Fisher-Kolmogorov equation 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 equation to reproduce the results of the paper [1]. The Fisher-Kolmogorov equation is usually used to model population dynamics and can be effectively used to model the spread of misfolded proteins in the brain, a process associated with numerous neurodegenerative diseases.

1.1 Fisher-Kolmogorov equation

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

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

- c: concentration of the misfolded protein in a region of the brain ($0 \leq c \leq 1$).
- α: constant of concentration growth, dependent on the velocity misfolded proteins reproduce by affecting healthy proteins.
- D: diffusion coefficient of the misfolded protein, dependent on the diffusion media.
  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 reaction term that can be used to model population growth. In this case it is used to model the spreading of proteins in the brain.

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 converted 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"

We obtained a minimum dimension for a segment of an element: $h_{\min} = 0.12\text{mm}$.

1.3 Assumptions

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.
Hence we used the value $D = 150\ \text{mm}^2/\text{year}$, as written in [1].
Also there is no distinction between white and grey matter in the mesh. Different kinds of matter have different reaction coefficients so we used a combination of the two for our simulations. So we obtained $\alpha = 0.5\ \text{year}^{-1}$, grossly averaging $\alpha = 0.3$, for grey matter, and $\alpha = 0.6$, for white matter, as these values are found in [1].

We used an initial seed based on the function

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

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.

1.4 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:

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

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

$$X_h = \{v_h \in C^0(\bar{\Omega}) | v_{h|k} \in \mathbb{P}^1(K), \forall K \in T_h\}$$

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:

For each $t \in (0, T]$ find $c_h \in V_h$ such that:

$$\int_\Omega \frac{\delta c_h}{\delta t} v_h d\Omega + d(c_h, v_h) + e(c_h, v_h) = 0 \qquad \forall v_h \in V_h$$

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

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 remove nonlinearities.
- An **implicit** scheme in which all terms in the equation have been treated implicitly; a nonlinear algebraic system is obtained in the end, so it is 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 $(0, T) = (0, N\Delta t) = \bigcup_{n=0}^{N-1}(t^n, t^{n+1}]$ 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:

Forall $n \in \{0, 1, \ldots, N-1\}$, find $c_h^{n+1} \in V_h$ such that:

$$\int_\Omega \frac{c_h^{n+1} - c_h^n}{\Delta t} v_h d\Omega + d(c_h^n, v_h) + e(c_h^n, v_h) = 0 \qquad \forall v_h \in V_h$$

given that $c_h^0 = c_{h,0}$.
We introduce the basis $\{\phi_i\}$ for the space V_h and we write c_h^{n+1} as $\sum_{j=1}^{N_h} c_j^{n+1}\phi_j$.
We can then group the coefficients into the vector $\mathbf{c}^{n+1}$.
Then, the problem can be written as:

Forall $n \in \{0, 1, \ldots, N-1\}$ find $\mathbf{c}^{n+1} \in V_h$:

$$\mathbf{M}\mathbf{c}^{n+1} = \mathbf{F}^n$$

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

$$\mathbf{M}_{ij} = \frac{1}{\Delta t}\langle \phi_j, \phi_i \rangle$$

and the **forcing term F** is:

$$\mathbf{F}_i^n = \frac{1}{\Delta t}\langle c_h^n, \phi_i\rangle - d(c_h^n, \phi_i) - e(c_h^n, \phi_i)$$

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 we obtain $\Delta t \leq \min(0.0048, 4) = 0.0048$. With such a restrictive limitation over the timestep, more than 8000 are needed to simulate over a time of 40 years, making it unfeasible for simulations on our machines. The following methods allow for a larger choice of Δt so we decided to not implement the explicit version.

2.2 Mixed explicit/implicit scheme

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

For all $n \in \{0, 1, \ldots, N-1\}$ find $c_h^{n+1} \in V_h$ such that:

$$\int_\Omega \frac{c_h^{n+1} - c_h^n}{\Delta t} v_h d\Omega + \int_\Omega D\nabla c_h^{n+1} \nabla v_h d\Omega - \int_\Omega \alpha c_h^{n+1} v_h d\Omega + \int_\Omega \alpha (c_h^n)^2 v_h d\Omega = 0$$

forall $v_h \in V_h$, given that $c_h^0 = c_h(t=0)$.
We introduce the basis $\{\phi_i\}$ for the space V_h and we write c_h^{n+1} as $\sum_{j=1}^{N_h} c_j^{n+1}\phi_j$.
We can than group the coefficients into the vector $\mathbf{c}^{n+1}$.
Then, the problem can be written as:

Forall $n \in \{0, 1, \ldots, N-1\}$ find $\mathbf{c}^{n+1} \in V_h$:

$$\mathbf{M}\mathbf{c}^{n+1} = \mathbf{F}^n$$

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

$$\mathbf{M}_{ij} = \frac{1}{\Delta t}\int_\Omega \phi_j\phi_i d\Omega + \int_\Omega D\nabla\phi_j\nabla\phi_i d\Omega - \int_\Omega \alpha\phi_j\phi_i d\Omega$$

and the **forcing term** is:

$$\mathbf{F}_i^n = \frac{1}{\Delta t}\int_\Omega c_h^n\phi_i d\Omega - \int_\Omega \alpha (c_h^n)^2\phi_i d\Omega$$

2.2.1 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:

For each $n \in \{0, 1, \ldots, N-1\}$ find $c_h^{n+1} \in V_h$ such that:

$$\int_\Omega \frac{c_h^{n+1} - c_h^n}{\Delta t} v_h d\Omega + d(c_h^{n+1}, v_h) + e(c_h^{n+1}, v_h) = 0 \qquad \forall v_h \in V$$

given that $c_h^0 = c_{h,0}$.
In this case a nonlinear system of equations has to be solved, since $e(c_h^{n+1}, v_h)$ is nonlinear in c_h^{n+1}.
Let us introduce the residual:

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

then the previous formulation translates as:
Find $c_h^{n+1} \in V_h$ such that:

$$R(c_h^{n+1}, v_h) = 0 \qquad \forall v_h \in V_h$$

given that $c_h^0 = c_{h,0}$
Let us introduce the Frechet derivative $a(c)(\delta, v) := \frac{dR}{dc}$ of $R(c, v)$.
Since $a(c)(\delta, v) + \sigma(delta) = R(c_h^{n+1} + \delta, v_h) - R(c_h^{n+1}, v_h)$ we obtain that:

$$a(c_h^{n+1})(\delta, v_h) + o(\delta) = \int \frac{\delta}{\Delta T} v_h d\Omega + \int D \nabla \delta \nabla v_h d\Omega - \int \alpha(1 - 2c_h^{n+1}) \delta v_h d\Omega$$

We now use the Newton method for solving the problem.

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 1: Nonlinear solver (with differential problem)

$n = 0$;
while $n\Delta t < T$ **do**
  $k = 0$;
  while *residual* $\geq$ *tol* ***and*** $n_{iter} < n_{iter,max}$ **do**
    Solve $a(c_h^{n+1,k})(\delta^{n+1,k}, v_h) = R(c_h^{n+1,k}, v_h)$;
    Update $c_h^{n+1,k+1} = c^{n+1,k} + \delta^k$;
    k++;
  end
  n++;
end

By introducing the discrete basis $\{\phi_i\}$, the algorithm can be rewritten as follows:

Algorithm 2: Nonlinear solver (with algebraic problem)

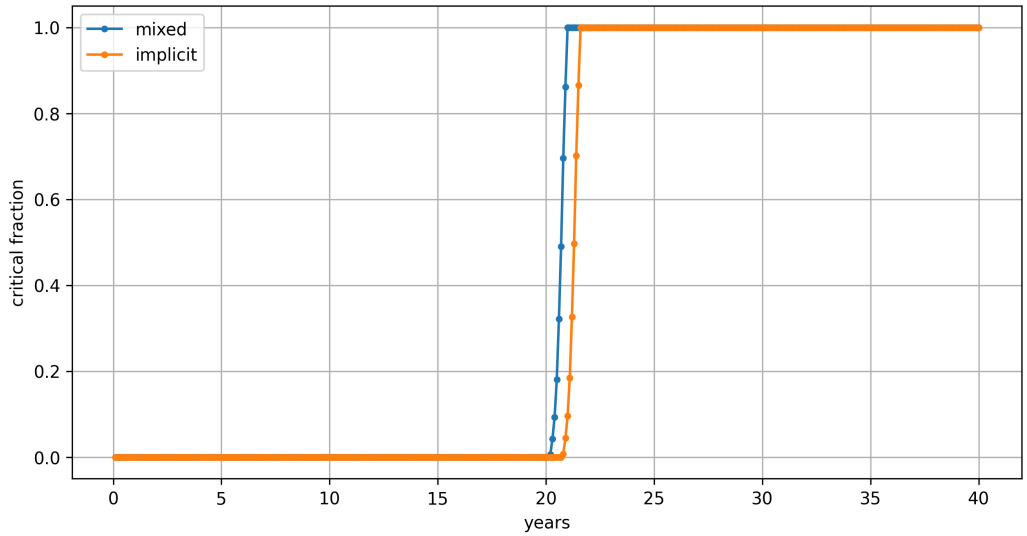
$n = 0$;
while $n\Delta t < T$ **do**
  $k = 0$;
  while *residual* $\geq$ *tol* ***and*** $n_{iter} < n_{iter,max}$ **do**
    Assemble $\mathbf{A}$: $\mathbf{A}_{ij} = a(c_h^{n+1,k})(\phi_j, \phi_i)$;
    Assemble $\mathbf{R}$: $\mathbf{R}_i = -R(c_h^{n+1,k}, \phi_i)$;
    Solve $\mathbf{A}\delta^{n+1,k} = \mathbf{R}$;
    Update $\mathbf{c}^{n+1,k+1} = \mathbf{c}^{n+1,k} + \delta^k$;
    k++;
  end
  n++;
end

2.3.2 Stability and Accuracy

The implicit method is stable for any choice of Δt and has an accuracy quadratic in h, $O(h^2)$, and linear in Δt, $O(\Delta t)$

3 Results and algorithmic comparison

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.



3.1 Results for implicit solver

The solution obtained describes the time evolution of the relative concentration of misfolded proteins. From this solution, it is possible to determine, for each node, the first time at which the critical concentration ($c > 0.95$) is reached, and, at each time, the fraction of nodes above this threshold — referred to respectively as the critical time and the critical fraction.
The Figures 1 - 4 show the critical time for some slices of the 3D mesh.

We used the following parameters:

- $T = 40$ years
- $\Delta t = 0.1$ years
- $r = 1$
- `newton_max_iter` $= 1000$
- `newton_residual_tol` $= 1 \times 10^{-6}$

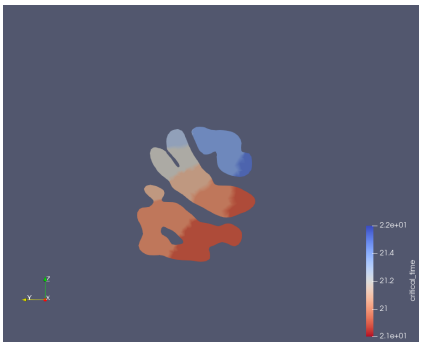


Figure 1: $x = 15$ mm

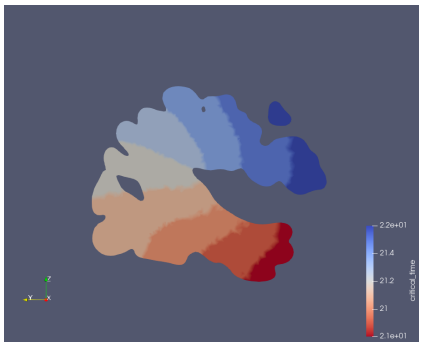


Figure 2: $x = 35$ mm

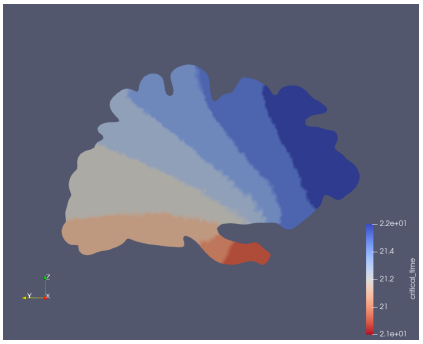


Figure 3: $x = 55$ mm

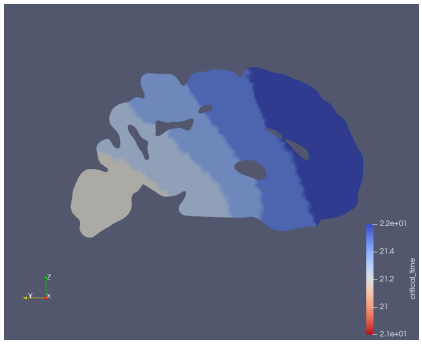
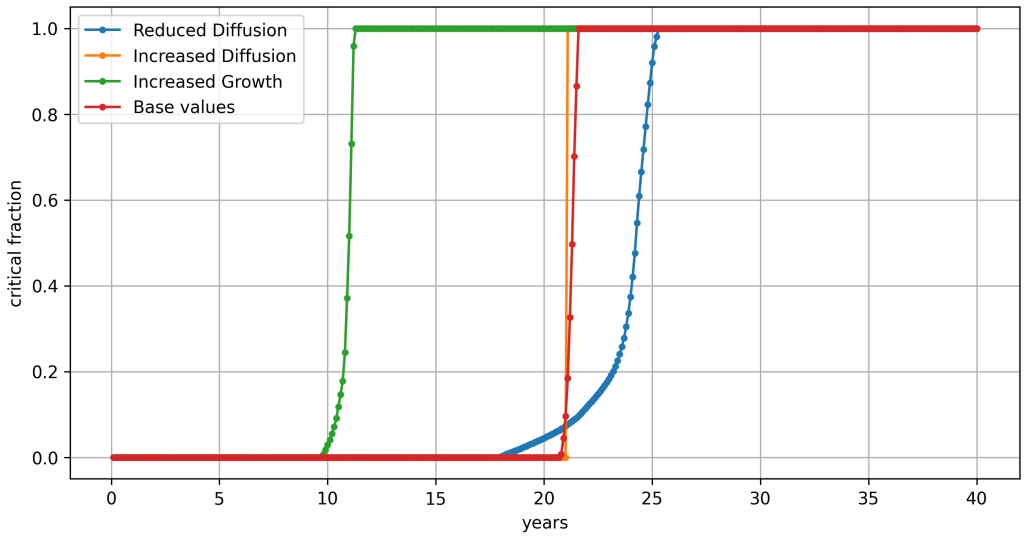


Figure 4: $x = 75$ mm

3.2 Sensitivity Analysis for implicit solver

We tried 4 different configurations of the parameters to see (roughly) how the solution would change. Here are listed the 4 configurations; the results instead are shown in Figures 5 - 8.

- **Baseline**: $\alpha = 0.5$, diff $= 150$ mm^2/year
- **Increased diffusion**: $\alpha = 0.5$, diff $= 600$ mm^2/year
- **Reduced diffusion**: $\alpha = 0.5$, diff $= 37.5$ mm^2/year
- **Increased growth**: $\alpha = 1.0$, diff $= 150$ mm^2/year



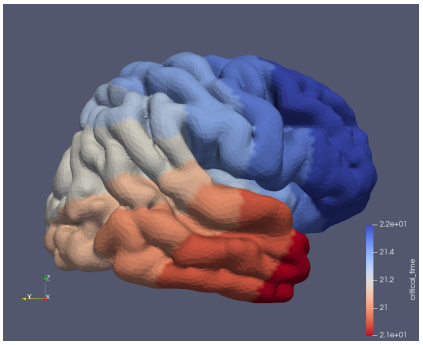


Figure 5: Baseline

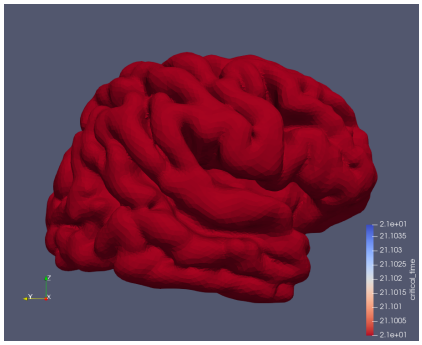


Figure 6: Increased diffusion

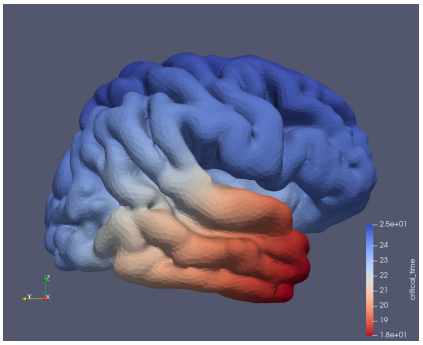


Figure 7: Reduced diffusion

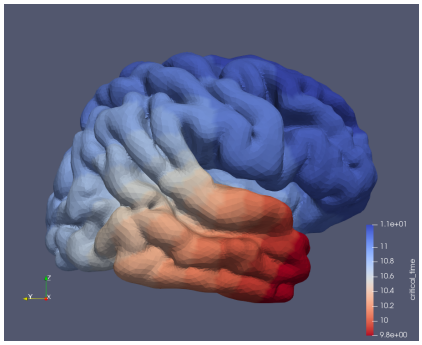
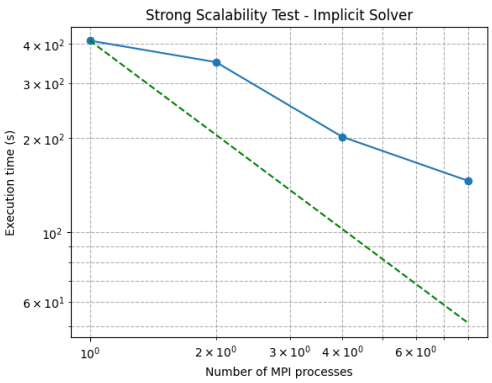


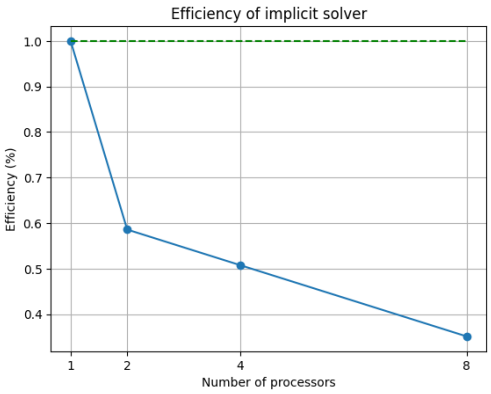
Figure 8: Increased growth

3.3 Strong Scalability Test

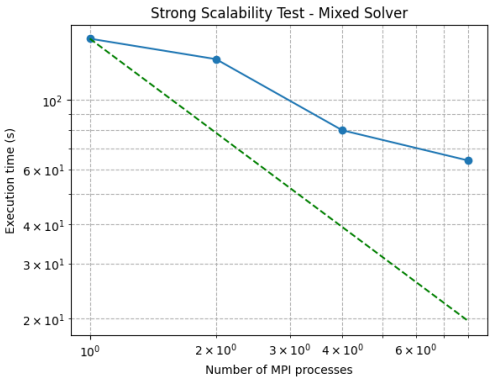
A strong scalability test has been done by varying the number of MPI processes and measuring the total execution time.
The benchmark machine is a laptop running Ubuntu through WSL on Win11 64bit with AMD Ryzen 7 6800H (8 physical cores) and RAM 16GB.

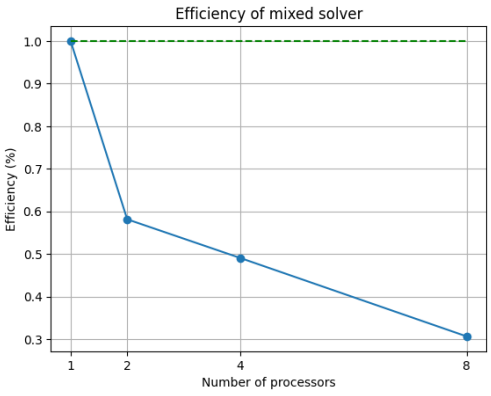
- **Implicit solver** ($T = 5$ years, $\Delta t = 0.5$)





- **Mixed solver** ($T = 20$ years, $\Delta t = 0.5$)





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.