discretization for the **mixed explicit/implicit** scheme is:

Find $c_h \in V_h$ such that, $\forall v_h \in V_h$ and $c_h(t = 0) = c_{h,0}$:

$$\int_\Omega \frac{c_h^{n+1} - c_h^n}{\Delta t} v_h d\Omega + a(c_h^{n+1}, v_h) + n(c_h^n, v_h) = 0$$

The problem can be rewritten, by introducing a basis $\{\phi_i\}$ for V_h as:

Find $c_h(t) \in V_h$ such that $c_h^0 = c_{h,0}$ and $\forall n \in \{0, N\}$:

$$Mc^{n+1} = F^n$$

where the **mass matrix** can be computed as:

$$M_{ij} = \frac{1}{\Delta t}\langle \phi_j, \phi_i \rangle + a(\phi_j, \phi_i)$$

and the **forcing term** is:

$$F_i^n = \frac{1}{\Delta t}\langle c_h^n, \phi_i \rangle - n(c_h^n, \phi_i)$$

2.2.1 Algorithm

The assembly of the M matrix is implemented by the following algorithm:

Algorithm 1: Mixed left-side matrix assemble

```
for cell in cells_iterator do
    for quadrature_node in cell do
        for $i = 0, i <$dofs_per_cell$; i++$ do
            for $j = 0, i <$dofs_per_cell$; j++$ do
                $M_{ij} = \Delta t^{-1} * \phi_j * \phi_i * d\Omega + a(\phi_j, \phi_i)$;
            end
        end
    end
end
```

To assemble the F^n vector we can use the precomputed value of c in the following algorithm:

Algorithm 2: Mixed right-side forcing term

```
for cell in cells_iterator do
    for quadrature_node in cell do
        for $i = 0, i <$dofs_per_cell$; i++$ do
            $F_i^n = \Delta t^{-1} * c_h^n * \phi_i * d\Omega - n(c_h^n, \phi_i)$;
        end
    end
end
```

Then the problem can be solved by applying a linear system solver.

2.2.2 Stability and Accuracy

The accuracy for this method is the same as the explicit one: $O(\Delta t)$ for time and $O(h^2)$ for space.
The stability condition though is better: $\Delta t \leq \frac{2}{\alpha} = 4$ allowing for a larger choice of Δt and a quicker convergence.

2.3 Implicit scheme

For the **implicit** scheme the fully discretized formulation is:

Find $c_h \in V_h$ such that, $\forall v_h \in V_h$ and $c_h(t=0) = c_{h,0}$:

$$\int_\Omega \frac{c_h^{n+1} - c_h^n}{\Delta t} v_h d\Omega + a(c_h^{n+1}, v_h) + n(c_h^{n+1}, v_h) = 0$$

In this case a nonlinear system of equations has to be solved. By renaming:

$$R^{n+1}(c_h^{n+1}, v_h) = \int_\Omega \frac{c_h^{n+1} - c_h^n}{\Delta t} v_h d\Omega + a(c_h^{n+1}, v_h) + n(c_h^{n+1}, v_h)$$

we can solve the following nonlinear problem:
Find $c_h \in V_h$ such that, $\forall v_h \in V_h$ and $c_h^0 = c_{h,0}$:

$$R^{n+1}(c_h^{n+1}, v_h) = 0$$

The problem can be solved with the Newton method. The Frechet derivative of the residual is:

$$a(c)(\delta, v) = \int \frac{d\delta}{dt} v d\Omega + \int D\nabla\delta\nabla v d\Omega - \int \alpha(1-2c)\delta v dx$$

and it can be used in the algorithm to compute the solution.

2.3.1 Algorithm

For the solution of this problem two loops are required: the outer one loops over time and the inner one loops over the iterations of the Newton method.

Algorithm 3: Nonlinear solver

```
$n = 0$;
**while** *stopping criteria over n* **do**
    $k = 0$;
    **while** *stopping criteria over k* **do**
        Solve $a(c_h^{n+1,(k)})(\delta^k, v_h) = -R(c_h^{n+1,(k)}, v_h)$;
        Update $c_h^{n+1,(k+1)} = c_h^{n+1,(k)} + \delta^k$;
    **end**
**end**
```

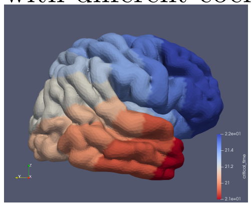
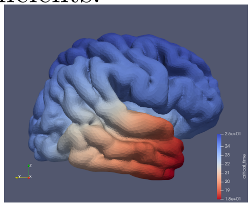
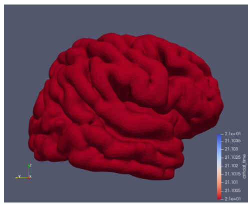
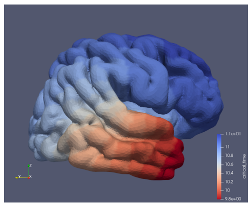
In the inner loop a matrix and a vector are assemble and then the linear system is solved.

2.3.2 Stability and Accuracy

The implicit method is stable for any choice of Δt and has an accuracy quadratic in h: $O(h^2)$.

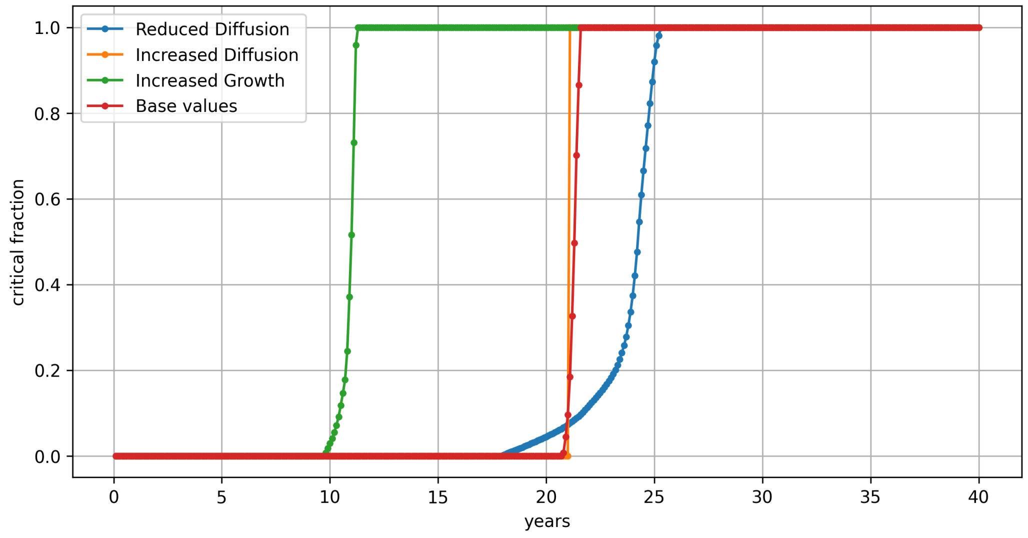
2.3.3 Results

The following images contain the results of the simulation at $t = 20[years]$ with different coefficients.

3 Results and algorithmic comparation

All the algorithms where implemented using Deal.II functions for parallelism. An interesting result we obtained from our simulations is that the concentration of proteins in the brain has approximately the same value in each region, so if the protein begins to be harmful at a specific concentration it quickly begins to affect the whole brain in a really short span of time. This correctly predicts the real examples of people affected by neurodegenerative diseases that have a really quick loss of brain functions once the disease begins to show it's signs. The following graph shows the percentage of zones that reached a critical level in the brain. As it can be seen in the graph, the critical threshold is reached with a graph similar to a sigmoid with a really steep slope, indicating a quick degeneration for the disease.



References

[1] J. Weickenmeier, M. Jucker, A. Goriely, and E. Kuhl. A physics-based model explains the prion-like features of neurodegeneration in Alzheimer's disease, Parkinson's disease, and amyotrophic lateral sclerosis. Journal of the Mechanics and Physics of Solids, 124:264–281, 2019.