# Project Report: Heterogeneous Catalysis

March 2023

## 1 Introduction

The rate of chemical reactions is often limited by the activation energy the reactants need for the reaction to occur at a fixed temperature and pressure. *Catalysts* are added substances that offer a different reaction path and effectively increase the reaction rate by lowering the activation energy. If a catalyst is in a different phase than the reactants it is called *heterogeneous*. To increase the efficiency of many industrial chemical processes, catalysts are used. The Chlor-Alkali electrolysis of sodium chloride for the production of chlorine, sodium hydroxide, and hydrogen is an example of an important industrial electrochemical process. Chloride ions in the aqueous electrolyte are oxidized at an anode that is chosen to catalyze the reaction. This is a typical example of a heterogeneous catalysis.

In this project a simplified model of a heterogeneous catalysis of a reaction $A \to B$ is considered. The reaction path due to the catalysis includes an intermediate reactant $C$ that only "lives" at a boundary of the domain. The oxidization of chloride ions, for example, includes as intermediate adsorbed chloride ions at the anode. The goal is to model the production rate of $B$ as a function of the process parameters such as the flow velocity and the available catalyst sites.

The second section introduces the continuous mathematical model of the catalysis. Next the discretization approach is explained and the different methods are discussed. In the fourth section details of the implementation are outlined. Experimental results are reported in the fifth section. The final section includes an outlook where improvements to the simplified model are discussed.

## 2 Mathematical Model

A 2-dimensional model in a rectangular domain $\Omega = (0, L) \times (0, H)$ with boundary $\Gamma$ is considered (see Figure 1).

The two species $A$ and $B$ are described by their molar densities $[A] = u_A$ and $[B] = u_B$ within the domain. The reaction path $A \leftrightarrow C$ , $C \leftrightarrow B$ includes the boundary species $C$ that is described at the catalytic boundary $\Gamma_{\text{cat}}$ by the molar surface density $[C] = u_C$.

To derive a continuous model for the evolution of the species $A$, $B$, and $C$ the conservation of mass is applied: Let $\omega \subset \Omega$ be a control volume. The amount of substance $A$ and $B$ in the control volume at some time $t$ is respectively

$$\int_\omega u_A(t,x)dV \qquad \text{and} \qquad \int_\omega u_B(t,x)dV.$$

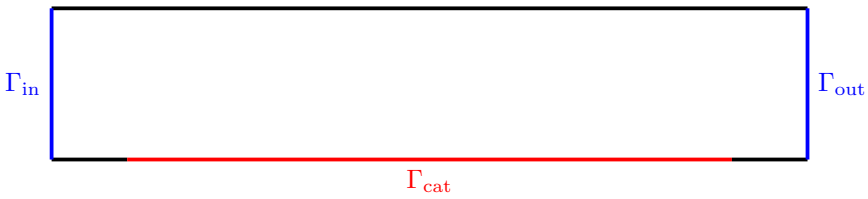



Figure 1: Domain of the 2-dimensional model

By the Reynold's transport theorem, the rate of change in the amount of substance is given by

$$\frac{d}{dt}\int_{\omega} u_A(t,x)\ dV = \int_{\omega} \frac{\partial u_A}{\partial t}(t,x)\ dV + \int_{\partial\omega} u_A(t,x)\vec{v}\cdot\vec{n}\ dS$$

where $\partial\omega$ is the boundary of $\omega$, $\vec{v}$ is the flow velocity, and $\vec{n}$ the outer surface normal vector. Applying the conservation of mass the rate of change must be due to a diffusive flux, $\vec{F}_A$ through the surface of the control volume and possible a source $f$ within the volume:

$$\int_{\omega} \frac{\partial u_A}{\partial t}(t,x)\ dV + \int_{\partial\omega} u_A(t,x)\vec{v}\cdot\vec{n}\ dS = -\int_{\partial\omega} \vec{F}_A(t,x)\cdot\vec{n}\ dS + \int_{\omega} f(t,x)\ dV. \tag{1}$$

A differential version of the continuity equation (1) can be derived by applying the divergence theorem:

$$\partial_t u_A(t,x) + \nabla\cdot\left(\vec{F}_A(t,x) + u_A(t,x)\vec{v}\right) = f(t,x)$$

for every $x\in\Omega$. The quantity $\vec{j}_A = \vec{F}_A(t,x) + u_A(t,x)\vec{v}$ is the total flux consisting of a diffusive and convective part. The diffusive flux within $\Omega$ is assumed to be diffusive following Fick's law

$$\vec{F}_A(t,x) = -D_A\nabla u_A(t,x)$$

where $D_A$ is the diffusion coefficient of the species $A$. The normal flux at the catalytic boundary $\Gamma_{\text{cat}}$ describes the reaction rate per area, $R_{AC}$ of the reaction $A\leftrightarrow C$. The term is modeled using the mass action law:

$$R_{AC}(u_A,u_C) = k_{AC}^{+}u_A(1-u_C) - k_{AC}^{-}u_C$$

where $k_A^{\pm}C$ are reaction rate coefficients. Moreover, the term $1-u_C$ is used to describe the surface concentration of free catalytic sites. Therefore, the normal flux at the boundary $x\in\Gamma_{\text{cat}}$ is given by

$$\vec{j}_A(t,x)\cdot\vec{n} = -S\ R_{AC}(u_A,u_C)$$

where $S$ is the fraction of $\Gamma_{\text{cat}}$ that functions as catalytic sites.

At the boundaries $\Gamma_{\text{in}}$ and $\Gamma_{\text{out}}$ Dirichlet boundary conditions are assumed. At the remaining boundary zero normal flux is assumed.

Similar considerations hold for the species $B$ and $C$. To summarize, the model is given by:

$$\int_\omega \partial_t u_A \, dV + \int_{\partial\omega\cap\Omega} (-D_A \nabla u_A + u_A \vec{v}) \cdot \vec{n} \, dS + \int_{\partial\omega\cap\Gamma_{\text{cat}}} -SR_{AC}(u_A, u_C) \, dS + \int_{\partial\omega\cap(\Gamma_{\text{in}}\cup\Gamma_{\text{out}}\cup\Gamma_{\text{rest}})} \vec{\jmath}_A \cdot \vec{n} \, dS = \int_\omega f_A \, dV \tag{2a}$$

$$\int_\omega \partial_t u_B \, dV + \int_{\partial\omega\cap\Omega} (-D_B \nabla u_B + u_B \vec{v}) \cdot \vec{n} \, dS + \int_{\partial\omega\cap\Gamma_{\text{cat}}} -SR_{BC}(u_B, u_C) \, dS + \int_{\partial\omega\cap(\Gamma_{\text{in}}\cup\Gamma_{\text{out}}\cup\Gamma_{\text{rest}})} \vec{\jmath}_B \cdot \vec{n} \, dS = \int_\omega f_B \, dV \tag{2b}$$

$$\int_{\partial\omega\cap\Gamma_{\text{cat}}} \partial_t u_C \, dS + \int_{\partial\omega\cap\Gamma_{\text{cat}}} S\left(R_{AC}(u_A, u_C) + R_{BC}(u_B, u_C)\right) dS = 0 \tag{2c}$$

together with the boundary conditions

$$\begin{aligned}
u_A(t,x) &= 1 && x \in \Gamma_{\text{in}} \\
u_A(t,x) &= 0 && x \in \Gamma_{\text{out}} \\
u_B(t,x) &= 0 && x \in \Gamma_{\text{in}} \\
u_A(t,x) &= 0 && x \in \Gamma_{\text{out}} \\
\vec{\jmath}_A \cdot \vec{n} &= 0 && x \in \Gamma_{\text{rest}} \\
\vec{\jmath}_B \cdot \vec{n} &= 0 && x \in \Gamma_{\text{rest}}
\end{aligned}$$

# 3 Discretization

In order to solve the model (2) numerically, a discretization scheme has to be chosen.

## 3.1 Spatial Discretization

A finite volume ansatz based on Voronoi cells is selected for the spatial discretization in this project (see Figure 2). I.e., the domain $\Omega$ is subdivided into a finite number of control volumes $\{\omega_k\}_{k\in\mathcal{N}}$ where each control volume is assigned a collocation point $x_k \in \Omega$. Let $\partial\Omega = \bigcup_{m\in\mathcal{G}} \Gamma_m$ where $\Gamma_m$ are straight lines and $\vec{n}|_{\Gamma_m} = \vec{n}_m$ denote the normal vector to $\Gamma_m$. The following admissibility requirements are to be satisfied:

1. $\overline{\Omega} = \bigcup_{k\in\mathcal{N}} \overline{\omega}_k$,
2. $\omega_k$ is a convex open domain for every $k \in \mathcal{N}$,
3. $\omega_k \cap \omega_\ell = \emptyset$ if $k \neq \ell$,
4. $\sigma_{k\ell} = \overline{\omega}_k \cap \overline{\omega}_\ell$ is either empty, a point, or a straight line: if it is a line then $\omega_k$ and $\omega_\ell$ are called neighbors, define $\mathcal{N}_k = \{\ell \in \mathcal{N} : \omega_k \text{ and } \omega_\ell \text{ are neighbors}\}$,
5. $\gamma_{km} := \omega_k \cap \gamma_m$ is empty or a straight line, define $\mathcal{G}_k = \{m \in \mathcal{G} : |\gamma_{km}| > 0\}$,
6. for every $\ell \in \mathcal{N}_k$, $x_k x_\ell$ is orthogonal to $\sigma_{k\ell}$,

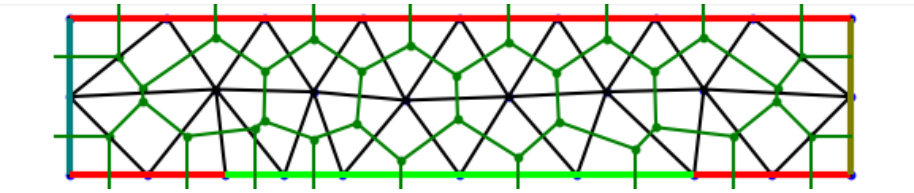


Figure 2: Spatial discretization based on a boundary conforming Delaunay triangulation (black). The control volumes are the Voronoi cells (green) with collocation nodes (black) being the triangulation nodes.

7. $|\partial\omega_k \cap \partial\Omega| > 0$ implies $x_k \in \partial\Omega$ for every $k \in \mathcal{N}$.

The Voronoi finite volume discretization uses Voronoi cells as control volumes. Given a set of fixed points $S$ in $\mathbf{R}^d$ the corresponding Voronoi diagram subdivides $\mathbf{R}^d$ into regions (called Voronoi cells) where two elements of $\mathbf{R}^d$ are in the same region if the point of $S$ that is closest coincides. The construction of the Voronoi cells uses the dual object to the Voronoi diagram the Delaunay triangulation. Connecting two of the points in $S$ by an edge if their respective Voronoi cells are adjacent yields a Delaunay triangulation. A triangulation of the convex hull of $S$ has the Delaunay property if each triangle of the triangulation does not contain any points of $S$. The duality is illustrated in Figure 2 and summarized in Table 1.

| **Voronoi digram** | **Delaunay triangulation** |
|---|---|
| collocation point | vertex of the triangulation |
| vertex of a Voronoi cell | circumcenter of a triangle |
| edges of neighborhood graph | triangulation edges |
| boundary intersects infinite cells only | all circumcenters are within domain |
| collocation points of infinite cells lie on boundary & restriction to boundary is again a Voronoi diagram | constrained to include the boundary |

Table 1: Duality of Voronoi diagrams and Delaunay triangulations

A Delaunay triangulation that is constrained to include the boundary $\partial\Omega$ and satisifes that all circumcenters of the triangles lie within the domain is called a *boundary conforming Delaunay triangulations*. The set of the corresponding Voronoi cells intersected with the domain $\Omega$ together with the collocation points satisfy the admissibility assumptions on the control volumes. Good algorithms for the construction of boundary conforming Delaunay triangulations exist (see e.g. [10] in 2 dimensions).

The densities $u_A(t,x)$, $u_B(t,x)$ and $u_C(t,x)$ are replaced by their values at the collocation points $\mathbf{u}_A = [u_A(t,x_k)]_{k\in\mathcal{N}}$, etc. To spatially discretize the system (2) the integrals are replaced by the following scheme:

- volume integrals of scalar fields: e.g.

$$\int_{\omega_k} \partial_t u \; dV \approx |\omega_k| \; \partial_t u(t,x_k)$$

- surface integral of vector fields: e.g.

$$\int_{\partial\omega_k\cap\Omega} (-D\nabla u + u\vec{v})\cdot\vec{n}\; dS \approx \sum_{\ell\in\mathcal{N}_k} \frac{|\sigma_{k\ell}|}{h_{k\ell}}\; g(u(t,x_k),u(t,x_\ell))$$

  where $h_{k\ell} = |x_k - x_\ell|$ and $g : \mathbf{R}\times\mathbf{R}\to\mathbf{R}$ is called *flux function*

- surface integral of scalar fields: e.g.

$$\int_{\partial\omega_k\cap\Gamma_{\text{cat}}} \partial_t u\; dS \approx |\gamma_{km}|\; \partial_t u(t,x_k)$$

The Dirichlet boundary conditions are included using the penalty method, e.g. $u_A(t,x) = 1$ for $x\in\Gamma_{\text{in}}$ is replaced with

$$-\vec{\jmath}_A(t,x_k)\cdot\vec{n} + \frac{1}{\epsilon}u_A(t,x_k) = \frac{1}{\epsilon}\cdot 1$$

for all $x_k\in\Gamma_{\text{in}}$ where $\epsilon\ll 1$. Therefore,

$$\int_{\partial\omega_k\cap(\Gamma_{\text{in}}\cup\Gamma_{\text{out}}\cup\Gamma_{\text{rest}})} \vec{\jmath}_A\cdot\vec{n}\; dS \approx \sum_{m\in\Gamma_{\text{in}}} |\gamma_{km}|\frac{1}{\epsilon}\left(u_A(t,x_k)-1\right) + \sum_{m\in\Gamma_{\text{out}}} |\gamma_{km}|\frac{1}{\epsilon}u_A(t,x_k)$$

for every $k\in\mathcal{N}$.

In summary the spatially discretized model is given by the following system

$$\begin{aligned}|\omega_k|\partial_t u_{A,k}(t) + \sum_{\ell\in\mathcal{N}_k}\frac{|\sigma_{k\ell}|}{h_{k\ell}}\; g(u_{A,k}(t),u_{A,\ell}(t)) - |\gamma_{km}|SR_{AC}(u_{A,k}(t),u_{C,k}(t))\\ + \sum_{m\in\Gamma_{\text{in}}}|\gamma_{km}|\frac{1}{\epsilon}\left(u_{A,k}(t)-1\right) + \sum_{m\in\Gamma_{\text{out}}}|\gamma_{km}|\frac{1}{\epsilon}u_{A,k}(t) = |\omega_k|f_{A,k}(t)\end{aligned}\tag{3a}$$

$$\begin{aligned}|\omega_k|\partial_t u_{B,k}(t) + \sum_{\ell\in\mathcal{N}_k}\frac{|\sigma_{k\ell}|}{h_{k\ell}}\; g(u_{B,k}(t),u_{B,\ell}(t)) - |\gamma_{km}|SR_{BC}(u_{B,k}(t),u_{C,k}(t))\\ + \sum_{m\in\Gamma_{\text{in}}}|\gamma_{km}|\frac{1}{\epsilon}\left(u_{B,k}(t)-1\right) + \sum_{m\in\Gamma_{\text{out}}}|\gamma_{km}|\frac{1}{\epsilon}u_{B,k}(t) = |\omega_k|f_{B,k}(t)\end{aligned}\tag{3b}$$

$$|\gamma_{km}|\;\partial_t u_{C,k}(t) + |\gamma_{km}|S\left(R_{AC}(u_{A,k}(t),u_{C,k}(t)) + R_{BC}(u_{B,k}(t),u_{C,k}(t))\right) = |\gamma_{km}|f_{C,k}(t)\tag{3c}$$

for $k\in\mathcal{N}$.

To simplify the notation the system (3) will sometimes be rewritten as

$$\begin{aligned}\partial_t\mathbf{u}_A(t) + \mathbf{X}(\mathbf{u}_A(t),\mathbf{u}_B(t),\mathbf{u}_C(t)) &= \mathbf{f}_A(t)\\ \partial_t\mathbf{u}_B(t) + \mathbf{Y}(\mathbf{u}_A(t),\mathbf{u}_B(t),\mathbf{u}_C(t)) &= \mathbf{f}_B(t)\\ \partial_t\mathbf{u}_C(t) + \mathbf{Z}(\mathbf{u}_A(t),\mathbf{u}_B(t),\mathbf{u}_C(t)) &= \mathbf{f}_C(t)\end{aligned}$$

### 3.2 Discretization in Time

Finally, the system (3) has to be discretized in time. The most common approach is to use the implicit Euler scheme. An initial value at the starting time $t_0$ must be provided. The values at consecutive times are defined recursively. Let $h_1, h_2, \ldots, h_N > 0$ be time steps and let $t_i = t_{i-1} + h_i$ for $i = 1, 2, \ldots, N$. The system (4) is replaced by the fully discretized, recursive system

$$\mathbf{u}_A^i - \mathbf{u}_A^{i-1} + h_i \mathbf{X}(\mathbf{u}_A^i, \mathbf{u}_B^i, \mathbf{u}_C^i) = h_i \mathbf{f}_A^i \qquad i = 1, 2, \ldots, N \tag{5a}$$

$$\mathbf{u}_B^i - \mathbf{u}_B^{i-1} + h_i \mathbf{Y}(\mathbf{u}_A^i, \mathbf{u}_B^i, \mathbf{u}_C^i) = h_i \mathbf{f}_B^i \qquad i = 1, 2, \ldots, N \tag{5b}$$

$$\mathbf{u}_C^i - \mathbf{u}_C^{i-1} + h_i \mathbf{Z}(\mathbf{u}_A^i, \mathbf{u}_B^i, \mathbf{u}_C^i) = h_i \mathbf{f}_C^i \qquad i = 1, 2, \ldots, N \tag{5c}$$

$$\mathbf{u}_A^0, \mathbf{u}_B^0, \mathbf{u}_C^0 \quad \text{given} \tag{5d}$$

where $\mathbf{f}_A^i = \mathbf{f}_A(t_i)$, etc.

An algorithm to determine the step sizes $h_1, h_2, \ldots, h_N$ has to be chosen. The simplest option is to use a fixed step size. Alternatively, adaptive strategies are often preferable:

- Constrained change of the solution:
  1. given $\mathbf{u}^{i-1}$, start with $h_i = h_0$,
  2. compute $\mathbf{u}^i(h_i)$,
  3. if $\|\mathbf{u}^i(h_i) - \mathbf{u}^{i-1}\| > \delta_{\text{abs}}$ reduce $h_i = \frac{1}{2} h_i$ and go back to (2), else done.
- Error estimation by step doubling (Richardson extrapolation): Assuming an asymptotic expansion of the global error in the step size, the error in the $i$th step can be estimated by comparing the solution $\mathbf{u}_h^i$ obtained from doing two steps at a given step size $h$ to the solution $\mathbf{u}_{2h}^i$ obtained from doing one big step with step size $2h$. From the asymptotic expansion with the determined error estimate, the optimal step size can be estimated and used in the next step (see [6]).

### 3.3 Solving Nonlinear Systems of Equations

The system (5) is for every $i \in \{1, 2, \ldots, N\}$ a nonlinear system of equations in $\mathbf{u} = [\mathbf{u}_A^i, \mathbf{u}_B^i, \mathbf{u}_C^i]$. To further simplify the notation introduce functions $\mathbf{A^i}$ such that $\mathbf{A}^i(\mathbf{u^i}) = \mathbf{f}^i$ where $\mathbf{f}^i = [\mathbf{f}_A^i, \mathbf{f}_B^i, \mathbf{f}_C^i]$. The Jacobian of $\mathbf{A}^i$ with respect to the unknowns has the form $\mathbf{1} + h_i \mathbf{D}$. Thus, for $h_i$ small enough the system has a solution in the neighborhood of $\mathbf{u}_A^{i-1}$, $\mathbf{u}_B^{i-1}$, $\mathbf{u}_C^{i-1}$ by the inverse function theorem.

In the project a Newton iteration with damping is used to solve the nonlinear system. For $j = 1, 2, \ldots$

$$\mathbf{r}_{j-1} = \mathbf{u}_{j-1} - \mathbf{f} \tag{6a}$$

$$\mathbf{A}'(\mathbf{u}_j)\mathbf{e}_{j-1} = \mathbf{A}\mathbf{r}_{j-1} \tag{6b}$$

$$\mathbf{u}_j = d_j \mathbf{e}_{j-1} \tag{6c}$$

where $d_1, d_2, \ldots, \in (0, 1]$ are the *damping factors* and the superscripts have been dropped for ease of notation. There exist many options for a stopping criterion. Two common approaches are:

- $\Delta u_j = \|\mathbf{u}_j - \mathbf{u}_{j-1}\|_\infty < \epsilon_{\text{abs}}$ or
- $\Delta u_j / \Delta u_1 < \epsilon_{\text{rel}}$.

The linear system (6b) is sparse. There are two classes of solvers which are used to solve such systems:

- *direct sparse solvers*: calculation of a LU-decomposition with ordering strategies that reduce *fill-in* (see e.g. [5], [2], [3])
- *iterative solvers*: combination of preconditioning and iteration scheme often Krylov subspace methods (see e.g. [8]).

A common algorithm for the control of the damping factors is simply by the easy recursive formula

$$d_i = \min(\alpha d_{i-1}, 1), \qquad d_0 \in (0,1) \qquad\qquad i = 1, 2, \dots$$

for some $\alpha > 1$.

### 3.4 Flux Functions

Recall that the discretization of the surface integral of the (total) flux introduced a flux function $g$. In order to analyze different formulas the flux $-D\nabla u + u\vec{v}$ is split into its diffusive part $-D\nabla u$ and the convective part $u\vec{v}$. Let $\omega_k$ be a control volume and $\ell \in \mathcal{N}_k$. Recall that the line segment between the collocation points $x_k x_\ell$ is orthogonal to the boundary $\sigma_{k\ell}$ of the two cells. Therefore, the following approximation of the diffusive flux is usually used

$$\int_{\sigma_{k\ell}} -D\nabla u \cdot \vec{n} \, dS \approx \frac{|\sigma_{k\ell}|}{h_{k\ell}} D\left(u(t, x_k) - u(t, x_\ell)\right).$$

For the convective flux different options exist which commonly approximate the integral over one boundary segment simply using one quadrature point at the center. The choice is in the approximation of the normal flux at that point. Since the control volumes are Voronoi cells and the collocation points are the defining points of the Voronoi diagram, note that the center of $\sigma_{k\ell}$ is at $(x_k + x_\ell)/2$. Thus, one obvious choice of approximation is

$$\int_{\sigma_{k\ell}} u\vec{v} \cdot \vec{n} \, dS \approx |\sigma_{k\ell}| \frac{u(t, x_k) + u(t, x_\ell)}{2} \, v_{k\ell}$$

where $v_{k\ell} = \vec{v}\,(t, (x_k + x_\ell)/2) \cdot \vec{n}$ is the normal velocity. This approximation is called *central difference* scheme. It has some disadvantages in advection-dominated problems. Next to its stability properties the approximation scheme introduces positive non-diagonal terms in the system matrix that prevents the application of the M-property for the analysis of discrete maximum principles [?]. The corresponding *central difference flux* function is

$$g_c(u_k, u_\ell) = D(u_k - u_\ell) + h_{k\ell} v_{k\ell} \frac{u(t, x_k) + u(t, x_\ell)}{2}.$$

Another choice of approximation is the *upwind* scheme:

$$\int_{\sigma_{k\ell}} u\vec{v} \cdot \vec{n} \, dS \approx |\sigma_{k\ell}| v_{k\ell} \begin{cases} u(t, x_k) & \text{if } \; v_{k\ell} > 0 \\ u(t, x_\ell) & \text{if } \; v_{k\ell} < 0 \end{cases}.$$

It remedies the disadvantages of the central difference scheme but at the cost of introducing numerical diffusion. The corresponding *upwind flux* function is

$$\begin{aligned} g_u(u_k, u_\ell) &= D(u_k - u_\ell) + h_{k\ell} v_{k\ell} \begin{cases} u(t, x_k) & \text{if } \; v_{k\ell} > 0 \\ u(t, x_\ell) & \text{if } \; v_{k\ell} < 0 \end{cases} \\ &= (D + D_{\text{num}}) (u_k - u_l) + h_{k\ell} v_{k\ell} \frac{u(t, x_k) + u(t, x_\ell)}{2} \end{aligned}$$

where $D_{\text{num}} = h_{k\ell} v_{k\ell}/2$ is the introduced numerical diffusion coefficient.

Finally, an approximation that does not treat the diffusive and convective flux separately is the *exponential fitting* scheme [9]:

$$g(u_k, u_\ell) = D \left( B\left( \frac{-v_{k\ell} h_{k\ell}}{D} \right) u_k - B\left( \frac{v_{k\ell} h_{k\ell}}{D} \right) u_\ell \right).$$

It has the same advantages as the upwind discretization scheme but introduces less numerical diffusion.

# 4 Implementation

The outlined algorithms have been implemented using the Julia programming language version 1.8 [1]. The code is available in a Pluto notebook created using the Pluto package [12].

The boundary conforming Delaunay triangulation with the corresponding Voronoi diagram are created using the software `Triangle` [11] through the Julia wrapper package `Triangulate`.

The discretization of the model is done using the package `VoronoiFVM` [4]. The package implements an implicit Euler solver with adaptive step size control applying the damped Newton's method. Next some details on the package are outlined.

## 4.1 `VoronoiFVM`

The physical models for the evolution of a vector of densities $\mathbf{u}$ in a domain $\Omega$ and surface densities $\mathbf{u}_b$ on $\partial\Omega$ in integral form are assumed to be of the form

$$\int_\omega \partial_t \mathbf{s}(\mathbf{u}) \; dV + \int_{\partial\omega} \mathbf{j} \cdot \mathbf{n} \; dS + \int_\omega \mathbf{r}(\mathbf{u}) \; dV = \int_\omega \mathbf{f} \; dV \qquad \text{for } \; \omega \subset \Omega \tag{7a}$$

$$\int_\sigma \partial_t \mathbf{s}_b(\mathbf{u_b}) \; dV_b + \int_{\partial\sigma} \mathbf{j}_b \cdot \mathbf{n}_b \; dS_b + \int_\sigma \mathbf{r}_b(\mathbf{u}, \mathbf{u}_b) \; dV_b = \int_\sigma \mathbf{f}_b \; dV_b \qquad \text{for } \; \sigma \subset \partial\Omega \tag{7b}$$

$$-\mathbf{j} \cdot \mathbf{n} + \boldsymbol{\alpha} \odot \hat{\mathbf{r}}_b(\mathbf{u}, \mathbf{u}_b) = \boldsymbol{\beta} \qquad \text{on } \; \partial\Omega \tag{7c}$$

where $\mathbf{s}$ is called the *storage* term, $\mathbf{j}$ contains the fluxes of all species, $\mathbf{r}$ is a *reaction* term, and the components of $\mathbf{f}$ are *source* terms. The transport of the surface is similar. At the boundary, there are different possible types of conditions on the densities $\mathbf{u}$ such as Dirichlet, Neumann or Robin conditions. Since Dirichlet boundary conditions are reduced to Robin conditions with the help of the penalty method, the most general form is described by equation (7c). Here, $\odot$ symbolizes component-wise multiplication and $\hat{\mathbf{r}}_b$ possibly describes the reaction term of a surface reaction.

Note that the model (2) is of the described form where $\partial\omega$ has been split into the components $\partial\omega \cap \Omega$ and $\partial\omega \cap \Gamma$. Further the boundary condition on $\Gamma_{\text{cat}}$ has already been incorporated into the continuity equation. The explicit formulas for the terms are summarized in the appendix.

A `VoronoiFVM.System` is the main data structure that combines the geometric data with the physics and contains additional information used in the time integration. The package provides a `solve` method that for transient problems implements an implicit Euler scheme for the time integration. A built-in `SolverControl` data structure is used to control several aspects of the time integration:

- *step size control*: An adaptive strategy based on a constrained change of the solution (see 3.2) is used. The initial time step $h_0$ is given as the parameter `Δt`. The parameter $\delta_{\text{abs}}$ can be provided as `Δu_opt` and the function `delta` of the `oldsolution` $\mathbf{u}^{i-1}$ and the new `solution` $\mathbf{u}^i(h_i)$ computes $\|\mathbf{u}^i(h_i) - \mathbf{u}^{i-1}\|$. By default, `Δt` $= 0.1$, `Δu_opt` $= 0.1$ and `delta` uses the infinity norm. how exactly does $\Delta\lambda$ change? what about `Δt_min` and `Δt_max`?
- *stopping criterion*: Both the absolute and relative stopping criteria are used. The corresponding parameters and their default values are `abstol` $= 10^{-10}$ and `reltol` $= 10^{-10}$ respectively.
- *damping factor control*: The strategy presented in section 3.3 is used. The parameters and their default values are `damp_initial` $= 1.0$ and `damp_growth` $= 1.2$.
- *linear solver control*: The method `method_linear` must be of type `LinearSolve.SciMLLinearSolveAlgorithm`. Many built-in solvers can be found in the package `LinearSolve`. By default `SparspakFactorization` is used for models for 2-dimensional domains.

See the documentation of the `VoronoiFVM` for additional control parameters. Furthermore, `VoronoiFVMDiffEq` provides an interface to use other time integrators from the package `DifferentialEquations` [7].

The non-homogeneous Dirichlet boundary condition at $\Gamma_{\text{in}}$ is enforced gradually to avoid sharp jumps in the initial value.

### 4.2 Choice of Parameters

Is this section necessary?

## 5 Experiments

The goal of the numerical experiments is to understand the dependence of the production rate of the species $B$ in the steady state on the fraction of available catalyst sites $S$ and the flow velocity $\vec{v}$. Two different velocity fields are considered:

- uniform horizontal velocity field: $\vec{v}(x, y) = [v_x \quad 0]^T$ and
- Hagen-Poiseuille flow[1]: $\vec{v}(x, y) = [6v_x(1 - y/H)y/H \quad 0]^T$.

[1]The Hagen-Poiseuille flow describes the laminar flow of an incompressible Newtonian fluid through a long cylindrical pipe of constant cross section.

We assume no source terms for all species, i.e. $f_A = f_B = f_C = 0$.

From the model (2) and its discretization (3) which preserves the conservation property, the production rate, $p_B$, of the species $B$ in the steady state can be measured in two different ways:

$$p_B = -S \int_{\Gamma_{\text{cat}}} R_{BC}(u_B, u_C)\, dS$$

which directly measures the production of $B$ through $C$ or it can be measured indirectly:

$$p_B = \int_{\Gamma_{\text{in}} \cup \Gamma_{\text{out}}} \vec{j}_B \cdot \vec{n}\, dS$$

By the enforced conservation property the terms should coincide also numerically.

The diffusion coefficients are chosen to be $D_A = 1.0$ and $D_B = 0.001$. The reaction constant coefficients are $k^+_{AC} = 100.0$, $k^-_{AC} = 1.0$, $k^+_{BC} = 0.1$ and $k^-_{BC} = 1.0$.

The results with $S$ and $v_x$ as parameters are given in Table 2 and illustrated in Figure 3. The steady state solution in the uniform velocity field is plotted in Figure 4

| $v_x$ | $S$ | | |
|---|---|---|---|
| | 0.1 | 0.3 | 0.6 |
| 0.0 | 2.47 | 1.01 | 0.55 |
| 0.2 | 3.60 | 1.47 | 0.79 |
| 0.4 | 4.73 | 1.88 | 1.00 |
| 0.6 | 5.49 | 2.35 | 1.24 |
| 0.8 | 5.69 | 2.87 | 1.49 |
| 1.0 | 5.76 | 3.43 | 1.77 |

Table 2: Production rate $p_B$ in dependence of $S$ and $v_x$

# 6 Discussion and Outlook

One can observe that the production rate of $B$ increases monotonically with the horizontal flow velocity $v_x$. This is to be expected because the flow transports the species $B$ away through $\Gamma_{\text{out}}$ which decreases $u_B$ at $\Gamma_{\text{cat}}$. In this way the reaction coefficient $R_{BC}$ becomes more negative.

The second observation is that an increase of $S$ yields a decrease in the production rate of $B$. It was observed that the for smaller $S$ the density $u_C$ is closer to 1. Therefore, the reaction coefficient $R_{BC}$ is more negative. An additional figure or table might be helpful. Waiting for response if $S$ should be included in the continuity equation of $u_C$

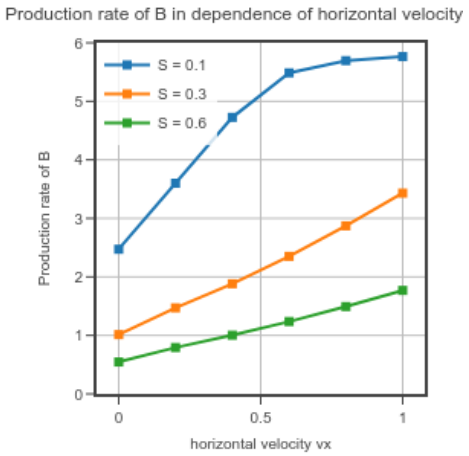
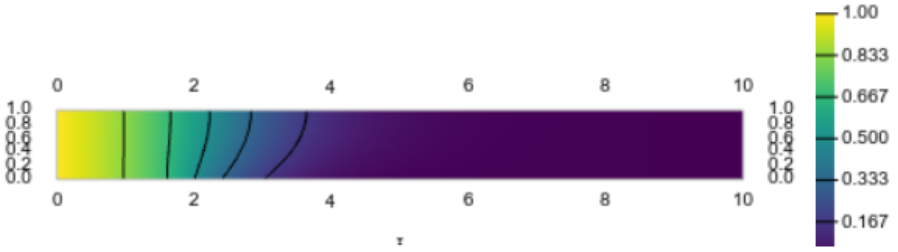
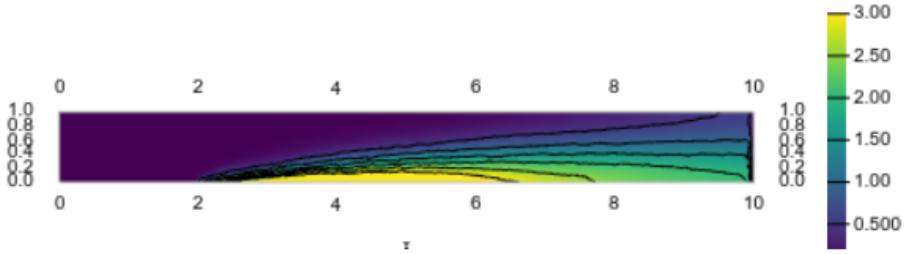


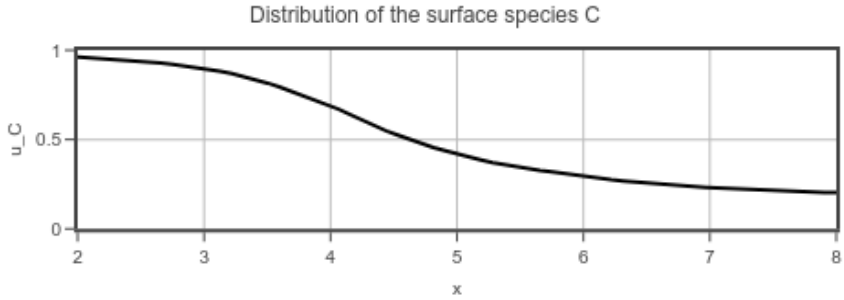

Figure 3: Production rate $p_B$ in dependence of $S$ and $v_x$




(a) Distribution of the density $u_A$ in $\Omega$




(b) Distribution of the density $u_B$ in $\Omega$




(c) Distribution of the surface density $u_C$ on $\Gamma_{\text{cat}}$

Figure 4: Steady state solution in an uniform velocity flow with $v_x = 0.5$ and $S = 0.3$?

# References


[1] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[2] Tim A. Davis. Algorithm 832: Umfpack – an unsymmetric-pattern multifrontal method with a column pre-ordering strategy. *ACM Trans. on Mathematical Software*, 30(2):196–199, 2004.

[3] Tim A. Davis and Ekanathan P. Natarajan. Algorithm 907: Klu, a direct sparse solver for circuit simulation problems. *ACM Trans. on Mathematical Software*, 37(3):1–17, 2010.

[4] J. Fuhrmann and contributors. VoronoiFVM.jl: Finite volume solver for coupled nonlinear partial differential equations. https://github.com/j-fu/VoronoiFVM.jl, 2019-2021.

[5] Alan George and Joseph W. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice Hall, 1 edition, 1981.

[6] Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer Series in Computational Mathematics. Springer, 2 edition, 2000.

[7] Christopher Rackauckas and Qing Nie. Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1):15, 2017.

[8] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Other Titles in Applied Mathematics. SIAM, 2 edition, 2003.

[9] D.L. Scharfetter and H.K. Gummel. Large-signal analysis of a silicon read diode oscillator. *IEEE Transactions on Electron Devices*, 16(1):64–77, 1969.

[10] Jonathan R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1–3):21–74, May 2002.

[11] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[12] Fons van der Plas and contributors. Pluto.jl. https://github.com/fonsp/Pluto.jl, 2023.

# Appendix

$$
\begin{aligned}
\mathbf{u} &= [u_A,\ u_B]^T \\
\mathbf{u}_b &= 1_{\Gamma_{\text{cat}}}\ u_C \\
\mathbf{s}(\mathbf{u}) &= \mathbf{u} \\
\mathbf{j} &= [(-D_A \nabla u_A + u_A \mathbf{v}),\quad (-D_B \nabla u_B + u_B \mathbf{v})]^T \\
\mathbf{r}(\mathbf{u}) &= 0 \\
\mathbf{f} &= [f_A, f_B]^T \\
\mathbf{s}_b(\mathbf{u}_b) &= \mathbf{u}_b \\
\mathbf{j}_b &= 0 \\
\hat{\mathbf{r}}_b(\mathbf{u}, \mathbf{u}_b) &= 1_{\Gamma_{\text{cat}}}\ S(R_{AC}(u_A, u_C) + R_{BC}(u_B, u_C)) \\
\mathbf{f}_b &= 0 \\
\boldsymbol{\alpha} &= \left[(1_{\Gamma_{\text{in}}} + 1_{\Gamma_{\text{out}}})\ \frac{1}{\epsilon} - 1_{\Gamma_{\text{cat}}}\ S\ R_{AC}(u_A, u_C),\ (1_{\Gamma_{\text{in}}} + 1_{\Gamma_{\text{out}}})\ \frac{1}{\epsilon} - 1_{\Gamma_{\text{cat}}}\ S\ R_{BC}(u_B, u_C)\right] \\
\boldsymbol{\beta} &= \left[1_{\Gamma_{\text{in}}}\ \frac{1}{\epsilon},\quad 0\right]
\end{aligned}
$$