Computers and Chemical Engineering 206 (2026) 109488

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/cace

A cloud-enabled digital twin platform for upstream processing in biotechnology: Integrating CFD, compartmentalization, and reaction kinetics

Xiyan Li [a,*], Sebastian L. Jensen [a], Noah B. Christiansen [a], Elham Ramin [a], Johan le Nepvou de Carfort [a], Johannes Schmölder [b], Eric von Lieres [b,c], Krist V. Gernaey [a]

[a] Department of Chemical and Biochemical Engineering, PROSYS, Technical University of Denmark, Søltofts Plads, Building 228 A, Lyngby, 2800, Denmark
[b] Forschungszentrum Jülich, Institute of Bio- and Geosciences 1: Biotechnology, Wilhelm-Johnen-Straße 1, Jülich, 52428, Germany
[c] RWTH Aachen University, Computational Systems Biotechnology, Templergraben 55, Aachen, 52062, Germany

ARTICLE INFO

Keywords:
Bioreactor
CFD
Compartment model
Reaction kinetics
Digital twin
API
Inno4Vac

ABSTRACT

Biotechnology process modeling has become an essential tool in both research and industrial settings, offering a cost-effective and efficient alternative to extensive physical experimentation. It enables the simulation and analysis of complex biological systems, supporting faster development cycles and better decision-making. To foster collaboration and knowledge sharing in this domain, we present an open source cloud-enabled platform for upstream biotechnology process modeling. The platform provides an integrated environment where users can combine computational fluid dynamics (CFD), compartmental models, and kinetic simulations within a unified and modular interface. Each component of the model operates independently, but can be seamlessly coupled through a standardized API. The system is designed to support both research and educational use cases, with an emphasis on accessibility, extensibility, and reproducibility. This paper outlines the platform architecture and implementation, highlights the technical challenges addressed in model integration, and discusses opportunities for future development. By lowering technical barriers and encouraging community-driven innovation, the platform aims to advance digital twin applications in biotechnology upstream processing.

1. Introduction

Biopharmaceutical process development involves cultivating living cells using sensitive biological processes. Numerous variables can impact these processes, making it crucial to predict outcomes early and efficiently. Model-based approaches, conducted within a virtual bioreactor, offer an alternative to traditional experimental methods (Ramin et al., 2024). This article explores the potential integration of a model-based approach for upstream simulations and full-stack development to expedite the development of upstream processes for cell-based biopharmaceutical products.

Model-based approaches have significantly advanced the understanding of complex fermentation processes. For instance, establishing predictive metabolic models and coupling them with Computational Fluid Dynamics (CFD) has proven invaluable (Ramin et al., 2024). These models enable researchers to explore various scenarios and optimize processes with a reduced reliance on physical experimentation. Biotechnology process models are not only cost-effective, as they eliminate the need for extensive laboratory experiments, but also fast and repeatable, providing valuable insights into system behavior. In addition, they facilitate informed decision-making. When utilized effectively, these models can assist major pharmaceutical companies in addressing challenges related to scaling up and scaling down.

CFD models offer substantial benefits in biomanufacturing by providing detailed insights to fluid flow, mixing, and mass transfer within bioreactors (Tajsoleiman et al., 2019). These models enable precise simulation of various conditions, allowing researchers to optimize parameters such as nutrient distribution, temperature control, and agitation rates. By accurately predicting the behavior of biological systems, CFD models help improve process efficiency, reduce costs, and minimize the need for extensive physical experimentation (Le Nepvou De Carfort et al., 2024). In addition, they facilitate the scaling up of production processes from laboratory to industrial scale, ensuring consistency and reliability (Wang et al., 2020). Overall, the use of CFD models in biomanufacturing leads to more informed decision-making, improved product quality, and accelerated development timelines.

* Corresponding author.
E-mail addresses: xiylia@dtu.dk, xiylia@kt.dtu.dk (X. Li).

https://doi.org/10.1016/j.compchemeng.2025.109488
Received 23 June 2025; Received in revised form 14 November 2025; Accepted 16 November 2025
Available online 4 December 2025
0098-1354/© 2025 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

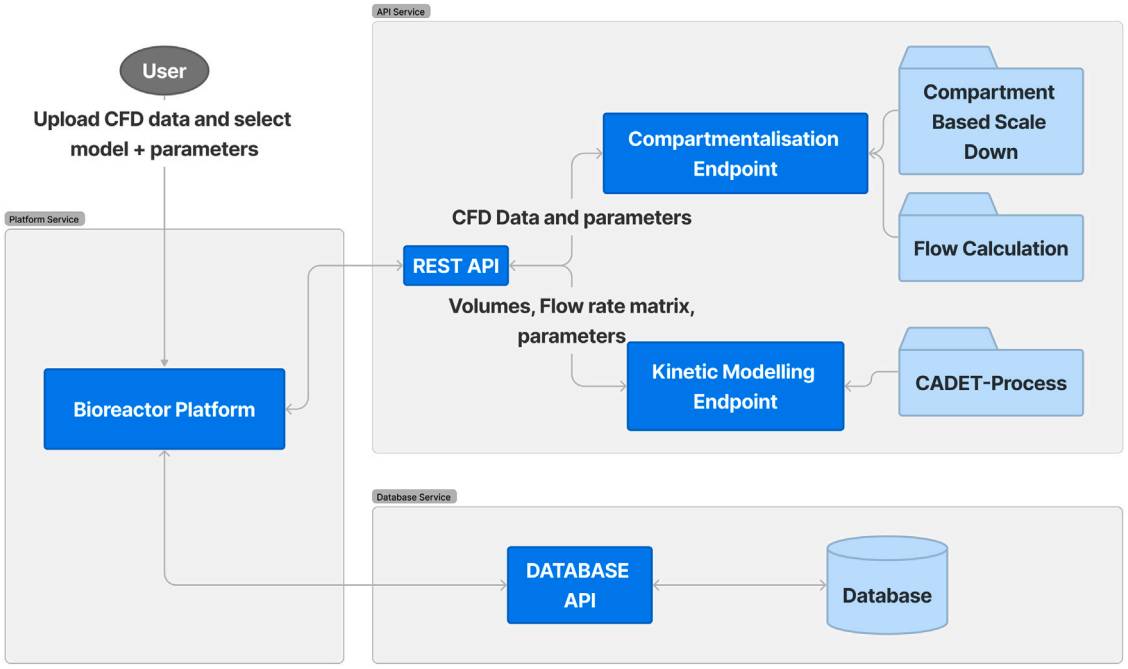


Fig. 1. Platform architecture design.

The compartment models provide a simplified alternative to CFD models by simplifying the grid complexity to approximate the fluid field within the reactor (Tajsoleiman et al., 2019; Le Nepvou De Carfort et al., 2024; Nadal-Rey et al., 2021; Liu and Wilhite, 2023; Bisgaard et al., 2022). This approach offers a cost-effective solution for upstream simulations in biomanufacturing. It has been a key analytical tool at the PROSYS research center for many years (Tajsoleiman et al., 2019; Le Nepvou De Carfort et al., 2024; Nadal-Rey et al., 2021; Bisgaard et al., 2022; Tajsoleiman, 2018) and is undergoing a significant transformation. Originally developed in a different programming language, our team has now translated and enhanced the models into Python, creating a robust and scalable package. This newly developed package is set to be launched on our online platform, providing researchers and analysts with a state-of-the-art tool for their studies. To expand the functionality of the platform, we have connected it with CADET (Leweke et al., 2025), where we can integrate compartment modeling with kinetic models. This is demonstrated using a mass action law type of reaction in the bulk reactor liquid.

Unfortunately, only a small handful of open source projects exist for biotechnology process modeling, making it difficult for students and researchers to gain insight into the intricacies of biotechnology process engineering without access to proprietary software. BioSimulators (Shaikh et al., 2021) is an example of an open source project that attempts to provide models for biotechnology processes across a variety of systems, allowing the user to choose from a list of models to use. However, the platform is faulty in places and does not support compartment models. Meanwhile, a platform named CADET-Hub is developed by the CADET team at Forschungszentrum Jülich, following the call from Ino4Vac (European Vaccine Initiative, 2025a), it is a platform that enables integrated vaccine manufacturing modeling across the Inno4Vac project, including the one we are presenting here in this paper.

In our project, our goal aims to leverage the benefits of model-based simulations while minimizing known limitations (accuracy and realism, complexity and cost, validation, repeatability) as much as possible. Therefore, we have developed an online platform that integrates the results of CFD models, compartment models, and kinetic models. Each model is independent of the other, but also connects through the platform. The CFD model is performed offline through commercial CFD software and will provide steady-state results. The compartment models are written in Python and can be run in the cloud, as is the kinetics module provided by CADET (Leweke et al., 2025). By integrating these three models into a single platform, we can efficiently simulate the entire upstream biomanufacturing process with ease and convenience. To the best of our knowledge, this is the first open source online platform that integrates all essential upstream biotechnology process components.

2. The structure of our online platform

The primary objective of this project is to develop an open source platform that democratizes access to upstream bioreactor fermentation modeling for researchers and students engaged in reactor dynamics and heterogeneity. This platform aims to be publicly accessible and user-friendly. A critical step in this endeavor is to design the overall architecture of the platform prior to implementation. This approach will enable informed decisions regarding the selection of appropriate frameworks during the development phase.

Fig. 1 illustrates the user interaction with the front-end interface of the bioreactor platform. Users can select from various models and input specific parameters for the chosen models. Subsequently, the platform forwards user requests to backend services via a REST API, which manages the request accordingly.

The algorithms used for the calculations will be formed as a package and made accessible via the API. Additionally, users may utilize them directly in a separate environment if they wish to access the functionality independently of the platform and its inherent limitations. A database is integrated with the platform; it initially stores user records and handles authentication services. In subsequent versions of the platform, the database can be expanded to include groups and projects, thereby facilitating the sharing and collaborative development of simulations.

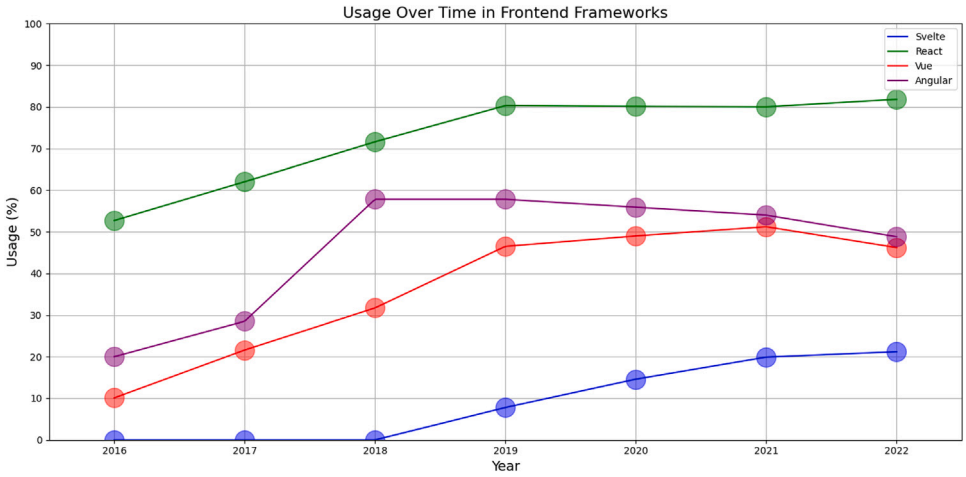


Fig. 2. The figure depicts the percentage of respondents using various frameworks.

2.1. Being open source

Open source development extends beyond public code access; it fosters transparency, community collaboration, and innovation. Transparency in open source projects ensures that anyone can review, audit, and understand the code, fostering trust and accountability within the community. This philosophy is at the core of the project and is closely aligned with the values of the research and educational community, where knowledge sharing and resource sharing are essential.

2.2. Open source licensing

Selecting an open source license is a critical task to be completed early in the project's timeline, as it establishes a clear legal framework for contributions, usage, and distribution of the software. Completing this task early in the project ensures that all contributors understand their rights. Moreover, the choice of license type can attract the appropriate users while deterring those who may not align with the project's goals. In essence, choosing the license model is akin to defining the project's identity; it is not merely a legal formality but the cornerstone of the project's ecosystem. The permissive nature and simplicity of the MIT license (MIT License contributors, 2025) make it a strong contender for this project. By adopting this license, we can ensure a clear legal framework for contributions, usage, and distribution of the software, thereby attracting the right contributors and users while establishing the project's identity and ecosystem.

2.3. Developing the platform

Developing a platform that allows users to perform compartment modeling for regime analysis is a novel task. This uniqueness is one of the key reasons why it is included in the Inno4Vac project (European Vaccine Initiative, 2025b). To ensure that the platform adheres to open source principles, the source code must be hosted on a publicly accessible platform. GitLab offers the same functionality as GitHub and more (GitLab Inc., 2025). Its standout features include a more integrated CI/CD solution, making it a more complete DevOps platform. Furthermore, DTU has its instances of GitLab with restricted access, which is beneficial for private or confidential projects, particularly those under institutional or corporate restrictions.

2.4. Front end and back end selection

The development of the platform requires a careful selection of the technology stack, especially the front-end framework. The choice of a front-end framework is important due to its impact on both user interaction and development efficiency. The framework must be easy to use, robust, user-friendly, and forward-looking, ensuring its relevance in the open source community now and in the future. Two key factors for choosing a front-end framework for an open source project are current popularity and long-term sustainability. To maximize participation, the framework should be actively maintained to ensure long-term viability. Otherwise, attracting and retaining contributors will be challenging. Additionally, the popularity of the framework is crucial, as it facilitates access of developers to documentation and technical support.

In recent years, software development has been transformed by the emergence of no-code and low-code platforms. These tools greatly simplify the process of creating applications and allows users with limited programming experience to understand and contribute to the development of applications. In the context of this simulation platform, the primary users and contributors are expected to have a background in chemical engineering, with low to intermediate programming experience.

This makes no-code and low-code tools particularly attractive, as they reduce the knowledge barrier to contribute to the project, and allow the field experts to directly build/shape the platform they use. By strategically incorporating no-code tools into the platform's architecture, the project can achieve a balance between technical performance and accessibility.

As shown in Figs. 2 and 3, Svelte has gained significant popularity over the years and is currently one of the most sought-after frameworks for learning. This is particularly important, as the success of the platform relies on continued community contributions. For sustained success, the framework must maintain its popularity both now and in the future. Given Svelte's high ratings in both usage and desirability, it is a well-supported and future-proof choice.

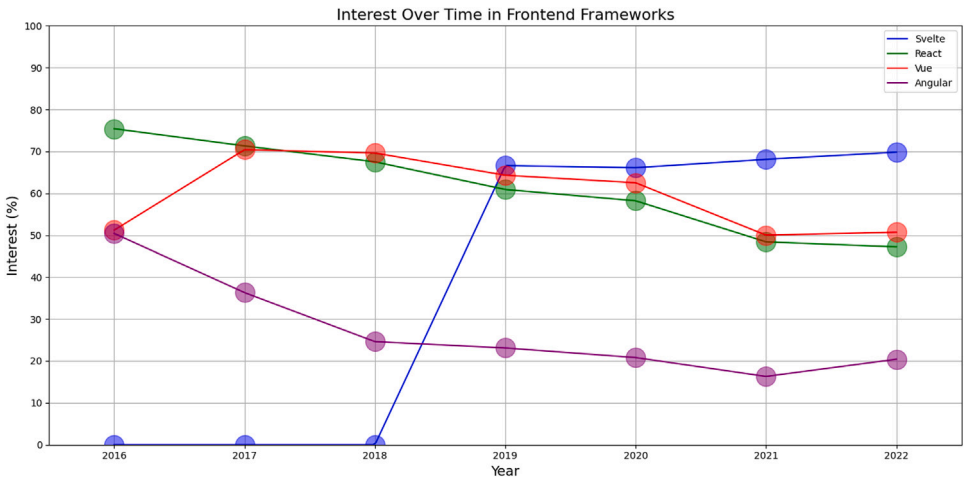


Fig. 3. This figure illustrates the percentage of respondents interested in learning various frameworks. Since its release, Svelte has consistently been the most sought-after framework for learning.

Svelte is also the only framework that offers an interactive tutorial that introduces developers to the core concepts related to Svelte. Documentation and instructions are provided on the left, and an IDE is provided on the right. For developers, this is an invaluable learning approach that offers hands-on experience and real-time feedback. Hence, Svelte has been selected as the front-end framework for this project.

Backend as a service (BaaS) refers to outsourcing the backend of an application to a cloud service provider (Cloudflare Inc., 2025). This service typically handles functionalities such as push notifications, database management, storage, user authentication, and hosting. BaaS reduces the developer's workload, allowing them to concentrate on front-end development, while the back-end remains configurable, usually through a web application. For most use cases, leveraging such a service enhances efficiency, except in highly specialized applications where complete control is required or where its use is prohibited. Supabase offers a unique feature that automatically generates an API from the database schema so that it can be accessed in the browser or as part of an API server (Supabase Inc., 2025a). The documentation for that API can also be automatically generated (Supabase Inc., 2025b).

What distinguishes Supabase is its strong engagement with the open source community, boasting more than 1000 contributors (Supabase Inc., 2025c). The platform's features, fixes, and updates are driven by user needs and contributions, providing significant flexibility as Supabase can be adapted to meet user requirements. Additionally, this open source approach ensures transparency and trust in data handling, as the code is continuously scrutinized and reviewed by a community of contributors and followers. Furthermore, Supabase offers a pay-as-you-go service, including a flat rate (Supabase Inc., 2025d), and provides a generous free plan for low-traffic solutions, similar to other popular backend services. Hence, Supabase has been selected as the back-end framework for this project.

2.5. REST API frameworks

An Application Programming Interface (API) defines how applications can interact and communicate with each other. An API typically specifies the types of requests that other applications can make, the data types that can be used as parameters for these requests, and the data types that can be expected in response to those queries. REST APIs are a subset of APIs that adhere to RESTful principles.

To expose functionality as a microservice, it is necessary to implement a REST API for our solution. Employing a microservice architecture enables the modularization and subsequent containerization of different components of the solution, thereby enhancing maintainability and the developer experience. In the context of science-based computing, we will primarily consider API frameworks that are compatible with scientific programming languages, particularly Python. In this project, we chose FastAPI after evaluating the advantages of various API frameworks in our domain, including Django REST, Flask, and FastAPI.

2.6. Cloud support

A key aspect of this project's vision is to ensure accessibility. This means allowing users to access the platform without the need for local hosting. This is particularly important for individuals without a technical background or those with limited resources. To achieve this goal, the platform and its submodules must be deployed and configured within a cloud-based environment.

There are numerous cloud providers offering a variety of features, with the most notable being AWS, Microsoft Azure, and Google Cloud. While none of these providers are open source themselves, they support open source projects and tools. All providers offer free credits upon registration and operate on a pay-as-you-go, on-demand pricing model (Microsoft Corporation (Azure team), 2025; Google LLC, 2025a; Amazon Web Service, 2025). For the project to thrive over time, securing financial support is essential, as cloud hosting is crucial to maintaining accessibility for non-developers. For this project, Google Cloud has been selected as the cloud provider. This decision is based on Google's strong commitment to the open source community (Google LLC, 2025b), as well as the extensive examples available on GitHub that demonstrate various use cases of their cloud platform (Google LLC, 2025c). This implies that computational tasks, including compartmentalization, post-processing, and kinetic modeling, should be executed on the backend of a hosted server. In contrast, tasks that are not computationally intensive, such as data formatting and conversions, can be efficiently managed on the client side.

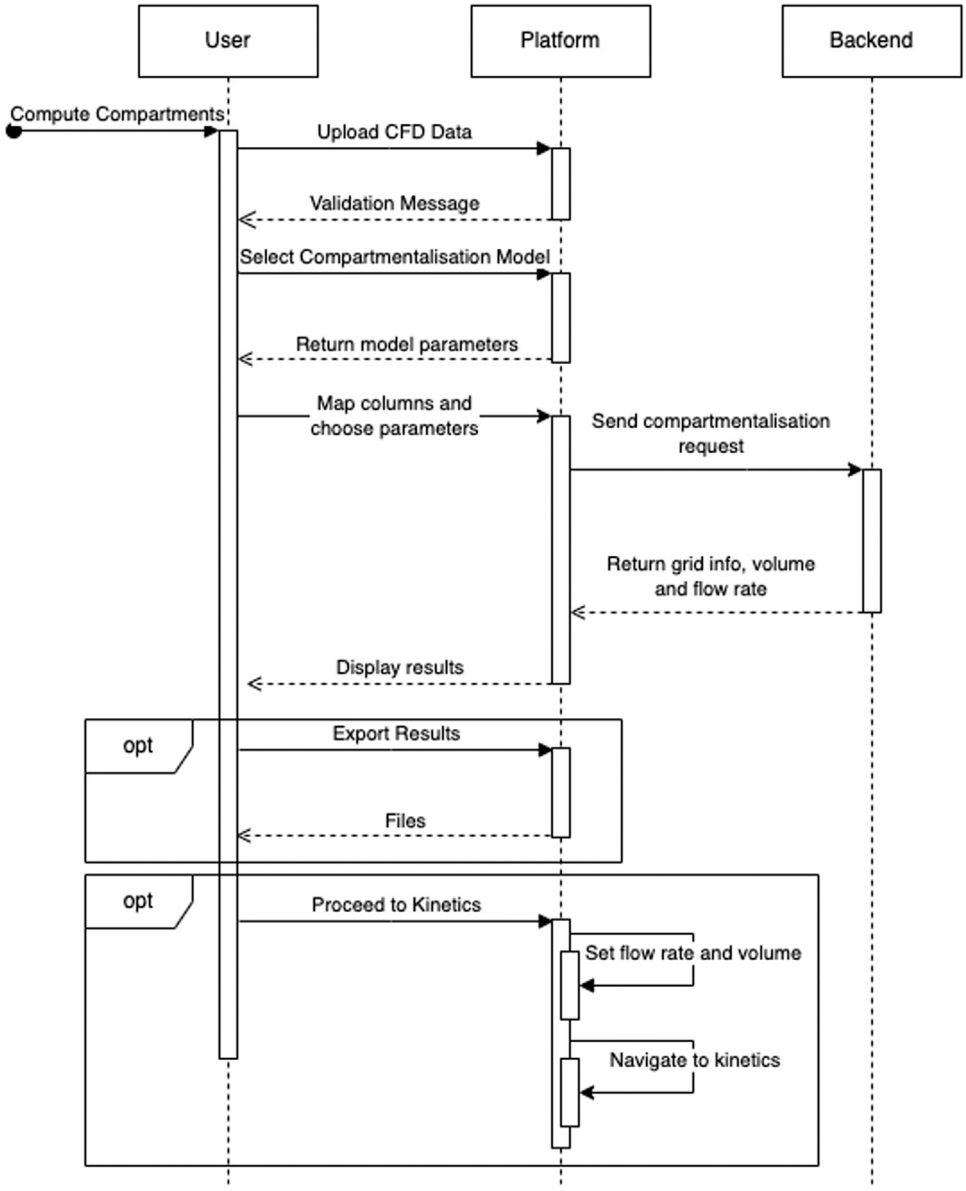


Fig. 4. Sequence diagram illustrating the calculation of compartments from provided CFD data.

2.7. Containerization and scaling

When developing applications, developers often face complex dependencies that require specific runtime requirements in which the application operates. Consequently, running applications on different computer systems can be challenging. Docker addresses this issue by encapsulating applications within a container, creating a consistent environment that behaves uniformly across all systems. This simplifies deployment and makes it easier to scale applications across different environments. A Docker container is instantiated from a Docker image, which is an executable package containing all the necessary applications (Docker). This approach allows the platform to be run offline within the research group, utilizing local computational resources to avoid cloud-related costs.

2.8. System interaction

To explain how the systems interact with each other, we examine the sequence of interactions for each of two primary scenarios: the compartmentalization scenario (Fig. 4) and the kinetic modeling scenario (Fig. 5). In both scenarios, there are three main participants: the user, the platform, and the back-end layer. For the sake of initial understanding and ease of use, the back-end layer is currently represented as a single component.

Fig. 4 depicts the communication flow between the user, the platform, and the backend in the compartmentalization scenario. The diagram indicates that the primary interaction occurs between the user and the platform, with a crucial interaction between the platform and the back-end, which handles computationally intensive tasks. Most interactions between the user and the platform are relatively simple, such as data input, selections, and minor operations, such as exporting previously stored data on the platform. The kinetic modeling workflow (Fig. 5) closely resembles

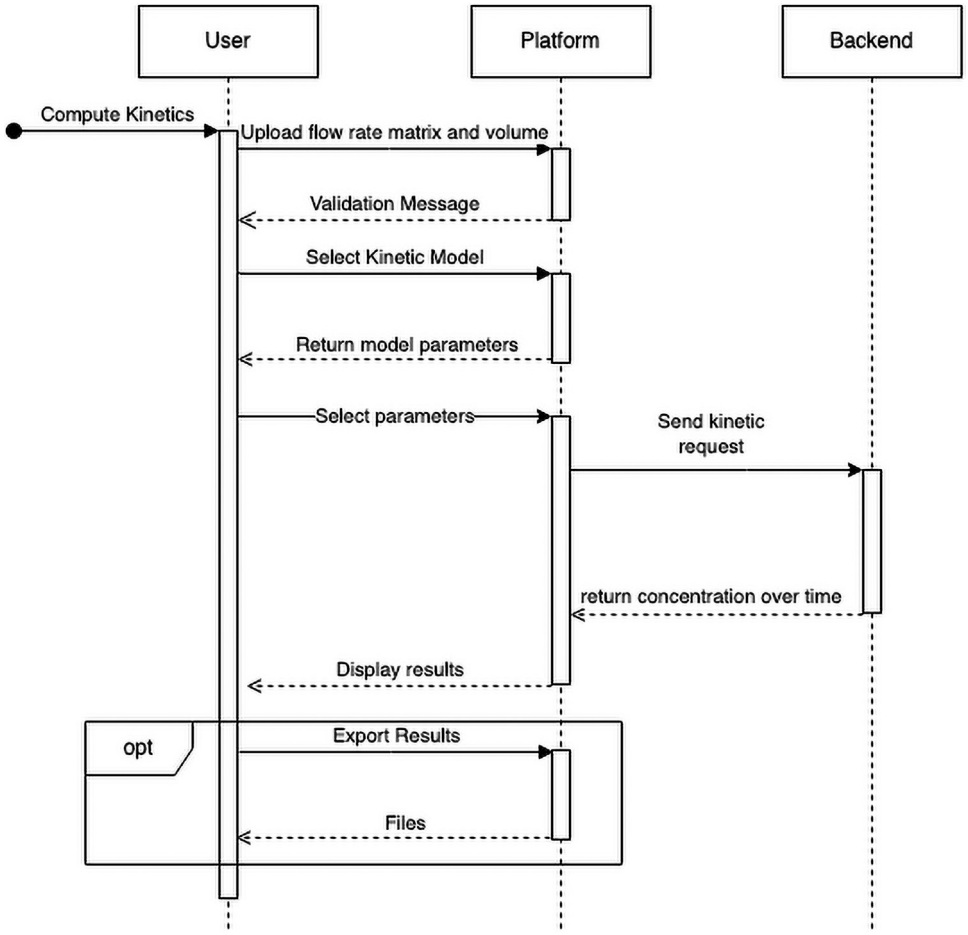


Fig. 5. Sequence diagram illustrating the scenario of using the kinetic model to calculate concentration over time for each compartment.

that of the compartmentalization scenario. The primary differences lie in the type of data uploaded and the back-end endpoint. Once the results are obtained, users can either return to the previous step or export the results.

The similarity between the two scenarios allows for the use of reusable components during platform development. Reusable components simplify the maintenance process, as modifications to a single component, such as the data upload component, will propagate throughout the entire platform.

2.9. Design space

The field of bioreactor modeling is constantly evolving, necessitating that the platform and project be designed to allow easy addition, comparison, and testing of new algorithms. Achieving this requires a modular design approach that allows additions, replacements, and removals without disrupting or compromising the rest of the system. This can be achieved through various methods, such as a plugin-based architecture for integrating new algorithms or a well-defined API that enables interaction with the platform. This project incorporates both strategies to ensure extensibility and modularity.

3. Models included in the platform and implementation

Three major mathematical models, or their outputs, have been implemented in the platform: CFD results, compartment models, and kinetic models. The CFD simulation is run locally and its results are subsequently uploaded to the platform for compartmentalization. Following this, the kinetics can be applied to the selected compartment to evaluate production performance. Users can easily log in to the platform by registering their email address, similar to other social platforms such as Facebook or Twitter. Once logged in, Fig. 6 shows the homepage interface, which includes a navigation bar, user profile access, and quick links to key features such as simulation setup and a video tutorial.

3.1. CFD modeling

Due to its inherent complexity and computational demands, implementing Computational Fluid Dynamics (CFD) on a cloud-based platform presents significant challenges. The high computational demands of CFD simulations require substantial processing power and memory, which can be difficult to achieve consistently in a cloud environment. Additionally, the need for real-time data processing and the large volume of data generated by CFD models can strain cloud resources, leading to potential latency issues and increased costs. Achieving accuracy and reliability in

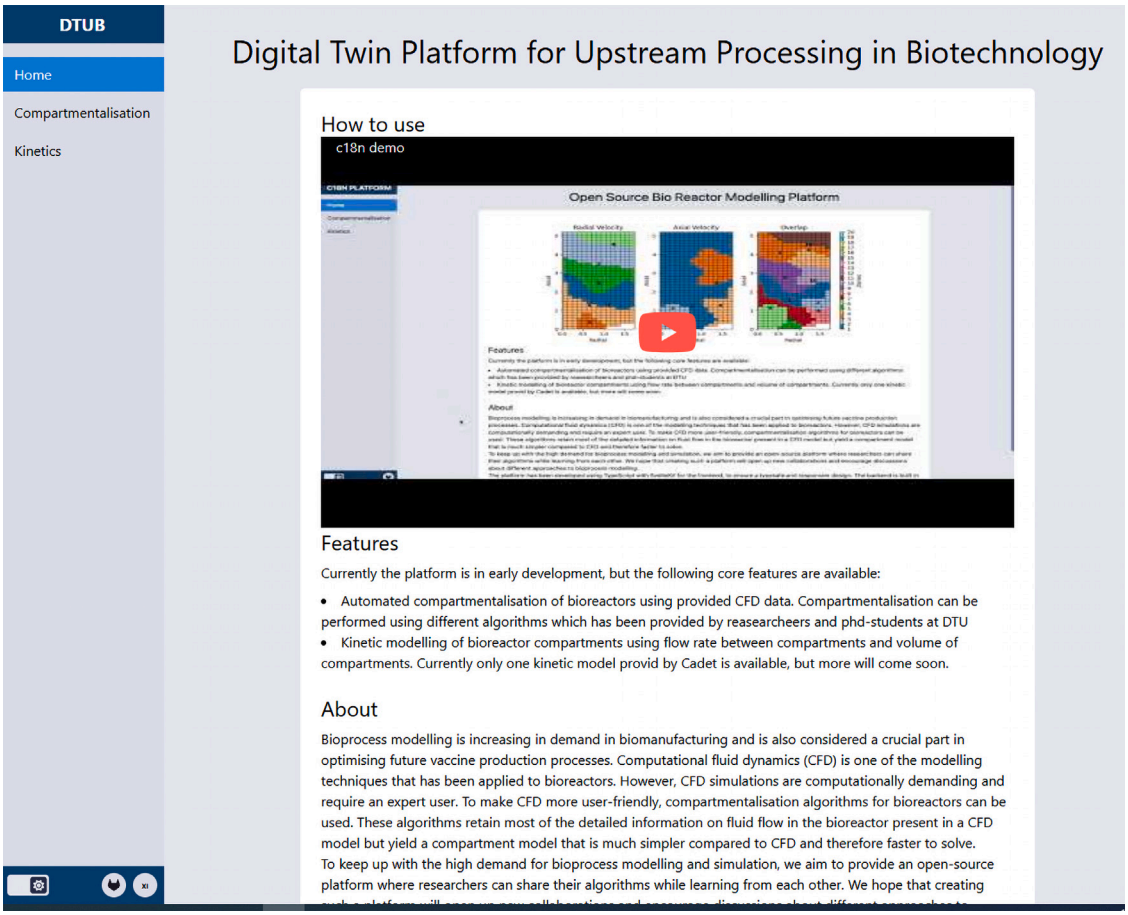


Fig. 6. The homepage of our DTUB platform.

a cloud-based CFD setup requires sophisticated infrastructure as well as robust data management practices. These requirements further complicate the implementation. For this reason, the CFD simulations are performed offline by professional CFD software, such as ANSYS Fluent or CFX.

CFX and Fluent were selected due to their established use in bioreactor simulation over the past two decades and their mature applications in bioreactor simulation, which are well-suited for this project. Consequently, the platform we are developing requires users to upload their CFD results as the initial step for compartment simulation. Given the extensive user base of ANSYS, we have opted to use results from CFX and Fluent as the platform's input. However, the user can choose which CFD software they want to use, as long as they can export the required data in the required data format.

In the online platform, CFD data is processed as an exported spreadsheet from ANSYS. Users are not required to sort the data in a specific order, as the platform provides options to select the matrix, as illustrated in Fig. 7. Each column in the spreadsheet is equipped with a drop-down button. An additional step has been incorporated into the algorithm to ensure that each data variable used for the algorithm has a unique relationship with one of the columns in the table; in such cases, only the most recent selection will be retained. Fig. 7 also illustrates that users can enter model-specific parameters in the upper section of the shown interface. These parameters are retrieved from the API after the user selects the model. The platform fetches the names, default values, and a brief help text for each parameter.

3.2. Compartment modeling

CFD-based compartment models are hybrid modeling approaches that combine Computational Fluid Dynamics (CFD) with Compartment Models (CM) to simulate the motion of fluids (Bezzo et al., 2004; Bezzo and Macchietto, 2004). Such models can be used to simulate the interactions between transport limitations and reaction kinetics in reactor and bioreactor systems (Jourdan et al., 2019; Delafosse et al., 2014; Öner et al., 2019). By integrating CFD and CMs, the resulting models combine computational efficiency and accurate hydrodynamics, which is required for coupling with reaction kinetics. Previous work by Tajsoleiman (2018) and Nadal-Rey et al. (2021) illustrate the ability of CFD-based CMs to couple hydrodynamics and reaction kinetics to resolve potential spatial gradients occurring within the bioreactor.

The simplicity of the compartment models allowed not only the coupling with reaction kinetics, but also allowed simulations over extended time spans, allowing for dynamic simulations over the entire fermentation time (often hours/days). Fig. 8 shows the zoning map following compartmentalization. The platform provides options to either return and modify the data or advance to the subsequent step.

3.3. Kinetic modeling

By pressing 'Proceed to Kinetics', the user will be directed to step 2 of the kinetic process, where the kinetic model can be selected. To facilitate data transfer between pages, we utilize a state management framework provided by SvelteKit, known as **store**, which serves as a shared state

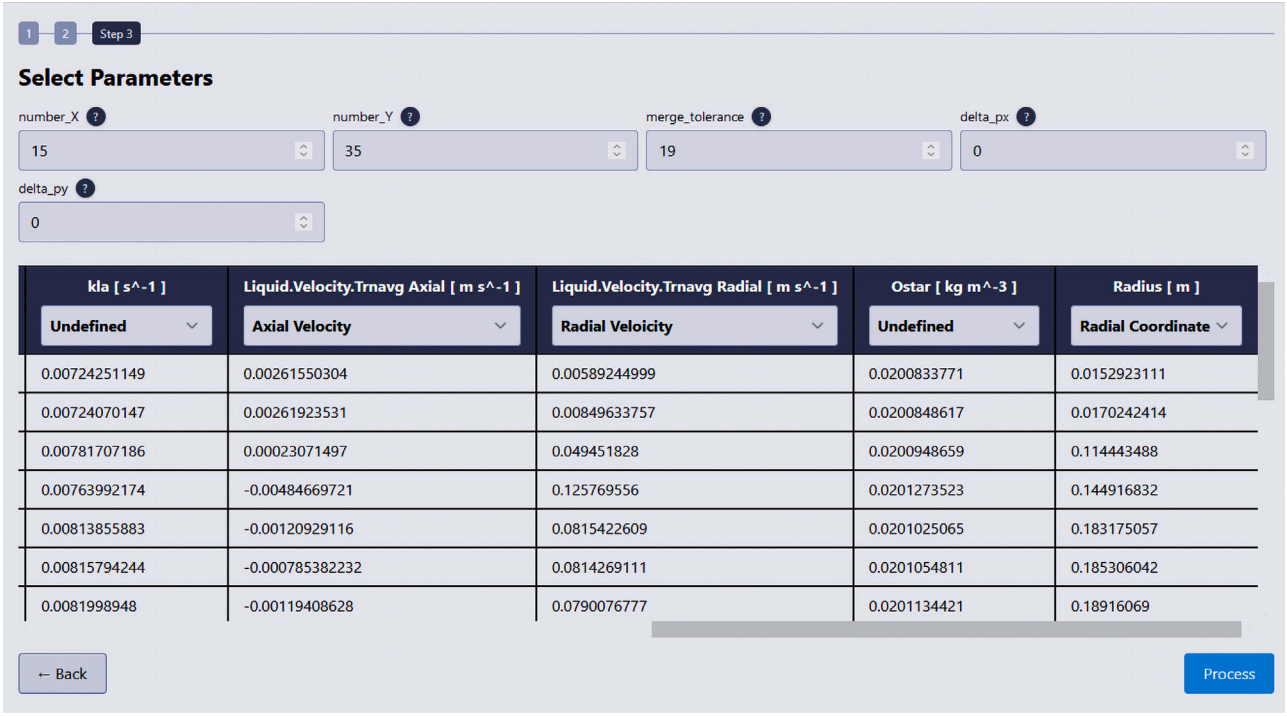


Fig. 7. Parameter input for compartmentalization and input column table.

accessible for reading and writing from various sources. The store is implemented in a TypeScript file named `store.ts`, located on the kinetic route. It comprises three writable variables. Defining these variables using **writable** establishes the interface that allows other sources to assign values to them.

We can then import the store on the compartmentalization page and write the result values to the shared store. The function `navigateToKinetics`, as depicted in Listing 2, maps the results of the compartmentalization to the required format specified by the kinetics store. After updating the store with the result values, we set the current step to 1, skipping the initial data upload step. Finally, we use the **goto** function to navigate to the kinetics page. The three highlighted features have been selected to demonstrate how small features can enhance the user experience by automating tedious, repetitive tasks common to everyday workflows.

Listing 1: Kinetic store for shared variables

```
import { writable } from 'svelte/store';

export const flow_rate_matrix = writable<Array<Array<number>>>();
export const volumes = writable<number[]>([]);
export const currentstep = writable<number>(0);
```

Listing 2: Writing results to the shared store

```
import { flow_rate_matrix, volumes, currentstep }
from '../kinetics/store.js';
...
function navigateToKinetics() {
    if (resultItem) {
        let temp_flow_rate_matrix: number[][] = [];
        for (let i = 0; i < resultItem.flow_rate_matrix[0].length; i++)
        {
            temp_flow_rate_matrix.push(resultItem.
            flow_rate_matrix.map((row) => Number(row[i])));
        }
        $flow_rate_matrix = temp_flow_rate_matrix;
        $volumes = resultItem.volume.map((row) => Number(row[1]));
        $currentstep = 1;
        goto('/kinetics');
    }
}
```

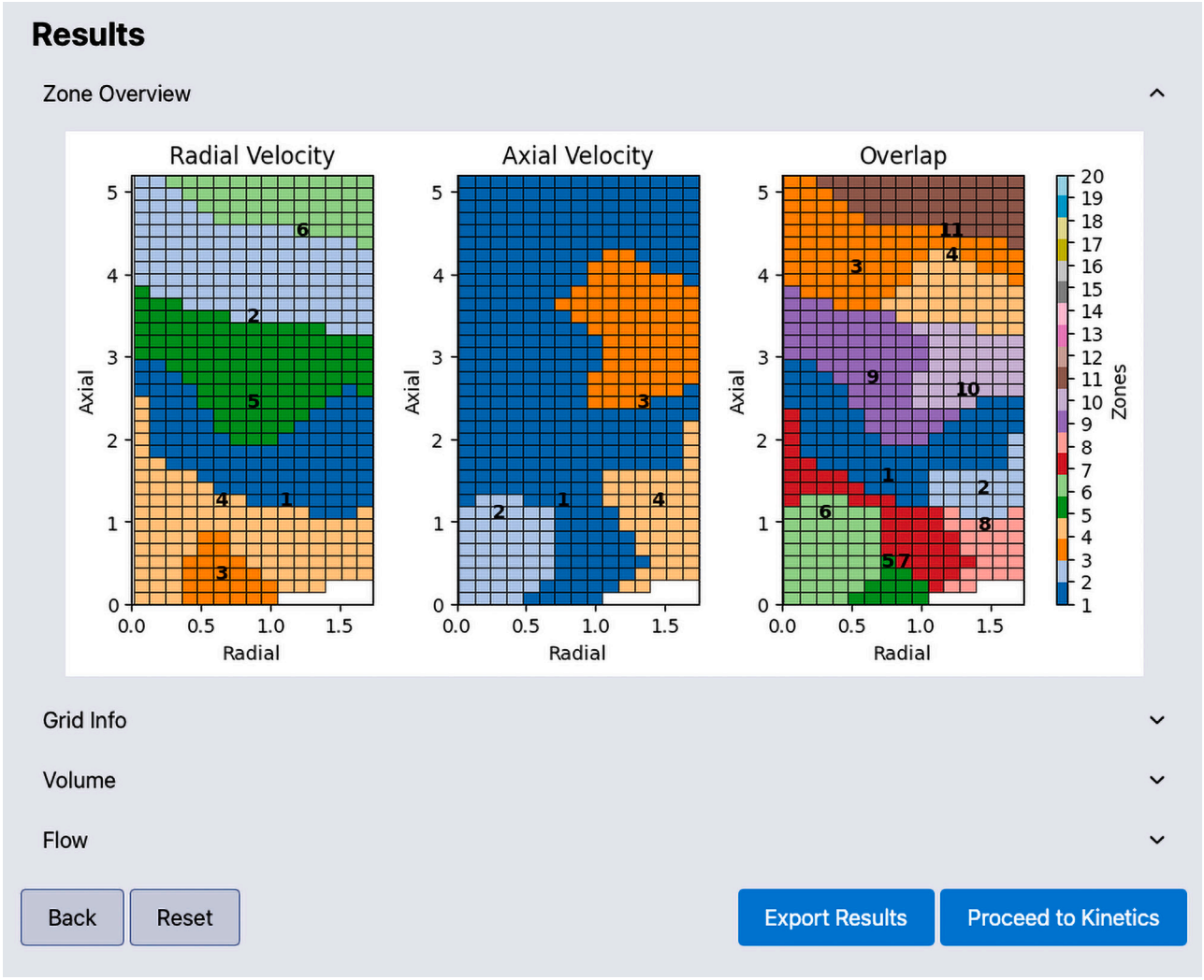


Fig. 8. Navigation from compartmentalization result directly to kinetic modeling.



Fig. 9. Proceed to a certain kinetics.

To demonstrate the functionality of the kinetics models, we present an example that illustrates how kinetic models can be implemented within the platform. This case study uses the previously published SOKA kinetic model (Sonnleitner, 1985), which captures the Crabtree effect in yeast metabolism. The fluid dynamics results shown in Fig. 8 serve as input for this analysis. Following the platform's workflow, we apply the SOKA model in the Kinetics section, as illustrated in Fig. 9. On the parameter selection page (Fig. 10), all configurable parameters are displayed, allowing users to adjust settings as needed. The resulting outputs for this specific kinetic setup are presented in Fig. 11. While this example serves as a reference, users are encouraged to explore and implement their own kinetic models using the platform.

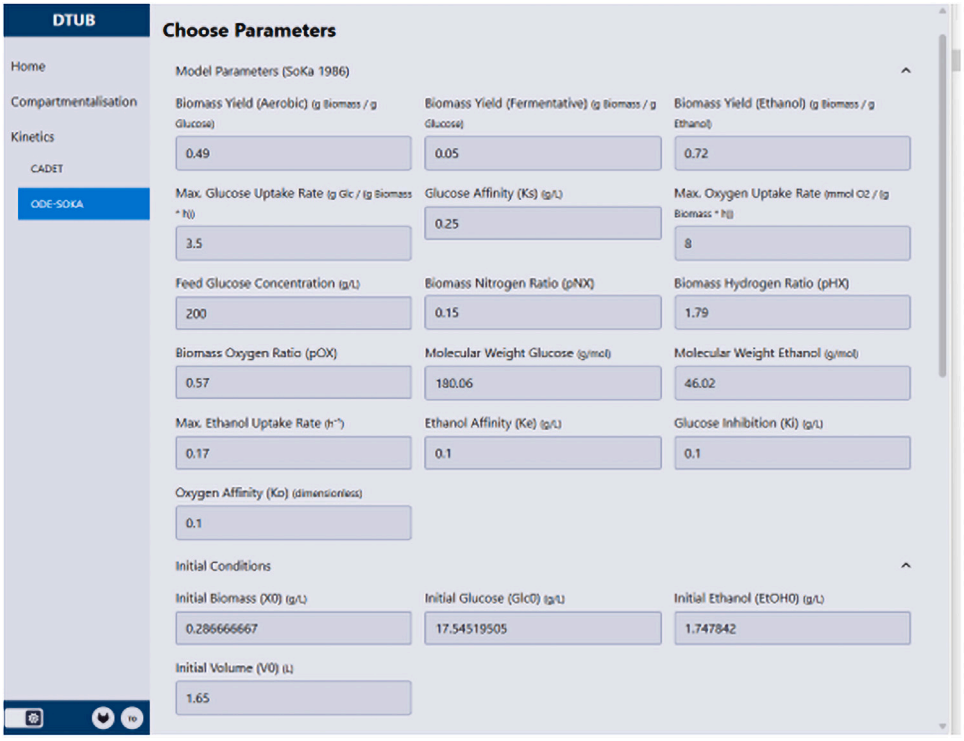


Fig. 10. Configuring parameters for the SOKA model.

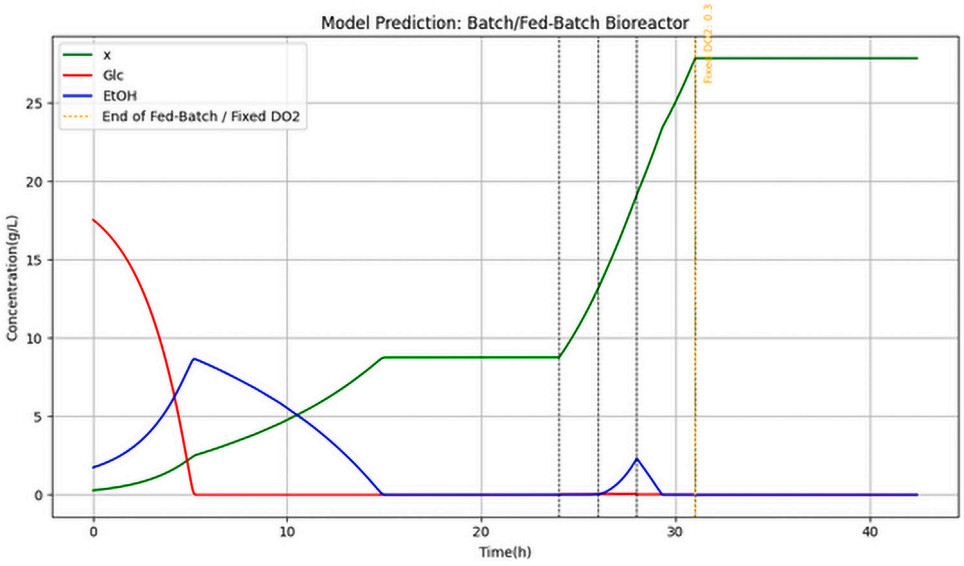


Fig. 11. Results produced from the SOKA model through the DTUB platform.

4. Testing

Following best practices, unit tests and integration tests have been implemented using Pytest to evaluate the various components of the system.

4.1. Testing CBSM

CBSM, which stands for the Compartment-Based Scale-down Algorithm, is a sophisticated algorithm that encompasses multiple sub-functions. When implementing such a complex algorithm, it is essential to implement rigorous tests to detect incorrect or unintended behavior. Designing straightforward tests, where the expected output can be easily deduced, is advantageous as it helps to uncover any fundamental errors that may occur during the implementation process.

Listing 3 presents one of the test cases for the split method. In this scenario, the center points of three cells, designated as points *A*, *B*, and *C*, are located within zone 1. The delta values are set to 1 in both coordinate directions. According to the definition of the split method, if a point

exceeds the delta distance in either coordinate direction, it should be assigned to a new zone.

$A = (1, 1)$

$B = (1.5, 1.5)$

$C = (3, 3)$

From the three points, it is readily deducible that point C has a distance greater than the delta values in either of its corresponding directions to the nearest cell within the same zone. Consequently, upon executing the split method, we should expect the three points to be assigned to the following zones:

$A = 1$

$B = 1$

$C = 2$

Listing 3: Split test case

```
def test_split():
    center_info = np.array([[1.0, 1.0, 1.0],
                            [1.5, 1.5, 1.0],
                            [3.0, 3.0, 1.0]])
    delta_x = 1.0
    delta_y = 1.0
    split_result = split(center_info, delta_x, delta_y)

    assert split_result[:, 2].tolist() == [1.0, 1.0, 2.0]
```

Such test cases have proven to be invaluable in clarifying function behavior and confirming implementation correctness.

4.2. Testing flow

As part of this project's commitment to ensuring seamless integration, test cases for the mass corrector have been established. Given the complexity of flow calculations, particularly on larger grids, the test case setup utilized smaller grids with dimensions of 3×3. This approach facilitated easier debugging and provided clearer insight into the precise functionality of the implemented methods. In total, five flow-related functions were tested and all 27 test cases passed successfully. A representative test case is shown below:

```
    def test_down_flow(self):
        rows, cols = 3, 3
        nodeX = np.array([0.0, 1.0, 2.0, 3.0])
        nodeY = np.array([0.0, 1.0, 2.0, 3.0])
        gU = np.ones((rows + 1, cols))
        gV = np.zeros((rows, cols + 1))

        flow_instance = Flow()
        flow_instance.nX = rows
        flow_instance.nY = cols
        flow_instance.nodeX = nodeX
        flow_instance.nodeY = nodeY
        flow_instance.gU = gU
        flow_instance.gV = gV

        expectedFx = np.ones((rows + 1, cols))
        expectedFy = np.zeros((rows, cols + 1))
        flow_instance.calculate_2d_flow()

        actualFx, actualFy = flow_instance.Fx, flow_instance.Fy

        assert np.array_equal(expectedFx, actualFx)
        assert np.array_equal(expectedFy, actualFy)
```

As indicated by the code, this function tests a scenario where the flow is downward. Initially, a grid of size 3×3 is established, resulting in four face locations per direction, with one face between each pair of neighboring cells. Each cell has a size 1×1 units, and the U velocity field is initialized with values of one, indicating a downward flow. Consequently, the flow in the x-direction is expected to be 1 for each cell, while it is 0 in the y-direction. Comparing the expected output with the actual output confirms this and validates the test case.

While this test case focuses on a single flow direction, seven other test cases explored the different directions. Most of the tested functions follow a simple verification strategy, where variables are initialized and the output is directly compared to the expected results. However, the Gauss–Seidel tests necessitate a different methodology due to their approximate nature. Nevertheless, testing this function is essential due to its critical role in ensuring mass conservation.

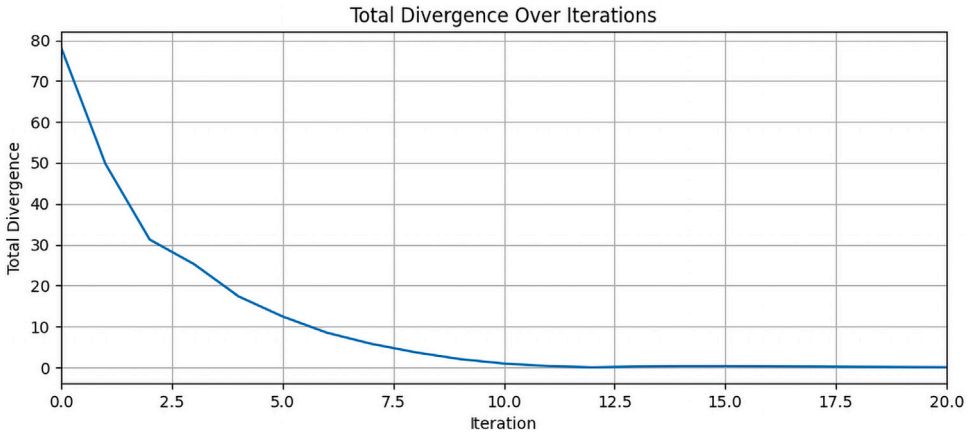


Fig. 12. The divergence norm as a function of the iterations. In this figure, a 10x10 grid instantiated with random uniformly distributed initial velocities between $[-10; 10]$ m/s was used. A 1000 iterations were run and a relaxation factor of $\omega = 1.6$ was used.

4.2.1. Testing Gauss–Seidel

Evaluating the effectiveness of the Gauss–Seidel method for flow approximations within the compartment poses significant challenges due to the inherent iterative nature of the method. In fact, convergence is ensured only if the matrix is strictly diagonally dominant, symmetric, and positive definite (Wolfram, 2025). However, it is essential to perform tests to verify that the implementation is correct and that the method behaves as expected. There are various methods to test the Gauss–Seidel method, with some of the most straightforward being the assessment of the rate of convergence and the verification that the solution's error is sufficiently minimized.

Error minimization can be evaluated in several ways. If an analytical solution to the system of equations is known, the exact or true error e_k can be computed as:

$$e_k = x^* - x_k \tag{1}$$

Where x^* is the analytical solution and x_k is the approximate solution at iteration k.

However, in many cases, the analytical solution is either unknown or nonexistent. It is therefore important to verify that the implementation functions correctly in scenarios where no analytical solution is available. There are several methods to achieve this. One such method involves computing the residual error as a proxy for the true error. The residual error is typically given as:

$$r_k = b - Ax_k \tag{2}$$

As previously established, the divergence $\nabla \mathbf{u}$ must be zero to satisfy the flow incompressibility condition. Consequently, the vector b should be zero. However, A is not easily computable as it does not appear explicitly in the implemented method used to adjust the velocities. Instead, the effect of A can be observed through the changes made in the velocities. This does not imply that a residual cannot be computed; rather, it will not be in the traditional sense. In our approach, the velocity fields are adjusted to reduce the divergence to zero. Consequently, the divergence of the velocity field serves as a practical residual metric. The divergence provides a measure of how well the incompressibility condition is satisfied at each point on the grid. It offers a quantifiable error, allowing us to confirm that the divergence decreases steadily and reaches near-zero values, thereby validating the implementation.

Since the divergence is calculated on a per-cell basis, it is necessary to save the divergence for each cell and for each iteration. To verify whether the divergence approaches zero, we can compute the $L2$ norm of the matrix of cell divergences for each iteration. Let A^k denote the kth iteration of a matrix A with dimensions $m \times n$. The L_2 norm is given by:

$$\|A^k\|_2 = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij}^k)^2} \tag{3}$$

Solving this flow for a simple 10×10 grid with random velocities and plotting the norm as a function of iterations yields a graph similar to the one depicted in Fig. 12:

Clearly, the divergence appears to approach zero well before the number of iterations reaches 1000. To confirm robustness, the method can be tested on multiple initial velocities. Solving for $n = 20$ grids offers reassurance, shown in Fig. 13. These results indicate that the Gauss–Seidel method reliably reduces divergences towards zero.

As part of the unit tests for the Gauss–Seidel method, these characteristics are specifically targeted. For the implementation of the unit tests, the priority was to select a smaller problem that could be easily verified manually, as was done for the other functions.

We used a simple 3×3 grid with velocity vectors initialized in upward, downward, leftward, and rightward directions. The number of iterations was set to $k = 10000$ with a relaxation factor of $\omega = 1.6$. The error is then checked to make sure that it does not increase every $n = 10$ iteration and that the final divergence is close to zero $\|A^k\|_2 < 1 \cdot 10^{-8}$ in all test cases. Depending on the grid structure, the grid elements and the parameters of the Gauss–Seidel method, these values will change.

There are also more indirect methods to evaluate whether the error decreases, such as testing the order of convergence decreases. Similarly, it is also possible to test that the velocity field converges to a steady state.

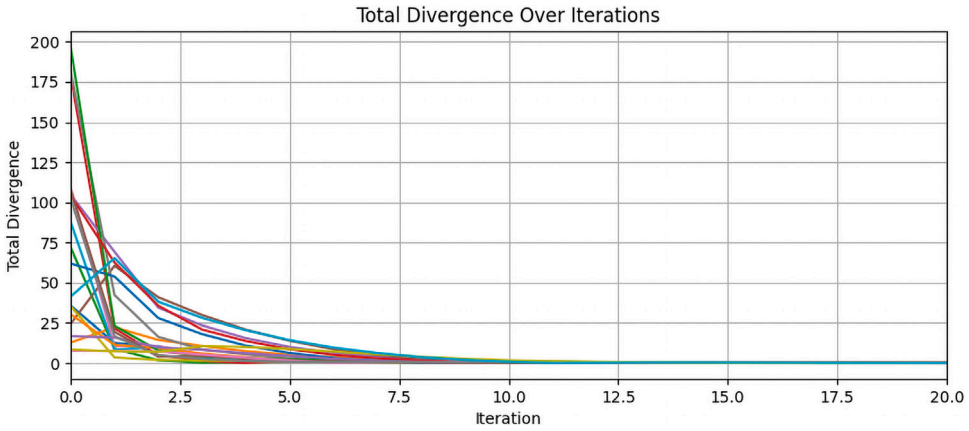


Fig. 13. Running the method with 20 different grids of size 10×10 and with uniformly distributed random initial velocities.

4.3. API integration test

The API serves as the central node of the system, importing compartmentalization, flow, and kinetic packages, which expose their functionality to the platform. To ensure reliability, we conduct large-scale integration tests on the API, where data is input in the format specified for each endpoint, and the correctness of the returned results is verified. To achieve this, we now focus on system-level integration and will also develop test cases utilizing full CFD data.

The following test case traverses the entire system, invoking both the compartmentalization and kinetic endpoints, thereby engaging the compartmentalization, flow, and kinetic functions.

Listing 4: Integration test on API

```
def test_scaledown_and_kinetic():
    filename = "app/resources/2_A310_60_BM_160rpm_FlowStats_003.xlsx"
    # Read CFD data
    data = pd.read_excel(filename)
    CFD_data = data.values
    # Extract data
    # start at 5 to avoid the header
    X = CFD_data[5:, 9].tolist()
    Y = CFD_data[5:, 1].tolist()
    Vx = CFD_data[5:, 7].tolist()
    Vy = CFD_data[5:, 6].tolist()

    response = client.put(
        "/compartmentalise/scaledown/tannaz",
        json = {
            'data' : TannazData(X=X, Y=Y, Vx=Vx, Vy=Vy).dict(),
            'params' : TannazParams(number_X=15, number_Y=35,
            merge_tolerance=19, delta_px=0, delta_py=0).dict()
        }

    )
    assert response.status_code == 200

    cadet_response = client.put(
        "/kinetics/cadet",
        json = {
            'modelParams' : CADETModelParams
            (volumes=np.array(response.json()['volume'])[:, 1].
            tolist(), flow_rate_matrix=np.array(response.json(
            ['flow_rate_matrix']).tolist(), number_of_components=2).
            dict(), 'modelTracer' : CADETModelTracer
            (compartment_index=4, concentration=[10, 10],
            injection_duration=10, flow_rate=0.1e-3, start_time=0).
            dict(),'modelKineticReaction':
            CADETModelKineticReaction(reactants=[0, 1], products=[-1,1],
            k_fwd=2, k_bwd=1).dict()
```

```
        }
    )
    assert cadet_response.status_code == 200
    assert cadet_response.json()['image'] != None
    assert len(cadet_response.json()['time']) > 0
    assert len(cadet_response.json()['concentration'].keys()) > 0
```

In the test case illustrated in Listing 4, we utilize a complete CFD data set and extract the necessary columns into lists that can be passed to the API using the data input parameter. The first step involves sending a request to the compartmentalization endpoint and asserting that the response status code is 200 (successful). If the compartmentalization response is successful, we proceed to send a request to the CADET kinetic model endpoint using the result of the compartmentalization response. Finally, we ensure that the kinetic response is also successful and that the result is not empty and contains valid output.

4.4. Automated testing

To verify correct functionality, we implemented automated testing, including unit tests for individual components and integration tests for system-wide interactions. Since our back-end is implemented entirely in Python, we utilize the Pytest framework for unit tests and a combination of Pydantic, FastAPI, and Pytest for integration tests. Ideally, these tests are executed on GitLab runners whenever a new push is made to the repository. However, due to restrictions on outgoing traffic from the virtual machines hosting the GitLab runners, we are unable to install any packages or fetch default Docker images. Consequently, until the project is migrated to another host, such as GitHub or the official GitLab host, it is not possible to set up testing as part of the CI/CD pipeline.

As a workaround, we have included Pytest in our Dockerfile, ensuring that the building of the Docker image will only complete if all tests pass. This approach ensures that deployments only proceed if all tests pass.

5. Further development

Several areas remain for further development and enhancement of the platform. For example:

- **CI/CD**: To attract more contributors from the user base, an alternative to gitlab.gbar.dtu.dk should be considered, allowing for the deployment of a true open source structure.
- **Scalability**: The group has continued to develop compartment models. The platform should provide access to all models that have been developed and are ready to be released as open source. This will also facilitate the comparison of all the models.
- **Project Collaborations**: The platform could be enhanced with social and collaborative features, such as the ability to create teams and share results. To implement these features, it would be necessary to extend the platform and the database. Additionally, some form of save capability should be implemented, allowing users to save the compartments they have computed. This would involve saving the uploaded data, selected parameters, and results.
- **Educational Applications**: The platform provides students with an opportunity to gain an understanding of upstream processing in biotechnology without requiring extensive knowledge of mathematical simulation. This is ideal for demonstrating and teaching at the entry levels of engineering education.
- **Integration**: During our project, we collaborated closely with Forschungszentrum Jülich to integrate CADET and CADET-Process with our platform and API. The integration with CADET-Process could still be extended, as the package offers numerous features. Additionally, other organizations and researchers are developing innovative solutions within the bioreactor simulation domain. Exploring such tools for new integrations could be beneficial in attracting more users from their respective user bases.
- **Model comparison**: After new models have been implemented in the platform, a comparison feature could be developed. This feature should allow users to view side-by-side comparisons and highlight the major differences between the models. Such a comparison tool would help researchers to understand how different models and parameters impact the resulting compartments. Incorporating performance metrics, such as computational time and resource utilization, could provide significant insights. These metrics are instrumental in evaluating the efficiency of various models, thereby guiding the selection process towards the most suitable option for the intended application.

6. Conclusions

We present an innovative online platform for the development of a biopharmaceutical process, which, to the best knowledge of the authors, has not been previously published. The presented platform bridges a critical gap in upstream biotechnology process simulation by integrating CFD, compartmental, and kinetic models into an accessible, open source environment. Its continued development is expected to foster collaboration, accelerate research, and support educational initiatives across the biomanufacturing community. The platform is currently hosted on GitLab at https://gitlab.gbar.dtu.dk/inno4vac/compartment-based-scaledown-model and is intended to become fully open source in future iterations. The paper further details the platform architecture, model deployment, and system validation through integration tests. Ultimately, platform tests are performed to validate the system. Ongoing developments will continue to enhance the platform, with progress updates available at https://platform-image-pfwhpn5qeq-ew.a.run.app/.

CRediT authorship contribution statement

Xiyan Li: Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Conceptualization. **Sebastian L. Jensen:** Validation, Methodology, Investigation. **Noah B. Christiansen:** Validation, Methodology, Investigation. **Elham Ramin:** Writing – review & editing, Supervision. **Johan le Nepvou de Carfort:** Writing – review & editing. **Johannes Schmölder:** Writing – review & editing, Supervision. **Eric von Lieres:** Writing – review & editing. **Krist V. Gernaey:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This project has received funding from the Innovative Medicines Initiative 2 Joint Undertaking under grant agreement No 101007799. This joint venture receives support from the European Union's Horizon 2020 research and innovation program and EFPIA. This communication reflects the author's view, and neither IMI nor the European Union, EFPIA, or any Associated Partners are responsible for any use that may be made of the information contained herein.

Data availability

Data will be made available on request.

References

Amazon Web Service, Amazon EC2 Pricing. [Online]. Available: https://aws.amazon.com/ec2/pricing/. (Accessed 16 December 2025).
Bezzo, F., Macchietto, S., 2004. A general methodology for hybrid multizonal/CFD models: Part ii. automatic zoning. Comput. Chem. Eng. 28, 513–525. http://dx.doi.org/10.1016/j.compchemeng.2003.08.010.
Bezzo, F., Macchietto, S., Pantelides, C.C., 2004. A general methodology for hybrid multizonal/CFD models: Part I. theoretical framework. Comput. Chem. Eng. 28, 501–511. http://dx.doi.org/10.1016/j.compchemeng.2003.08.004.
Bisgaard, Jonas, Zahn, James A., Tajsoleiman, Tannaz, Rasmussen, Tue, Huusom, Jakob K., Gernaey, Krist V., 2022. Data-based dynamic compartment model: Modeling of E. coli fed-batch fermentation in a 600 m3 bubble column. J. Ind. Microbiol. Biotechnol. 49 (5), 1–11. http://dx.doi.org/10.1093/jimb/kuac021.
Cloudflare Inc., BaaS. What is BaaS? – Backend-as-a-Service vs. serverless. [Online]. Available: https://www.cloudflare.com/en-gb/learning/serverless/glossary/backend-as-a-service-baas/. (Accessed 16 December 2025).
Delafosse, Angélique, Collignon, Marie Laure, Calvo, Sébastien, Delvigne, Frank, Crine, Michel, Thonart, Philippe, Toye, Dominique, 2014. CFD-based compartment model for description of mixing in bioreactors. Chem. Eng. Sci. 106, 76–85. http://dx.doi.org/10.1016/j.ces.2013.11.033.
Docker, Developers bring their ideas to life with Docker. [Online]. Available: https://www.docker.com/why-docker/. (Accessed 16 December 2025).
European Vaccine Initiative, Inno4Vac Open Call. [Online]. Available: https://www.inno4vac.eu/open-call. (Accessed 16 December 2025).
European Vaccine Initiative, Inno4Vac project. [Online]. Available: https://www.inno4vac.eu/. (Accessed 16 December 2025).
GitLab Inc., Why GitLab. [Online]. Available: https://about.gitlab.com/why-gitlab/. (Accessed 16 December 2025).
Google LLC, 2025a. Google Cloud. [Online]. Available: https://cloud.google.com/pricing/. (Accessed 16 December 2025).
Google LLC, 2025b. Google Open Source. [Online]. Available: https://opensource.google/. (Accessed 16 December 2025).
Google LLC, 2025c. Google Cloud Platform. [Online]. Available: https://github.com/GoogleCloudPlatform. (Accessed 16 December 2025).
Jourdan, Nicolas, Neveux, Thibaut, Potier, Olivier, Kanniche, Mohamed, Wicks, Jim, Nopens, Ingmar, Rehman, Usman, Le Moullec, Yann, 2019. Compartmental Modelling in chemical engineering: A critical review. Chem. Eng. Sci. 210, 115196. http://dx.doi.org/10.1016/j.ces.2019.115196.
Le Nepvou De Carfort, Johan, Pinto, Tiago, Krühne, Ulrich, 2024. An Automatic Method for Generation of CFD-Based 3D Compartment Models: Towards Real-Time Mixing Simulations. Bioengineering 11 (2), 169. http://dx.doi.org/10.3390/bioengineering11020169.
Leweke, Samuel, Schmölder, Johannes, Rao, Jayghosh, Hassan, Jazib, Heymann, William, Breuer, Jan, Azzouzi, Salah, von Lieres, Eric, CADET. CADET – An open platform for integrated process modelling and simulation. [Online]. Available: https://zenodo.org/records/8179016. (Accessed 16 December 2025).
Liu, Guanyang, Wilhite, Benjamin A., 2023. A generalized zoning framework for development of CFD-based reduced-order compartment models. AIChE J. 69 (7), 1–14. http://dx.doi.org/10.1002/aic.18063.
Microsoft Corporation (Azure team), Azure pricing. [Online]. Available: https://azure.microsoft.com/en-us/pricing/#product-pricing. (Accessed 16 December 2025).
MIT License contributors, The MIT License. [Online]. Available: https://opensource.org/license/mit. (Accessed 16 December 2025).
Nadal-Rey, Gisela, McClure, Dale D., Kavanagh, John M., Cassells, Benny, Cornelissen, Sjef, Fletcher, David F., Gernaey, Krist V., 2021. Development of dynamic compartment models for industrial aerobic fed-batch fermentation processes. Chem. Eng. J. 420 (P3), 130402. http://dx.doi.org/10.1016/j.cej.2021.130402.
Öner, Merve, Stocks, Stuart M., Abildskov, Jens, Sin, Gürkan, 2019. Scale-up Modeling of a Pharmaceutical Antisolvent Crystallization via a Hybrid Method of Computational Fluid Dynamics and Compartmental Modeling. In: Computer Aided Chemical Engineering, vol. 46, Elsevier Masson SAS, pp. 709–714. http://dx.doi.org/10.1016/B978-0-12-818634-3.50119-3.
Ramin, Elham, Cardillo, Antonio Gaetano, Liebers, Reinhard, Schmölder, Johannes, von Lieres, Eric, Van Molle, Wim, Niebel, Bastian, Natalis, Laurent, Meln, Irina, Perea-Vélez, Mónica, Clénet, Didier, Jørgensen, John Bagterp, Nilsson, Bernt, Bracewell, Daniel G., Gernaey, Krist V., 2024. Accelerating vaccine manufacturing development through model-based approaches: current advances and future opportunities. Curr. Opin. Chem. Eng. 43, 100998. http://dx.doi.org/10.1016/j.coche.2023.100998.
Shaikh, Bilal, Marupilla, Gnaneswara, Wilson, Mike, Blinov, Michael L., Moraru, Ion I., Karr, Jonathan R., 2021. RunBioSimulations: An extensible web application that simulates a wide range of computational modeling frameworks, algorithms, and formats. Nucleic Acids Res. 49, W597–W602. http://dx.doi.org/10.1093/nar/gkab411.
Sonnleitner, B., 1985. Growth of Saccharomyces cerevisiae Is Controlled by Its Limited Respiratory Capacity : Formulation and Verification of a Hypothesis *. Biotechnol. Bioeng. 28, 927–937.
Supabase Inc., 2025a. REST API. [Online]. Available: https://supabase.com/docs/guides/api. (Accessed 16 December 2025).
Supabase Inc., 2025b. Auto-generated Docs. [Online]. Available: https://supabase.com/docs/guides/api/rest/auto-generated-docs. (Accessed 16 December 2025).
Supabase Inc., 2025c. Supabase Platform. [Online]. Available: https://github.com/supabase/supabase. (Accessed 16 December 2025).
Supabase Inc., 2025d. Supabase Pricing. [Online]. Available: https://github.com/supabase/supabase. (Accessed 16 December 2025).
Tajsoleiman, Tannaz, 2018. Automating experimentation in miniaturized reactors (PhD thesis). Technical University of Denmark.
Tajsoleiman, Tannaz, Spann, Robert, Bach, Christian, Gernaey, Krist V., Huusom, Jakob Kjøbsted, Krühne, Ulrich, 2019. A CFD based automatic method for compartment model development. Comput. Chem. Eng. 123, 236–245. http://dx.doi.org/10.1016/j.compchemeng.2018.12.015.
Wang, Guan, Haringa, Cees, Noorman, Henk, Chu, Ju, Zhuang, Yingping, 2020. Developing a Computational Framework To Advance Bioprocess Scale-Up. Trends Biotechnol. 38 (8), 846–856. http://dx.doi.org/10.1016/j.tibtech.2020.01.009.
Wolfram, Gauss-Seidel Method. [Online]. Available: https://mathworld.wolfram.com/Gauss-SeidelMethod.html. (Accessed 16 December 2025).