Computers and Chemical Engineering 192 (2025) 108913

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

CADET-Julia: Efficient and versatile, open-source simulator for batch chromatography in Julia

Jesper Frandsen [a], Jan Michael Breuer [b], Johannes Schmölder [b], Jakob Kjøbsted Huusom [a], Krist V. Gernaey [a], Jens Abildskov [a], Eric von Lieres [b,c,*]

[a] Technical University of Denmark, Department of Chemical and Biochemical Engineering, PROSYS, Kongens Lyngby, Denmark
[b] Forschungszentrum Jülich, IBG-1: Biotechnology, Jülich, Germany
[c] RWTH Aachen University, Computational Systems Biotechnology, Aachen, Germany

ARTICLE INFO

Keywords:
Column liquid chromatography
General rate model
Lumped rate model with pores
Lumped rate model without pores
Discontinuous Galerkin spectral element method
CADET

ABSTRACT

This study introduces CADET-Julia, an open-source, versatile and fast chromatography solver implemented in the Julia programming language. The software offers a platform for rapid prototyping and numerical refinement for a range of chromatography models, including the general rate model (GRM). The interstitial column mass balance was spatially discretized using a strong-form discontinuous Galerkin spectral element method (DGSEM) whereas a generalized spatial Galerkin spectral method (GSM) was applied for the particle mass balance. Three different benchmarks showcased the computational efficiency of CADET-Julia: A baseline benchmark was established by comparing the Julia implementation to a C++ implementation that employed the same mathematical methods and time integrator (CADET-DG). Various Julia time integrators were tested, and with the best-performing settings, the Julia implementation was benchmarked against CADET-DG and a finite volume (FV) based implementation in C++ (CADET-FV). Overall, Julia implementations performed better than C++ implementations and Galerkin methods were generally superior to finite volumes.

1. Introduction

Chromatography is an essential unit operation in many industries, for example, the biopharmaceutical industry, where several chromatography steps are usually integrated in the downstream process (Kumar and Lenhoff, 2020; Schmidt-Traub et al., 2020; Zydney, 2016). In the purification of biopharmaceuticals, the role of chromatography in the downstream process is typically to capture, purify and polish the product before it is sent to formulation (Carta and Jungbauer, 2020). These chromatography steps are generally expensive and can account for up to 80% of the total manufacturing costs (Kozorog et al., 2023). Thus, optimizing the chromatography processes is crucial to ensure, e.g., high purity and yield, minimizing solvent consumption, enhancing high productivity and product titer, extending resin life-time, reducing manufacturing costs. Quantitative mathematical modeling can be essential for this optimization (Kumar and Lenhoff, 2020). Three transport models are customarily used to mathematically describe the solute transport through a chromatography column which, in ascending order of complexity, are the LRM, the lumped rate model with pores (LRMP) and the GRM. These models all describe transport of solutes using partial differential equations (PDE). Adsorption is mathematically described by isotherms or binding kinetics. The adsorption process is often assumed to be in rapid equilibrium, resulting in partial differential algebraic equations (PDAE) (Kumar and Lenhoff, 2020). These transport models, coupled with an isotherm or binding kinetic, can be used to optimize the chromatography process by formulating an objective function which is minimized subject to constraints. As closed-form solutions are not available for most chromatography models, the models must be solved numerically. During the optimization, the chromatography models must be solved a great number of times, depending on the number of optimization variables. Consequently, the efficiency of the optimization process is heavily reliant on the simulation speed and reliability of the numerical methods employed for solving the models.

To solve the PDE and PDAE, the method of lines is commonly used. That means the PDAE are discretized in space to yield a differential algebraic equation (DAE) system and the DAE are discretized in time to yield a purely algebraic system. These can be solved with publicly available ordinary differential equation (ODE) and DAE solvers, respectively. There are many methods to discretize in time. Various software packages have been developed to solve chromatography models (Andersson et al., 2023; Meyer et al., 2020; Berninger et al., 1991; Hahn et al., 2015; Leweke and von Lieres, 2018; Schmölder and

* Correspondence to: Forschungszentrum Jülich, IBG-1: Biotechnology, 52425 Jülich, Germany.
E-mail address: e.von.lieres@fz-juelich.de (E. von Lieres).

https://doi.org/10.1016/j.compchemeng.2024.108913
Received 11 July 2024; Received in revised form 30 September 2024; Accepted 30 October 2024
Available online 12 November 2024

Kaspereit, 2020; Zafar et al., 2023). The most powerful and commonly used open-source numerical solver for solving chromatography models is CADET-Core (Leweke and von Lieres, 2018), formerly simply known as CADET, which is now the name of the umbrella framework comprehending tools such as CADET-Process in addition to the numerical core solver(s). The original implementation started as a chromatography solver using a spatially weighted essentially non oscillatory (WENO) finite volume (FV) method (von Lieres and Andersson, 2010). Since then, CADET-Core was continuously expanded, enhancing both its numerical and modeling capabilities. The most significant advancements include algorithmic differentiation for computing the system Jacobian and parameter sensitivities (Püttmann et al., 2013), and the DGSEM discretization for chromatography models (Breuer et al., 2023). At the same time, the model family in CADET-Core was extended by including many adsorption models (e.g. HIC-isotherm (Jäpel and Buyel, 2022)), a 2D GRM, reactors, simulated moving beds (He et al., 2018), and population balance models for crystallization/precipitation (Zhang et al., 2024).

The CADET-Core code base is entirely developed in the programming language C++. Whereas C++ is generally recognized for its great computational performance, it is a low-level compiled language and thus demands more programming expertise compared to high-level programming languages such as Python. The abstract and extensive code base implemented in an object-oriented manner makes CADET-Core modular and extendable but also hard to comprehend for less experienced programmers. While the extensive of CADET-Core makes it very attractive for users, it requires a lot of programming and inside knowledge on the developer side to implement new models. For complex chromatography modes, such as mixed-mode chromatography, where the knowledge base and selection of isotherm models are limited (Kumar and Lenhoff, 2020), discovering and implementing new isotherm models is frequently required. The new isotherm models could either be mechanistic models or machine learning models (Nogueira et al., 2022; Santana et al., 2023). As the code-base is written in C++, it must be compiled before it can be used. Typically, a Python wrapper is used to run the CADET-Core code base. Furthermore, if a new model is implemented in CADET-Core, the entire C++ code base must be recompiled, which is not required in higher programming languages such as Python. Contrary to C++, Python is a dynamic high-level programming language that is easy to use and thus requires less programming expertise by the user. However, the ease of using Python comes at the expense of lower computational performance. Meanwhile, the programming language Julia leverages the advantages of C++ and Python as it is a dynamic, high-level programming language that simultaneously generates fast, low-level machine code (Bezanson et al., 2017). Upon the initial run of code, Julia automatically compiles low-level machine code which is then reused for subsequent executions, thus offering high computational performance. Additionally, Julia has a broad range of ODE solvers for stiff problems (Rackauckas and Nie, 2017), which are often encountered in chromatography modeling (Kumar and Lenhoff, 2020).

To address the above mentioned challenges, we introduce CADET-Julia, a more streamlined but limited code base. It retains fewer functionalities and a smaller model family but substantially simplifies the model implementation process. With Julia's high-level syntax and self-compiling capabilities, users can implement new models with limited programming expertise. Despite the simplified model implementation, CADET-Julia cannot compromise on computational performance, as it must ensure efficiency in simulation and optimization tasks. Therefore, it is important to implement a reliable and fast discretization method in CADET-Julia. Lately, fast higher order discretization methods have been implemented for chromatography models. Meyer et al. (2020) implemented the first arbitrary order DGSEM to discretize the column in the axial direction for the GRM, complemented by an arbitrary order GSM to discretize the particles. Breuer et al. (2023) added a collocation DGSEM to discretize the axial column direction, which proved to be slightly more efficient computationally. Their spatial discretization was complemented by a DGSEM particle discretization, which allows for targeted resolution of the particles (as opposed to GSM) and impenetrable particle cores. Both studies showed that the DGSEM was superior to the finite volume method in most cases. A DGSEM has also been successfully applied on the LRM for specific Langmuir isotherm variants considering non-isothermal conditions (Zafar et al., 2021; Khan et al., 2021) as well as reactions (Zafar et al., 2023). Due to the great accuracy of the DGSEM in solving chromatography models, CADET-Julia was based on the CADET-Core DGSEM implementation. From here on, we refer to the FV and DGSEM implementations in CADET-Core as CADET-FV and CADET-DG.

In light of these developments, this study aims to perform extensive benchmarks of CADET-Julia against the C++ implementations CADET-FV and CADET-DG in terms of convergence of maximum absolute error (MAE), degrees of freedom (DoF) and simulation time for various batch case studies. The following sections will detail the methodology and results of these benchmarks, starting with an analysis of spatial discretization variants, i.e. GSM vs DGSEM for particle discretization. Next, we conduct a baseline benchmark to isolate the performance differences between the programming languages C++ and Julia as closely as possible. Subsequently, various Julia ODE solvers and a DAE solver will be benchmarked for various chromatography models and settings to identify the fastest time integrator. Finally, an ultimate performance benchmark is conducted, in which the fastest settings from CADET-Julia will be directly compared to CADET-FV and CADET-DG. The implications of these performance differences will be discussed.

2. Chromatography models

We consider chromatography models to be modular w.r.t. transport and adsorption (binding), i.e. any transport model can be combined with any adsorption model. In this work, we consider the three most commonly used transport models in chromatography, namely the GRM, LRMP, LRM. As these models have been reported many times (Guiochon et al., 2006; Gu, 2015; Schmidt-Traub et al., 2020), we focus on a concise description of the GRM. In the interstitial column volume, the GRM accounts for convection and dispersion of the bulk concentrations c_i^b in axial direction $z \in (0, L)$ as well as film diffusion into the particles, with particle liquid concentration c_i^p, for each component $i \in \{1, \dots, N_c\}$. These mass transfer effects are governed by the equation

$$\frac{\partial c_i^b}{\partial t} = -u\frac{\partial c_i^b}{\partial z} + D_{ax,i}\frac{\partial^2 c_i^b}{\partial z^2} + \frac{(1-\varepsilon_c)}{\varepsilon_c} k_{f,i}\frac{3}{R_p}(c_i^b - c_i^p|_{r=R_p}) \tag{1}$$

in $(0, T_{\text{end}}) \times (0, L)$ with Danckwerts boundary conditions

$$uc_{\text{in},i} = \left(uc_i^b - D_{\text{ax},i}\frac{\partial c_i^b}{\partial z}\right)\bigg|_{z=0} \qquad \text{on } (0, T_{\text{end}}), \tag{2a}$$

$$0 = -D_{\text{ax},i}\frac{\partial c_i^b}{\partial z}\bigg|_{z=L} \qquad \text{on } (0, T_{\text{end}}), \tag{2b}$$

where $t \in [0, T_{end}]$ is time, T_{end} is the end time, L is the length of the column. The film mass transfer coefficient is denoted by $k_{f,i}$, D_{ax} is the axial dispersion coefficient and $c_{\text{in},i}$ denotes the column inlet concentrations.

In the particles, diffusion–reaction equations hold in $(0, T_{\text{end}}) \times (0, L) \times (R_c, R_p)$:

$$\frac{\partial c_i^p}{\partial t} + \frac{(1-\varepsilon_p)}{\varepsilon_p}\frac{\partial c_i^s}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 D_{p,i}\frac{\partial c_i^p}{\partial r}\right) + \frac{(1-\varepsilon_p)}{\varepsilon_p}\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 D_{s,i}\frac{\partial c_i^s}{\partial r}\right), \tag{3}$$

$$0 = f_{\text{bind}}(c_0^p, \dots, c_{N_c}^p, c_0^s, \dots, c_{N_c}^s). \tag{4a}$$

Here, c_i^s is the stationary particle phase concentration of component i, $r \in [R_c, R_p]$ is the radial particle coordinate, $R_p > R_c$ is the particle radius and $R_c \geq 0$ is the impenetrable particle core radius, ε_p is the particle porosity, D_p is the pore diffusion coefficient, and f_{bind} is defined by the adsorption model. We note that Eq. (4a) is used for rapid equilibrium adsorption, i.e. when adsorption happens practically infinitely fast. This requires that the problem is discretized and solved as a system of DAEs. If the rate of the adsorption process is limited by kinetics, the binding is modeled by the following ODE instead

$$\frac{\partial c_i^s}{\partial t} = f_{\mathrm{bind}}(c_0^p, \dots, c_{N_c}^p, c_0^s, \dots, c_{N_c}^s) \tag{4b}$$

In this work, we considered the Langmuir isotherm, the SMA isotherm and the linear isotherm. For description and mathematical formulation of the isotherms, we refer to existing literature, e.g. Schmidt-Traub et al. (2020). Rapid equilibrium can be approximated by multiplying f_{bind} by a large value, e.g. 10^8, which was used in this work. This allowed the system to be solved as a system of ODEs instead of a system of DAEs which generally is less challenging numerically (see e.g. Hairer and Wanner (1996)).

The boundary conditions for Eq. (3) are given on $(0, T_{\mathrm{end}}) \times (0, L)$ by

$$\left(\varepsilon_p D_{p,i} \frac{\partial c_i^p}{\partial r} + (1-\varepsilon_p) D_{s,i} \frac{\partial c_i^s}{\partial r} \right) \bigg|_{r=R_p} = k_{f,i} \left(c_i^b - c_i^p\big|_{r=R_p} \right), \tag{5a}$$

$$-\left(\varepsilon_p D_{p,i} \frac{\partial c_i^p}{\partial r} + (1-\varepsilon_p) D_{s,i} \frac{\partial c_i^s}{\partial r} \right) \bigg|_{r=R_c} = 0. \tag{5b}$$

3. Numerical methods

To discretize the space–time domain of the PDAE system given in Section 2, we apply the so-called method of lines. That is, we separately discretize the equations in space and time, where we partially implement our own method and draw on available methods, respectively.

3.1. Spatial discretization

The spatial DGSEM discretization of the bulk Eq. (1) used in this work is similar to the one in Breuer et al. (2023). The general DGSEM framework is detailed in Kronbichler (2021) and Winters et al. (2021) and the book of Hesthaven and Warburton (2008). As there is sufficient literature on the method and its application in chromatography, we only provide concise definitions essential for seamless comprehension of this work. We provide a more detailed derivation solely for our modified particle discretization, which extends the GSM derived by Meyer et al. (2020) with impenetrable particle cores. The GSM uses a global polynomial approximation with weakly enforced boundary conditions. It can thus be thought of as a single element DGSEM. The advantages and disadvantages of these discretizations are discussed in the corresponding benchmark comparison 4.1. In the following, we generalize the GSM derived by Meyer et al. (2020) to allow for impenetrable particle-cores i.e. $R_c \geq 0$.

The GSM discretization is based on a polynomial approximation of the solution variable $c : \Omega^r \mapsto \mathbb{R}$, which is computed on a computational reference element $E = [-1, 1]$. The affine mapping from the reference element to the spatial particle domain $\Omega^r = (R_c, R_p)$ is given by

$$r(\xi) = R_c + \frac{\Delta r}{2}(\xi + 1), \quad \xi \in [-1, 1], \tag{6}$$

with $\Delta r = R_p - R_c$ and the Jacobian of the inverse mapping

$$J^{-1} = \frac{\mathrm{d}\xi}{\mathrm{d}r} = \frac{2}{\Delta r}. \tag{7}$$

We approximate the solution via nodal (Lagrange) polynomial interpolation of arbitrary degree N_d on $N_n = N_d{+}1$ Lagrange–Gauß–Lobatto (LGL) points

$$c_h(\xi) = \sum_{j=0}^{N_d} \ell_j(\xi) \underline{c}_j \approx c(\xi), \tag{8}$$

with $\underline{c}_j := c(\mathbf{z}(\xi_j)) \in \mathbb{R}$ and $\{\xi_j\}_{j=0}^{N_d}$ being the LGL points, and $\{\ell_j\}_{j=0}^{N_d}$ being the Lagrange polynomials

$$\ell_j : E \to \mathbb{R}, \quad \xi \mapsto \prod_{\substack{i=0 \\ i \neq j}}^{N_d} \frac{\xi - \xi_i}{\xi_j - \xi_i} \quad \forall j \in \{0, \dots, N_d\}. \tag{9}$$

Discrete equivalents of (partial) integration and differentiation of such polynomials are computed using the following operators

$$\mathcal{D}_{i,j} := \frac{\partial \ell_j}{\partial \xi}(\xi_i), \tag{10a}$$

$$\mathcal{M}_{i,j}^{(\alpha,\beta)} := \int_E \ell_i(\xi)\ell_j(\xi)(1+\xi)^{\alpha}(1-\xi)^{\beta}\,\mathrm{d}\xi, \tag{10b}$$

$$\mathcal{L}_{i,0} := \oint_{\partial E} \ell_i(\xi)\ell_0(\xi)\boldsymbol{n}\,\mathrm{d}\xi, \quad \mathcal{L}_{i,1} := \oint_{\partial E} \ell_i(\xi)\ell_{N_d}(\xi)\boldsymbol{n}\,\mathrm{d}\xi, \tag{10c}$$

with $\mathcal{D} \in \mathbb{R}^{(N_n \times N_n)}$, $\mathcal{M}^{(\alpha,\beta)} \in \mathbb{R}^{(N_n \times N_n)}$, $\mathcal{L} \in \mathbb{R}^{(N_n \times 2)}$ defining the so-called differentiation matrix, mass matrix and lifting matrix, respectively.

Considering a spherical core–shell $\Omega^r := (R_c, R_p)$, we multiply Eq. (3) by a smooth test function $\phi \in C^{\infty}(\Omega^r)$ and integrate over Ω^r in spherical coordinates

$$\int_{R_c}^{R_p} \phi \left(\frac{\partial c^p}{\partial t} + \frac{1}{\beta_p}\frac{\partial c^s}{\partial t} \right) r^2\,\mathrm{d}r = \int_{R_c}^{R_p} \phi \frac{\partial}{\partial r}\left(r^2 \left(D_p \frac{\partial c^p}{\partial r} + \frac{1}{\beta_p} D_s \frac{\partial c^s}{\partial r} \right)\right) \mathrm{d}r. \tag{11}$$

We transform the equation to the computational reference element using the affine mapping (6) and get

$$\int_{-1}^{1} \phi \left(\frac{\partial c^p}{\partial t} + \frac{1}{\beta_p}\frac{\partial c^s}{\partial t} \right) r^2(\xi)\,\mathrm{d}\xi = \left(\frac{2}{\Delta r}\right)^2 \int_{-1}^{1} \phi \frac{\partial}{\partial \xi}\left(r^2(\xi) \left(D_p \frac{\partial c^p}{\partial \xi} + \frac{1}{\beta_p} D_s \frac{\partial c^s}{\partial \xi} \right)\right) \mathrm{d}\xi. \tag{12}$$

We approximate the sought solution variables using polynomial interpolation of degree $N_d^p \geq 1$

$$c^p(t,\xi) \approx \sum_{k=0}^{N_d^p} \underline{c}_k^s(t)\ell_k(\xi) =: c_h^p, \tag{13a}$$

$$c^s(t,\xi) \approx \sum_{k=0}^{N_d^p} \underline{c}_k^s(t)\ell_k(\xi) =: c_h^s. \tag{13b}$$

In order to obtain a discrete weak form of the equations, we reduce the test space to the finite dimensional polynomial space $\mathbb{P}^{N_d^p}([-1,1])$, and perform an integration by parts, which gives us for all Lagrange basis functions ℓ_i:

$$\int_E \left(\frac{\partial c_h^p}{\partial t} + \beta_p \frac{\partial c_h^s}{\partial t} \right) \ell_i r^2(\xi)\mathrm{d}\xi = \left(\frac{2}{\Delta r}\right)^2 \left[+ \oint_{\partial E} \frac{\partial}{\partial \xi}\left(D_p c_h^p + \frac{1}{\beta_p} D_s c_h^s \right) \times \ell_i r^2(\xi)\boldsymbol{n}\mathrm{d}s - \int_E \frac{\partial}{\partial \xi}\left(D_p c_h^p + \frac{1}{\beta_p} D_s c_h^s \right) \times \frac{\partial \ell_i}{\partial \xi} r^2(\xi)\mathrm{d}\xi \right].$$

We have to compute integrals of the form

$$\int_E \frac{\partial c_h^p}{\partial \xi}\frac{\partial \ell_i}{\partial \xi} r^2(\xi)\mathrm{d}\xi = \sum_{k=0}^{N_d^p} \underline{c}_k^s(t) \int_E \frac{\partial \ell_k(\xi)}{\partial \xi}\frac{\partial \ell_i}{\partial \xi} r^2(\xi)\mathrm{d}\xi,$$

with

$$r_k^2(\xi) = R_c^2 + R_c \Delta r(1+\xi) + \left(\frac{\Delta r}{2}\right)^2 (1+\xi)^2, \tag{14}$$

and hence collect the following expression into a discrete operator

$$A_{i,j}^{(a,b)} := \int_E \frac{\partial \ell_i}{\partial \xi}\frac{\partial \ell_j}{\partial \xi}(1-\xi)^{\alpha}(1+\xi)^{\beta}\,\mathrm{d}\xi$$

$$= \sum_{n=0}^{N_d^p} \sum_{m=0}^{N_d^p} \frac{\partial \ell_i}{\partial \xi}\Big|_{\xi_n} \frac{\partial \ell_j}{\partial \xi}\Big|_{\xi_m} \int_E \ell_n \ell_m (1-\xi)^a (1+\xi)^b \, \mathrm{d}\xi$$
$$= \left(D^T M^{a,b} D\right)_{i,j},$$

for which the entries are calculated exactly via a transformation of the nodal polynomial to the respective normalized Jacobi polynomial basis, i.e. with $\alpha = 0$ and $\beta \in \{0, 1, 2\}$, respectively (see Hesthaven and Warburton (2002)).

Defining the following discrete operators

$$\mathcal{A}^r := \mathcal{A}^{0,0} R_c^2 + \mathcal{A}^{0,1} R_c \Delta r + \mathcal{A}^{0,2} \left(\frac{\Delta r}{2}\right)^2, \tag{16a}$$

$$\mathcal{M}^r := \mathcal{M}^{0,0} R_c^2 + \mathcal{M}^{0,1} R_c \Delta r + \mathcal{M}^{0,2} \left(\frac{\Delta r}{2}\right)^2, \tag{16b}$$

we get the particle discretization as

$$\underline{\dot{c}}^p + \beta_p \underline{\dot{c}}^s = \left(\frac{2}{\Delta r}\right)^2 (\mathcal{M}^r)^{-1} \left[-(\mathcal{A}^r)\left(D_p \underline{c}^p + \frac{1}{\beta_p} D_s \underline{c}^s\right) + \mathcal{L} \begin{pmatrix} \frac{\partial}{\partial \xi}\left(D_p c_h^p + \frac{1}{\beta_p} D_s c_h^s\right) R_c^2 \\ \frac{\partial}{\partial \xi}\left(D_p c_h^p + \frac{1}{\beta_p} D_s c_h^s\right) R_p^2 \end{pmatrix} \right], \tag{17}$$

with $\underline{\dot{c}}^p, \underline{c}^p, \underline{c}^s$ being the vectors of the respective nodal polynomial coefficients.

Inserting the boundary conditions (mapped to the reference element), we get

$$\underline{\dot{c}}^p + \beta_p \underline{\dot{c}}^s = \left(\frac{2}{\Delta r}\right)^2 (\mathcal{M}^r)^{-1} \left[-(\mathcal{A}^r)\left(D_p \underline{c}^p + \frac{1}{\beta_p} D_s \underline{c}^s\right) + \mathcal{L} \frac{\Delta r}{2} \begin{pmatrix} 0 \\ \frac{1}{\epsilon_p} k_f \left(c^b - c^p\right) R_p^2 \end{pmatrix} \right]. \tag{18}$$

We note that this formulation is equivalent to the one in Meyer et al. (2020), if $R_c = 0$, i.e. without impenetrable particle cores.

The discretized weak formulation of the complementing (here rapid-equilibrium) binding Eq. (4) is given for all test functions ℓ_j of the Lagrange basis $\{\ell_i\}_{i=0}^{N_d^p}$ by

$$\int_E \dot{c}_h^p \ell_j(\xi) R_c^2(\xi) \frac{\Delta r}{2} \, \mathrm{d}\xi = \int_E \sum_{k=0}^{N_d^p} f_{\text{bind}}(c_h^p(\xi_k), c_h^s(\xi_k)) \ell_k(\xi) \ell_j(\xi) R_c^2(\xi) \frac{\Delta r}{2} \, \mathrm{d}\xi. \tag{19}$$

Here, $\dot{c}_h^p$ is the spatial polynomial approximation of the time derivative variable $\frac{\partial c^p}{\partial t}$. Applying the inverse mass matrix yields

$$\underline{\dot{c}^s} = f_{bind}(\underline{c}^b, \underline{c}^s). \tag{20a}$$

Note that for rapid equilibrium bindings, we get

$$0 = f_{bind}(\underline{c}^b, \underline{c}^s). \tag{20b}$$

3.2. Temporal discretization

To integrate the semi-discretized system in time, we make use of available, efficient and well-established methods. Here, we exploit the flexibility of CADET-Julia, to apply various time integration methods, while CADET-Core is fixed to using an adaptive time step, variable order backwards differentiation formula (BDF) method implemented in the IDA solver of the SUNDIALS software package (Hindmarsh et al., 2005; Gardner et al., 2022). Whereas this method has proven to be suitable for a general purpose simulation tool like CADET-Core, our goal is to find the best suited methods in CADET-Julia for the specific problems considered in this work.

Two different approaches of solving the rapid-equilibrium were implemented in CADET-Julia. First, the true rapid equilibrium Eq. (4a) was implemented, resulting in a DAE system. Here, the semidiscrete scheme is complemented by the IDA BDF method. In CADET-Julia, the IDA time integrator was used through the Sundials.jl package (Rackauckas and Nie, 2017), which implements an interface to the C++ implementation of the Sundials package, and uses a Sundials linear solver as the internal Newton method. In CADET-Core, IDA is complemented with a linear solver from the Eigen library (Guennebaud and Benoît, 2010).

Alternatively, the rapid equilibrium was approximated using Eq. (4b) multiplied with a large kinetic constant, resulting in an ODE system. For solving the ODE systems, Julia has many different ODE solvers available through the DifferentialEquations.jl package (Rackauckas and Nie, 2017). These native Julia solvers use the LinearSolve.jl library as their linear solver. Of the many different ODE solvers available in the DifferentialEquations.jl package, only the implicit stiff solvers were benchmarked as the chromatography models are generally stiff problems (Kumar and Lenhoff, 2020). A quick screening showed that the FBDF, QNDF and the QBDF solvers were by far the most useful. The FBDF is a fixed-leading coefficient adaptive-order with adaptive timestep BDF (Shampine, 2002). The QNDF is an adaptive order quasi-constant timestep numerical differentiation formula (NDF) method which uses Shampines accuracy-optimal kappa values (Shampine et al., 1997). The QBDF is an adaptive order quasi-constant timestep BDF method which is a special case of QNDF method where the kappa values are set to zero. These solvers are all up to 5th order implicit solvers. To benchmark these solvers for the case studies, a prototype Jacobian, generated using FD, was provided to the solver to set the sparsity pattern. Then, four different approaches for setting up the ODE solvers were benchmarked to identify the fastest solver for the studied models. For each solver, four modes for the Jacobian computation and factorization were tested: Either provide the analytical Jacobian to the solver or let it calculate the Jacobian via FD. For both modes, the solver was tested with and without the computation of a preconditioning for the Jacobian factorization. The analytical Jacobian was split into a static Jacobian, which could be predetermined, and a dynamic Jacobian which was updated every time the Jacobian was updated (Meyer et al., 2020). For the Julia ODE solvers, the linear solver was left empty, meaning a suitable linear solver was selected automatically for each individual problem (LinearSolve.jl, 2024).

3.3. Software and implementation

CADET-Julia encompasses a modular implementation of three transport models (GRM, LRMP, LRM) and three binding models (Linear, Langmuir, SMA), and their spatial discretization using two DGSEM variants (exact integration and collocation) for the bulk transport Eq. (1) and a GSM for the particle transport Eq. (3). The GSM was also implemented in CADET-Core.

CADET-Julia is published on GitHub as open-source software and can be found on https://github.com/cadet/CADET-Julia. CADET-Core is also publicly available on github under https://github.com/modsim/CADET.

In the following, we consider some implementation details of CADET-Julia, which potentially cause differences in computational performance compared to CADET-Core, that are unrelated to the programming language, even though the mathematical methods are similar. The general state vector representation for CADET-Core and CADET-Julia is given as

$$\begin{aligned} RHS =& [c_{0,0}^b, \dots, c_{\text{end},0}^b, c_{0,1}^b, \dots, c_{\text{end,end}}, \\ & c_{0,0,0}^p, \dots, c_{\text{end},0,0}^p, \dots, c_{\text{end,end},0}^p, \dots, c_{\text{end,end,end}}^p \\ & c_{0,0,0}^s, \dots, c_{\text{end},0,0}^s, \dots, c_{\text{end,end},0}^s, \dots, c_{\text{end,end,end}}^s]. \end{aligned} \tag{21}$$

The order of the state vector in CADET-Julia is axial position - radial (particle) position - component position - major, whereas in CADET-Core it is component - axial position - radial (particle) position - major. Thus in CADET-Julia, first all the components in the bulk phase in a component-wise manner are listed, then all the components in the particle phase in a component-wise manner are listed and finally all the components in the stationary phase in a component-wise manner are listed. With that, we enable easy matrix multiplication in component-wise manner which follows the derivation of the final form of the DGSEM discretization from Eq. (18). The implementation of DGSEM in CADET-Core had to follow another state vector order to match the interfaces, e.g. for binding models.

4. Results and discussion

In the following, we investigate case studies similar to the ones in Breuer et al. (2023), as they cover a wide range of linear and non-linear settings for all three transport models. The linear settings are particularly interesting for method validation, as we compare to analytical solutions. The SMA settings are most representative for real world applications, as they resemble classical load–wash–elute cases. The Langmuir setting is chosen to specifically challenge the numerical methods, as it causes steep concentration fronts, for which unstabilized high order methods typically struggle (Breuer et al., 2023). All problems were solved using relative and absolute time integrator tolerances of $1 \cdot 10^{-10}$ and $1 \cdot 10^{-12}$, respectively. To reproduce the results, the evaluation scripts have been made publicly available (Frandsen et al., 2024). The CADET-Core version 4.4 from git commit 1572ca6 and the CADET-Julia git commit a49a9ac were used for the benchmarks. For each simulation setting, triplicates were run and the fastest runtime was kept. All simulations were conducted on a Dell Latitude 7310 with an Intel(R)-Core TM-i5-10310U 1.70 GHz processor and 16 GB RAM. For the all the benchmarks in this chapter, we consider the MAE at the column outlet. To this end, we computed high resolution numerical reference solutions using CADET-Core, which has been rigorously verified by experimental order of convergence (EOC) tests e.g. using analytical solutions for linear models and CADET-DG and CADET-FV mutually verifying each other for non linear-models. Exceptionally, for the LRM with linear isotherm, the MAE was evaluated by comparing to an analytical solution (Javeed et al., 2013). As Breuer et al. (2023) showed that the case studies were solved most efficiently using a minimum of fourth order polynomials for the bulk phase, all benchmarks were conducted with a minimum of fourth order polynomials for the bulk phase discretization. Furthermore, Breuer et al. (2023) also demonstrated that, in general, using collocation DGSEM was slightly more efficient than exact integration DGSEM. Therefore, only collocation DGSEM was used for the benchmarks.

4.1. GSM vs. DGSEM for particle discretization

In this subsection, we investigate the performance of the GSM and DGSEM particle discretizations in CADET-DG. As mentioned before, the GSM can be thought of as a single element DGSEM. The DGSEM, however, requires the computation of an additional auxiliary equation to incorporate the second order derivative into the DGSEM framework (Bassi and Rebay, 1997). Accordingly, the GSM requires less arithmetic operations to compute a polynomial approximation of similar accuracy compared to a single element DGSEM and is, in that case, computationally more efficient. This is confirmed in benchmark 1 for a GRM with the SMA isotherm. For MAE down to $7 \cdot 10^{-4}$ mol/m^3, the GSM discretization is the fastest. For more precise solutions, the DGSEM discretization using two polynomials in the pore phase is faster. For the linear and Langmuir isotherm case studies, the GSM discretization was the fastest which is shown in Supplementary Material Figures S1–S2. The DGSEM approach of Breuer et al. (2023) on the other hand allows for non-equidistant resolution of the particles with arbitrarily spaced

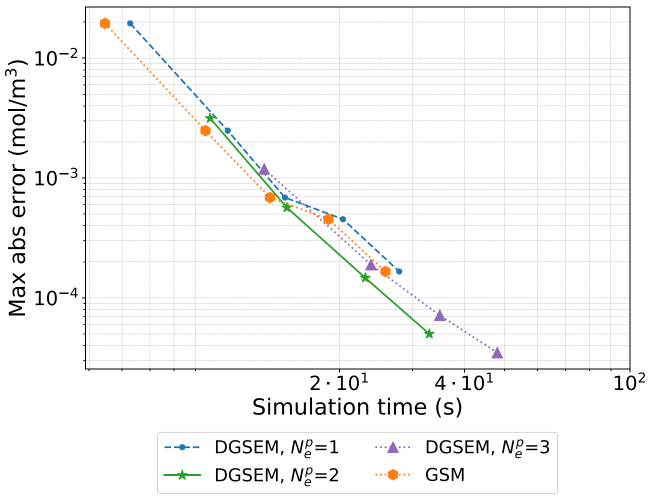

Fig. 1. Simulation time and maximum absolute error comparison between DGSEM and GSM for particle discretization for the GRM with the SMA isotherm case study with parameters found in Table S4. The bulk phase resolution was fixed such that the particle discretization error dominates. The number of particle elements for the DGSEM was fixed and the particle resolution was refined by increasing the polynomial degree. The tests were run in CADET-DG.

elements. The computational advantage of the DGSEM is expected to show if there is a specific part of the particles, e.g. near the boundary, that should be resolved more accurately, or if we have strong gradients in the particles, i.e. where the artificial dispersion via the numerical fluxes between multiple elements is beneficial to stabilize the scheme. For the settings considered in this work however, the GSM has shown to be the fastest approach down to engineering precision, i.e., down to MAE of $1 \cdot 10^{-3}$ mol/m^3 (Meyer et al., 2020), as we can see e.g. for the GRM SMA case in Fig. 1. For that reason, the GSM particle discretization was implemented in CADET-Julia.

4.2. Baseline benchmark

In this subsection, the performance difference of CADET-Julia and CADET-DG, that may arise due to the difference in code languages, is investigated. To this end, we have mitigated all potential sources of performance differences by using the same numerical methods. That is, we used the same Sundials IDA time integrator, which is implemented in C++ and called through a Julia interface. The Jacobians were determined analytically. To isolate the performance difference in code languages even further, no adsorption was considered for the LRM, LRMP and GRM case studies with parameters found in Supplementary Material S1 Table S1. This ensures that different adsorption implementations would not influence the performance benchmark. Hence, this baseline benchmark only considers convection and dispersion through a porous column. For the GRM, the tests were carried out using fourth order polynomials for the bulk phase at a varying number of DG elements in the bulk phase and varying particle phase polynomials. For the LRM and LRMP, the DG elements in the bulk and bulk phase polynomials were varied. To solve the algebraic systems within the ODE solver, CADET-Core uses a LU factorization (Guennebaud and Benoît, 2010), whereas a KLU factorization was used in CADET-Julia (Hindmarsh et al., 2005). Thereby, we remark that the linear solver used by the time integration methods were not the exact same. Additionally, CADET-Core utilizes Sundials version 3.2.1 (Leweke and von Lieres, 2018), whereas CADET-Julia calls Sundials version 5.2.0.

Fig. 2 depicts the convergence per DoF for a GRM and shows the same errors for CADET-DG and CADET-Julia, thereby confirming the implementation of the methods.

The DoF and simulation time for the GRM case study are shown in Fig. 3.

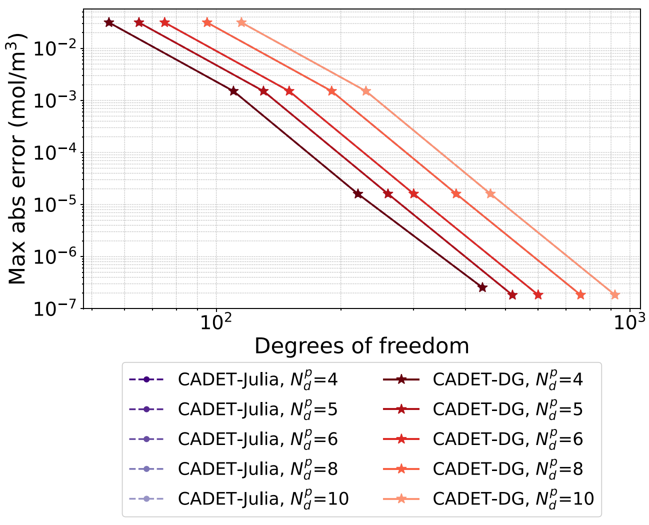

Fig. 2. The degrees of freedom and the maximum absolute error for CADET-Julia and CADET-DG using fourth order polynomials at various DG elements in the bulk phase for collocation DGSEM. The number of bulk phase elements was gradually increased (doubled, starting with 1 element). The sundials IDA time integrator was used. The case study is the GRM with parameters shown in Supplementary Material S1 Table S1. Note that the CADET-DG and CADET-Julia graphs are overlapping.

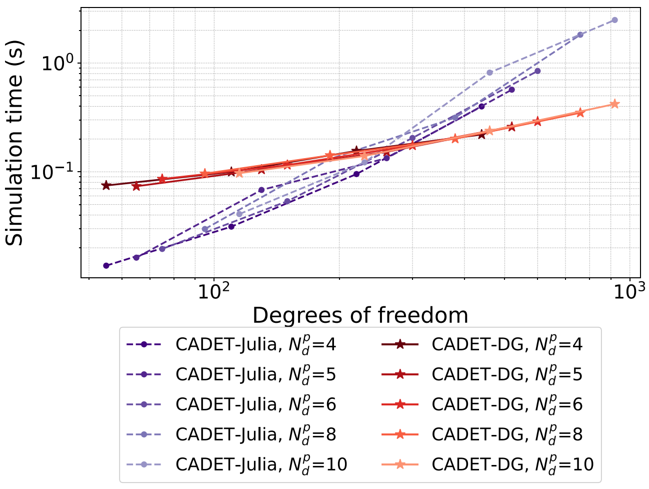

Fig. 3. The degrees of freedom and the simulation time for CADET-Julia and CADET-DG using fourth order polynomials at various DG elements in the bulk phase for collocation DGSEM. The number of bulk phase elements was gradually increased (doubled, starting with 1 element). The sundials IDA time integrator was used. The case study is the GRM with parameters shown in Supplementary Material S1 Table S1.

Fig. 3 shows that CADET-Julia is faster for a small number of DoFs. At increasing DoFs, CADET-DG scales significantly better than CADET-Julia. Using more than 300 DoF, CADET-DG becomes faster than CADET-Julia for this case study. The relatively poor scaling of CADET-Julia was also observed for the LRMP and LRM case studies which are shown in Supplementary Material S3, Figures S3–S8.

4.3. Time integrators CADET-Julia

In this subsection, different time integrators for ODE problems, as covered in method Section 3.2, are compared to find the fastest time integrator for the settings considered here, with implications on general chromatography problems. To this end, we study both, linear, highly non-linear and specifically stiff settings. The computational performance is investigated for multiple spatial resolutions, and time integration tolerances are chosen so that the spatial discretization error

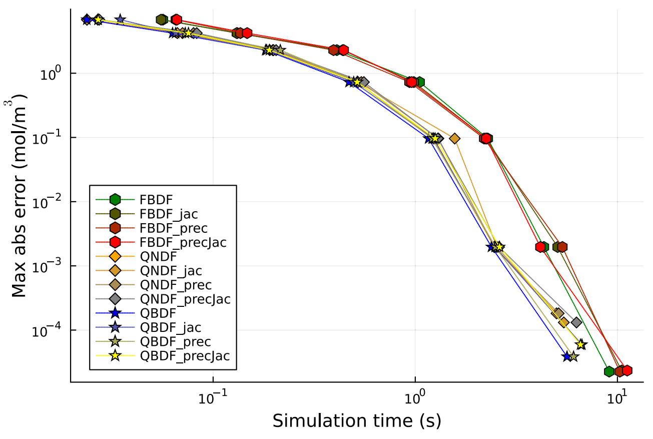

Fig. 4. The simulation time and the maximum absolute error for Julia ODE time integrators on a log–log plot evaluated at fourth order polynomials. The number of bulk phase elements was gradually increased (doubled, starting with 2 elements). The case study is the LRM with the Langmuir isotherm with parameters found in Supplementary Material S1 Table S3. For the FBDF, QNDF and QBDF, the Jacobian was determined using the FD for the different time integrators. For _jac, the Jacobian was determined analytically. For _prec, a preconditioning function was provided to the time integrators and for _precjac, a preconditioning function was provided to the time integrator and the Jacobian was determined analytically.

dominates such that the performance for a specific MAE is benchmarked. For each solver, we have tested four modes for the Jacobian computation and factorization: We either provide the analytical Jacobian to the solver or we let it calculate the Jacobian via FD. For both modes, we tested the solver with and without the computation of a preconditioning for the Jacobian factorization. As a note, the IDA DAE solver was excluded from time integrator benchmarks because of the poor scaling, see Fig. 3.

Throughout all the benchmarks, which are given in the Figs. 4–5 and in supplementary material section S4, Figures S9–S14, the FBDF solver performed worse than the QNDF and QBDF solvers. Generally, the QNDF solver exhibited the best performance which is also shown in Fig. 5 for the GRM with the Linear isotherm. For some case studies, the QBDF solver showed similar efficiency to the QNDF solver, usually for low discretizations and only for few settings. Moreover, all the solvers seemed to scale very well compared to for example the IDA solver, see Fig. 3. For the LRM with the Langmuir isotherm case study shown in Fig. 4, there was a difference in convergence between the ODE solvers. Here, the FBDF actually converged to a lower MAE compared to the QNDF and QBDF solvers.

Providing the analytical Jacobian did not reduce the simulation time significantly. This is unexpected as providing the analytical Jacobian usually decreases simulation time. However, for the QNDF and QBDF solvers, the Jacobian is only updated if the Newton iteration is converging slowly, meaning the Jacobian is rarely updated (Shampine et al., 1997). Likewise, preconditioning of the Jacobian was also expected to reduce simulation time. For these case studies, the default option for linear solver in Julia was specified, meaning the most suitable linear solver was automatically chosen based on each individual problem. When preconditioning did not reduce the simulation speed, this indicates that the linear solver, chosen by the solver algorithm, did not use the preconditioning. Since the difference between using the analytical Jacobian and determining the Jacobian using FD is insignificant, this suggests that determining the Jacobian is not contributing significantly to the overall simulation time for QNDF and QBDF ODE solvers. Additionally, providing the analytical Jacobian led to a lower MAE convergence in only one of the case studies. Furthermore, not being required to derive the analytical Jacobian makes it simpler to implement new models. It can be very tedious or even impossible to derive the analytical Jacobian, especially when dealing with complex models.

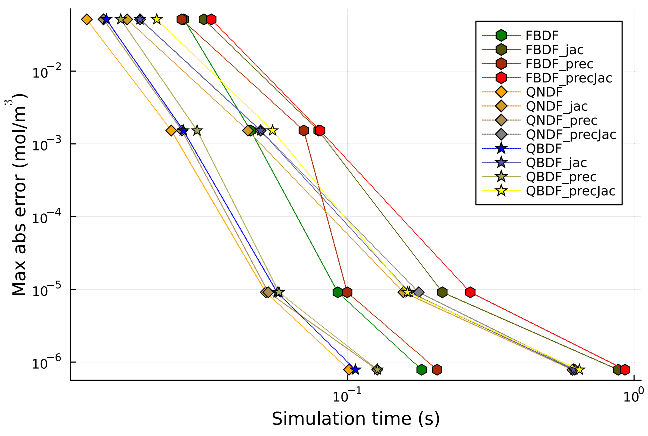

Fig. 5. The simulation time and the maximum absolute error for Julia ODE time integrators on a log–log plot evaluated at fourth order polynomials. The number of bulk phase elements was gradually increased (doubled, starting with 2 elements). The case study is the GRM with the Linear isotherm. For the FBDF, QNDF and QBDF, the Jacobian was determined using the FD for the different time integrators. For _jac, the Jacobian was determined analytically. For _prec, a preconditioning function was provided to the time integrators and for _precjac, a preconditioning function was provided to the time integrator and the Jacobian was determined analytically.

Then, not having to derive and implement the analytical Jacobian can speed up the implementation of new models. Thus, for the next subsection, the QNDF ODE solver is used and the Jacobian is determined using FD. For a single simulation, the difference in the magnitude of the simulation time is not significant. However, in applications requiring many iterations, such as optimization or Monte Carlo simulations, or in more complex chromatography setups involving multiple columns, the cumulative difference in simulation time based on the ODE solver becomes significant.

4.4. Ultimate performance benchmark

In this subsection, the most performant solver choices for CADET-Julia were benchmarked against CADET-FV and CADET-DG. Thus, in CADET-Julia, the models were solved as ODE problems by approximating the rapid equilibrium, Eq. (4b). The QNDF ODE solver was used with the Jacobian determined using FD. For the GRM benchmarks, fourth order polynomials in the bulk phase were used, and the number of DG elements in the bulk phase and the order of the particle phase polynomials were varied. For the LRM and the LRMP benchmarks, the number of DG elements in the bulk phase and the order of the bulk phase polynomials were varied. For the GRM with the SMA isotherm case study, the concentration profiles at different bulk phase DG elements are shown in Fig. 6.

Fig. 6 shows the concentration profiles of the proteins at the outlet using one and four DG elements for the bulk phase. Using one DG element is insufficient as the concentration profile displays non-physical behavior by simulating negative concentrations. Such non-physical behavior disappears when increasing the number of DG elements as seen when using four DG elements. The Simulation time and MAE for CADET-Julia, CADET-DG and CADET-FV are shown in Fig. 7.

Fig. 7 shows that CADET-Julia performs slightly better than CADET-DG and significantly better than CADET-FV. As expected, as the spatial methods are the same and dominate the error, we observe the same convergence behavior for CADET-DG and CADET-Julia, however, CADET-Julia is a factor two times faster for this case study.

For the LRM with the Langmuir isotherm case study, the simulated concentration profiles are shown in Fig. 8.

Fig. 8 shows that using too few DG elements, the solution oscillates and even yields negative concentrations. This is especially present at two DG elements. Increasing the number of DG elements decreases the

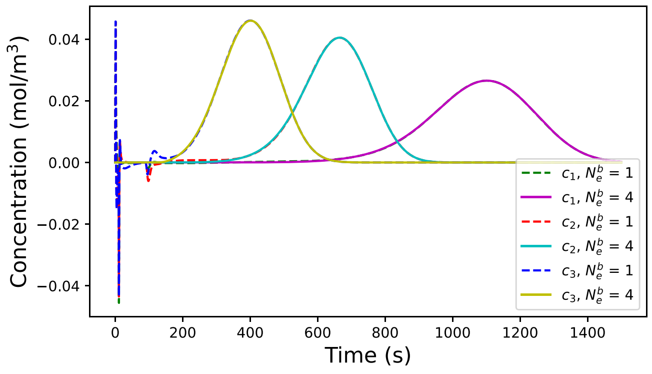

Fig. 6. Outlet concentration profiles for component 1,2 and 3 using one and four DG elements and fourth order polynomials for the bulk and particle phase. The case study is the GRM with the SMA isotherm with parameters found in Supplementary Material S1 Table S4.

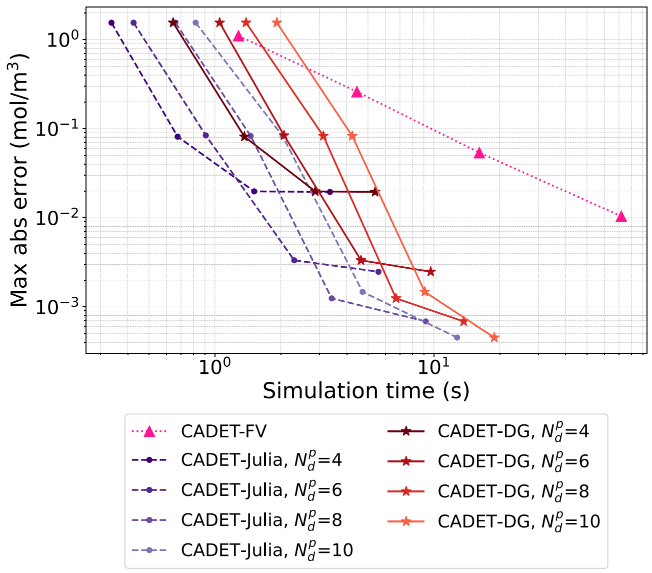

Fig. 7. Simulation time and maximum absolute error comparison between CADET-FV, CADET-Julia and CADET-DG evaluated for various particle phase polynomial degrees, N_d^p, and fourth order bulk phase polynomials. The number of bulk phase elements was gradually increased (doubled, starting with 1 element). The case study is GRM with the SMA isotherm case study with parameters found in Supplementary Material S1 Table S4.

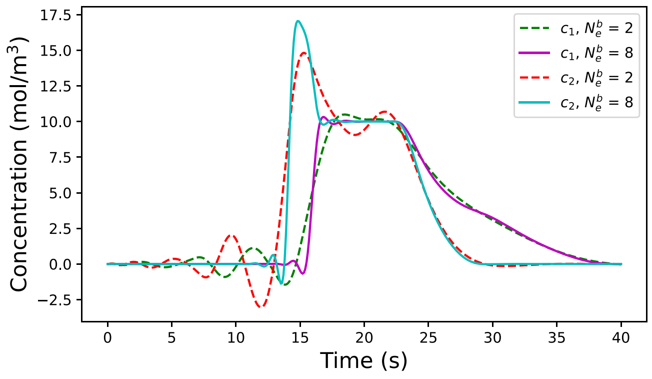

Fig. 8. Outlet concentration profiles for component 1 and 2 using two and eight DG elements and fourth order polynomials for the bulk and particle phase. The case study is the LRM with the Langmuir isotherm with parameters found in Supplementary Material S1 Table S3.

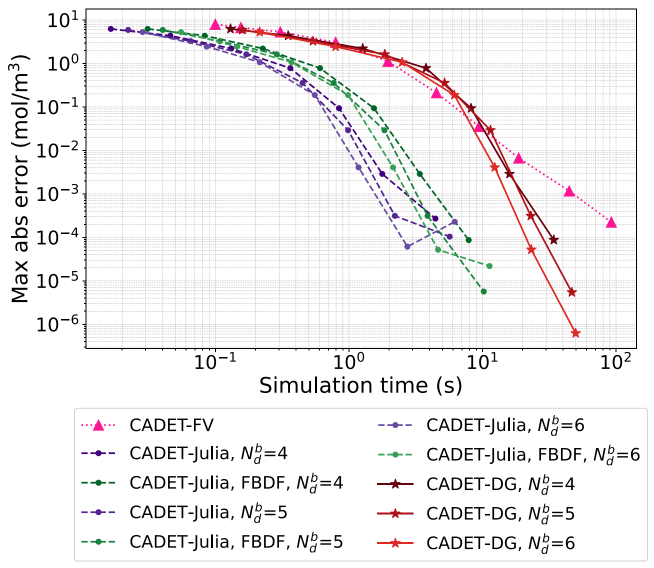

Fig. 9. Simulation time and maximum absolute error comparison between CADET-FV, CADET-Julia and CADET-DG evaluated for various particle phase polynomial degrees, N_d^p, and fourth order bulk phase polynomials. The number of bulk phase elements was gradually increased (doubled, starting with 2 elements). The case study is LRM with the Langmuir isotherm case study with parameters found in Supplementary Material S1 Table S3.

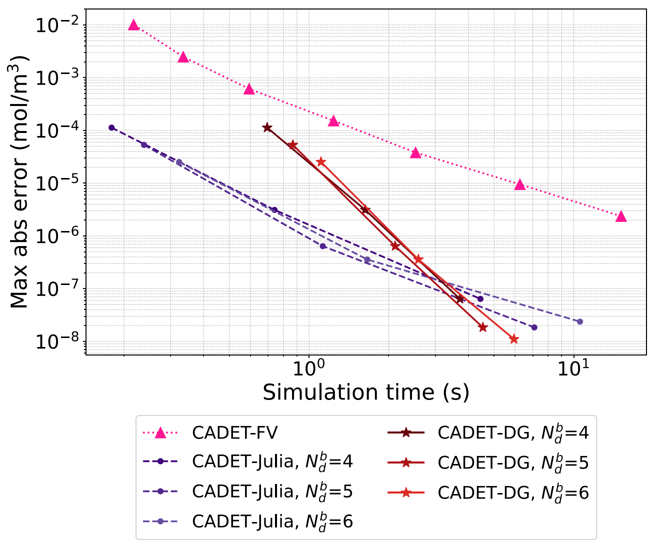

Fig. 10. Simulation time and maximum absolute error comparison between CADET-FV, CADET-Julia and CADET-DG evaluated for various particle phase polynomial degrees, N_d^p, and fourth order bulk phase polynomials. The number of bulk phase elements was gradually increased (doubled, starting with 2 elements). The case study is LRM with the SMA isotherm case study with parameters found in Supplementary Material S1 Table S4.

oscillations and reduces the magnitude of the negative concentrations, however, eight DG elements are still not sufficient to get rid of non-physical behavior. Though non-physical, this is in accordance with previous observations for the same case study (Breuer et al., 2023). The MAE and the runtime for different bulk phase polynomials and DG elements are shown in Fig. 9.

Fig. 9 shows that CADET-Julia is faster than both CADET-DG and CADET-FV, using both the default (QNDF) and FBDF ODE solver. However, at MAEs lower than $6 \cdot 10^{-4}$ mol/m^3, CADET-Julia stops converging and the MAE stops decreasing when using the QNDF solver. For that reason, the FBDF solver was also used for this case study as Fig. 4 showed that FBDF converged to lower MAEs. When using the FBDF solver, the MAE follows the same as for CADET-DG except for the last data point at $N_d^b = 6$. The reason why the last data point does not match the MAE is not clear. Likewise, it is not clear why the QNDF stops converging at MAEs lower than $6 \cdot 10^{-4}$. However, the convergence struggles for this case study are caused by the time integrator. Also, solving the models down to a MAE less than $1 \cdot 10^{-3}$ mol/m^3 is sufficient for most engineering purposes. CADET-Julia also struggled to solve the LRM with the SMA isotherm case study efficiently. Using the FBDF, QNDF or QBDF ODE solvers resulted in severe non-physical oscillations of the solution. Manually changing the linear solver to different options also did not remove the oscillations. As suggested by Shampine et al. (1997), reducing the maximum order of approximation of the ODE solver to increase stability also did not help on the stability issues for this case study. Instead, only the Sundials CVODE-BDF solver from the Sundials.jl library was able to solve the problem without non-physical oscillations. The MAE and the simulation time for the LRM with the SMA isotherm case study are shown in Fig. 10.

Fig. 10 shows that CADET-Julia does not scale well compared to CADET-DG. As for the Baseline benchmark study in Section 4.2, where the poor scaling of the Sundials IDA DAE solver in Julia was demonstrated, the Sundials CVODE-BDF also does not scale well for this case study. The reasons for the poor scaling are probably the same as for the Sundials IDA solver in Julia, as discussed in Section 4.2. That being said, CADET-Julia performs satisfactorily way below engineering precision of $1 \cdot 10^{-3}$ mol/m^3 for the MAE (Meyer et al., 2020).

The results of the remaining case studies can be found in Supplementary Material S5, Figures S15–S26. A summary of the performance differences for the different case studies is given in Table 1.

As seen in Table 1, CADET-Julia generally gives great advantage in terms of simulation time. The QNDF solver struggled with very stiff problems as for LRM with the Langmuir isotherm and the LRM with SMA isotherm case studies, but for the rest of the case studies, CADET-Julia gave the same convergence as CADET-DG at faster simulation times. At MAEs approaching $1 \cdot 10^{-8}$, the convergence stops for CADET-Julia i.e. the MAE does not decrease further at increasing DG elements beyond that error. The reason for this is that the rapid equilibrium is approximated by Eq. (4b), i.e. by multiplying the whole isotherm expression with a constant of $1 \cdot 10^8$, which becomes the limiting factor. This behavior can be observed for some of the case studies, see Supplementary Material S5. Furthermore, CADET-Julia offers enhanced flexibility in terms of implementing new models. The ease of implementation is particularly evident as it is not required to provide any information about the Jacobian. Not being required to derive the analytical Jacobian, simplifies implementing new models. Furthermore, the limited code base of CADET-Julia (compared to CADET-Core) comes with a reduced compilation time which eases the implementation and troubleshooting of new models, thereby enabling rapid prototyping.

To evaluate why CADET-Julia is generally faster than CADET-DG, the differences between the two implementations can be pinpointed.

1. CADET-Julia solves an ODE problem whereas CADET-DG solves a DAE problem.
2. CADET-Julia uses the QNDF ODE solver while CADET-DG uses the Sundials IDA solver.
3. CADET-Julia and CADET-DG have a different order of state vector, see Section 3.3.
4. CADET-DG is a large system solver with numerous functionalities whereas CADET-Julia has fewer functionalities.
5. CADET-Julia is written in the programming language Julia whereas CADET-DG is written in the programming language C++.
6. CADET-Julia automatically chooses a linear solver whereas CADET-DG uses LU-factorization.

Table 1
Summary of CADET-Julia performance. The Avg. speed up is the average speed up of CADET-Julia compared to CADET-DG, computed from all spatial resolutions for each setting.

Transport model	Isotherm model	Avg. Speed up	Comment
LRM	Linear	9.4	Good convergence, fast simulation time.
	Langmuir	9.6	CADET-Julia struggled to converge to MAE below $6 \cdot 10^{-4}$ mol/m^3, see Fig. 9
	SMA	2.0	Native Julia ODE solvers yielded oscillatory solutions. CVODE_BDF was used, see Fig. 10
LRMP	Linear	2.4	Good convergence, fast simulation time.
	Langmuir	1.9	Good convergence, fast simulation time.
	SMA	2.8	Good convergence, fast simulation time.
GRM	Linear	3.5	Good convergence, fast simulation time.
	Langmuir	2.2	Good convergence, fast simulation time.
	SMA	1.8	Good convergence, fast simulation time.

7. CADET-Julia has a slightly different SMA implementation compared to CADET-DG.

These differences between the two implementations contribute to why CADET-Julia is faster than CADET-DG. Solving a DAE compared to an ODE system is generally more challenging numerically (see e.g. Hairer and Wanner (1996)), which often shows in the respectively required computational effort. Moreover, the order of the state vector (see Eq. (21)) influences both the computation of the residual and the sparsity pattern of the Jacobian. In CADET-Julia, the DGSEM operations in the residual are expected to be faster as the required data are stored continuously in the state vector. Additionally, the CADET-Core Jacobian implementation contains more fill in zeros, resulting in a wider non-zero pattern around the main diagonal of the Jacobian which influences the performance of the factorization. Furthermore, the QNDF solver is very efficient and has demonstrated to be more efficient than the Sundials CVODE-BDF solver (SciML, 2021). For large arrays of equations, a general pattern of preference for the QNDF solver over the Sundials CVODE-BDF solver was observed (SciML, 2021), which is confirmed in this study. Thus, the difference in time integrators between CADET-Julia and CADET-DG is an important factor that can explain the large performance difference between both software packages. Furthermore, the more streamlined code base of CADET-Julia also contributes significantly to a speed up as the many functionalities in CADET-DG may result in overhead on the simulation time. Moreover, CADET-Julia automatically chooses a linear solver for the ODE problem whereas CADET-DG uses a fixed LU-factorization method as linear solver. This difference in linear solvers could also contribute to the performance difference. In terms of the performance differences between programming languages C++ and Julia, some studies suggest that Julia might be slightly faster for some operations like matrix multiplication but not fast enough to explain the relatively large differences between CADET-DG and CADET-Julia in simulation time (Eschle et al., 2023). Regarding the different SMA implementations, the speed ups for SMA case studies align with the general speed up trend, meaning the differences in the SMA implementation is probably not contributing significantly to the differences in the speed up.

To further improve the performance of the CADET-Julia code base, extensive profiling of the code to identify and optimize performance bottlenecks could be carried out. Moreover, different spectral numerical methods could be tested to find the most efficient one. To make the CADET-Julia more robust by default, an automatic, adaptive algorithm that switches the ODE solver could be implemented. For instance, in the LRM with the Langmuir and SMA isotherm case studies, the default solver (QNDF) did not perform well. Hence, in those cases, an adaptive algorithm that switches to a more stable ODE solver by default would enhance the robustness of the code base for solving chromatography models. Finally, the code base could be further developed to support parallelization which could improve the computational performance even further.

5. Conclusions

In this study, we have presented CADET-Julia, applied it to various case studies and benchmarked it against CADET-DG and CADET-FV. From these benchmarks, the following conclusions can be drawn.

1. The GSM for the particle phase was more efficient than DGSEM, hence this was implemented in CADET-Julia.
2. To reveal the performance difference due to different programming languages, the baseline simulations showed that CADET-Julia was faster for smaller systems but CADET-Julia scaled worse than CADET-DG.
3. Approximating the rapid equilibrium problems with a high kinetic constant (see Eq. (4b)), enabled to solve the problems as ODE systems instead of DAE systems. This is an advantage due to the many efficient ODE solvers implemented in Julia. Furthermore, The QNDF solver was the most performant ODE solver generally, but the solver struggled with solving very stiff systems.
4. CADET-Julia was generally faster than CADET-DG for various reasons, but the most important factor was probably the different time integrators in each of the software packages.

This study has shown that CADET-Julia is an efficient and flexible Julia package for solving chromatography models. Additionally, CADET-Julia does not need any Jacobian information to rapidly compute the solution. This work showcases the advantage of CADET-Julia as a tool for rapid prototyping and experimentation: Conducting a study on implicit time integration methods in CADET-Core, as performed here with CADET-Julia, is possible, but would have been remarkably more complex and time consuming. Whereas CADET-Julia is useful for rapid prototyping, CADET-Core is an established code base with a more extensive model base and complex functionalities (e.g. parameter sensitivities). For the shared models between CADET-Julia and CADET-Core, it is possible to call CADET-Julia from CADET-Process if desired to use the Julia code base.

CRediT authorship contribution statement

Jesper Frandsen: Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis. **Jan Michael Breuer:** Writing – original draft, Software, Methodology, Investigation. **Johannes Schmölder:** Writing – review & editing, Software. **Jakob Kjøbsted Huusom:** Writing – review & editing, Supervision. **Krist V. Gernaey:** Writing – review & editing, Supervision. **Jens Abildskov:** Writing – review & editing, Supervision, Project administration. **Eric von Lieres:** Writing – review & editing, Supervision, Resources, Project administration.

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 (Inno4Vac). This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme 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.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.compchemeng.2024.108913.

Data availability

Data will be made available on request.

References

Andersson, D., Sjögren, R., Corbett, B., 2023. Numerical simulation of the general rate model of chromatography using orthogonal collocation. Comput. Chem. Eng. 170, 108068. http://dx.doi.org/10.1016/j.compchemeng.2022.108068, URL: https://www.sciencedirect.com/science/article/pii/S0098135422004008.

Bassi, F., Rebay, S., 1997. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations. J. Comput. Phys. 131 (2), 267–279. http://dx.doi.org/10.1006/jcph.1996.5572, URL: https://www.sciencedirect.com/science/article/pii/S0021999196955722.

Berninger, J.A., Whitley, R.D., Zhang, X., Wang, N.H.L., 1991. A versatile model for simulation of reaction and nonequilibrium dynamics in multicomponent fixed-bed adsorption processes. Comput. Chem. Eng. 15 (11), 749–768. http://dx.doi.org/10.1016/0098-1354(91)85020-U, URL: https://www.sciencedirect.com/science/article/pii/009813549185020U.

Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B., 2017. Julia: A fresh approach to numerical computing. SIAM Rev. 59, 65–98. http://dx.doi.org/10.1137/141000671.

Breuer, J.M., Leweke, S., Schmölder, J., Gassner, G., von Lieres, E., 2023. Spatial discontinuous Galerkin spectral element method for a family of chromatography models in CADET. Comput. Chem. Eng. 177, 108340. http://dx.doi.org/10.1016/j.compchemeng.2023.108340, URL: https://www.sciencedirect.com/science/article/pii/S0098135423002107.

Carta, G., Jungbauer, A., 2020. Protein chromatography : process development and scale-up. vol. 1485, Wiley-VCH, p. 423. http://dx.doi.org/10.1007/978-1-4939-6412-3,

Eschle, J., Gál, T., Giordano, M., Gras, P., Hegner, B., Heinrich, L., Acosta, U.H., Kluth, S., Ling, J., Mato, P., Mikhasenko, M., Briceño, A.M., Pivarski, J., Samaras-Tsakiris, K., Schulz, O., Stewart, G.A., Strube, J., Vassilev, V., 2023. Potential of the julia programming language for high energy physics computing. Comput. Softw. Big Sci. 7, http://dx.doi.org/10.1007/s41781-023-00104-x.

Frandsen, J., Breuer, J.M., Schmölder, J., Abildskov, J., Abildskov, J.K.H., Gernaey, K.V., von Lieres, E., 2024. Supplement for "CADET-Julia: An efficient and versatile, open-source chromatography solver in Julia with extensive benchmarking". URL: https://github.com/jespfra/CADET-Julia-Batch-Benchmarks.

Gardner, D.J., Reynolds, D.R., Woodward, C.S., Balos, C.J., 2022. Enabling new flexibility in the SUNDIALS suite of nonlinear and differential/algebraic equation solvers. ACM Trans. Math. Softw. http://dx.doi.org/10.1145/3539801.

Gu, T., 2015. Mathematical Modeling and Scale-Up of Liquid Chromatography: With Application Examples. Springer, Google-Books-ID: PiryBwAAQBAJ.

Guennebaud, G., Benoît, J., 2010. Eigen v3. URL: http://eigen.tuxfamily.org.

Guiochon, G., Felinger, A., Shirazi, D.G., 2006. Fundamentals of Preparative and Nonlinear Chromatography. Elsevier.

Hahn, T., Huuk, T., Heuveline, V., Hubbuch, J., 2015. Simulating and optimizing preparative protein chromatography with ChromX. J. Chem. Educ. 92 (9), 1497–1502. http://dx.doi.org/10.1021/ed500854a, Publisher: American Chemical Society.

Hairer, E., Wanner, G., 1996. Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems. vol. 14, http://dx.doi.org/10.1007/978-3-662-09947-6, Journal Abbreviation: Springer Verlag Series in Comput. Math. Publication Title: Springer Verlag Series in Comput. Math..

He, Q.L., Leweke, S., von Lieres, E., 2018. Efficient numerical simulation of simulated moving bed chromatography with a single-column solver. Comput. Chem. Eng. 111, 183–198. http://dx.doi.org/10.1016/j.compchemeng.2017.12.022.

Hesthaven, J.S., Warburton, T., 2002. Nodal high-order methods on unstructured grids: I. time-domain solution of Maxwell's Equations. J. Comput. Phys. 181 (1), 186–221. http://dx.doi.org/10.1006/jcph.2002.7118, URL: https://www.sciencedirect.com/science/article/pii/S0021999102971184.

Hesthaven, J.S., Warburton, T., 2008. Nodal Discontinuous Galerkin Methods. Texts in Applied Mathematics, Vol. 54, Springer, New York, NY, http://dx.doi.org/10.1007/978-0-387-72067-8, URL: http://link.springer.com/10.1007/978-0-387-72067-8,

Hindmarsh, A.C., Brown, P.N., Grant, K.E., Lee, S.L., Serban, R., Shumaker, D.E., Woodward, C.S., 2005. SUNDIALS. ACM Trans. Math. Softw. http://dx.doi.org/10.1145/1089014.1089020, URL: https://dl.acm.org/doi/10.1145/1089014.1089020.

Jäpel, R.C., Buyel, J.F., 2022. Bayesian optimization using multiple directional objective functions allows the rapid inverse fitting of parameters for chromatography simulations. J. Chromatogr. A 1679, 463408. http://dx.doi.org/10.1016/j.chroma.2022.463408, URL: https://www.sciencedirect.com/science/article/pii/S0021967322005830.

Javeed, S., Qamar, S., Ashraf, W., Warnecke, G., Seidel-Morgenstern, A., 2013. Analysis and numerical investigation of two dynamic models for liquid chromatography. Chem. Eng. Sci. 90, 17–31. http://dx.doi.org/10.1016/j.ces.2012.12.014.

Khan, A., Perveen, S., Qamar, S., 2021. Discontinuous Galerkin scheme for solving a lumped kinetic model of non-isothermal liquid chromatography with bi-langmuir isotherms. Ind. Eng. Chem. Res. 60, 12592–12601. http://dx.doi.org/10.1021/acs.iecr.1c01074.

Kozorog, M., Caserman, S., Grom, M., Vicente, F.A., Pohar, A., Likozar, B., 2023. Model-based process optimization for mAb chromatography. Sep. Purif. Technol. 305, http://dx.doi.org/10.1016/j.seppur.2022.122528.

Kronbichler, M., 2021. The discontinuous Galerkin method: Derivation and properties. In: Kronbichler, M., Persson, P.-O. (Eds.), Efficient High-Order Discretizations for Computational Fluid Dynamics. In: CISM International Centre for Mechanical Sciences, Springer International Publishing, Cham, pp. 1–55. http://dx.doi.org/10.1007/978-3-030-60610-7_1.

Kumar, V., Lenhoff, A.M., 2020. Mechanistic modeling of preparative column chromatography for Biotherapeutics. Annu. Rev. Chem. Biomol. Eng. 11 (1), 235–255. http://dx.doi.org/10.1146/annurev-chembioeng-102419-125430, _eprint: https://doi.org/10.1146/annurev-chembioeng-102419-125430.

Leweke, S., von Lieres, E., 2018. Chromatography analysis and design toolkit (CADET). Comput. Chem. Eng. 113, 274–294. http://dx.doi.org/10.1016/j.compchemeng.2018.02.025, URL: https://www.sciencedirect.com/science/article/pii/S0098135418300966.

von Lieres, E., Andersson, J., 2010. A fast and accurate solver for the general rate model of column liquid chromatography. Comput. Chem. Eng. 34 (8), 1180–1191. http://dx.doi.org/10.1016/j.compchemeng.2010.03.008, URL: https://www.sciencedirect.com/science/article/pii/S0098135410000992.

LinearSolve.jl, 2024. Linear system solvers. URL: https://docs.sciml.ai/LinearSolve/stable/solvers/solvers/.

Meyer, K., Leweke, S., von Lieres, E., Huusom, J.K., Abildskov, J., 2020. ChromaTech: A discontinuous Galerkin spectral element simulator for preparative liquid chromatography. Comput. Chem. Eng. 141, 107012. http://dx.doi.org/10.1016/j.compchemeng.2020.107012, URL: https://www.sciencedirect.com/science/article/pii/S009813542030329X.

Nogueira, I.B.R., Santana, V.V., Ribeiro, A.M., Rodrigues, A.E., 2022. Using scientific machine learning to develop universal differential equation for multicomponent adsorption separation systems. Can. J. Chem. Eng. 100, 2279–2290. http://dx.doi.org/10.1002/cjce.24495.

Püttmann, A., Schnittert, S., Naumann, U., von Lieres, E., 2013. Fast and accurate parameter sensitivities for the general rate model of column liquid chromatography. Comput. Chem. Eng. 56, 46–57. http://dx.doi.org/10.1016/j.compchemeng.2013.04.021, URL: https://www.sciencedirect.com/science/article/pii/S0098135413001440.

Rackauckas, C., Nie, Q., 2017. DifferentialEquations.jl – A performant and feature-rich ecosystem for solving differential equations in julia. J. Open Res. Softw. 5, 15. http://dx.doi.org/10.5334/jors.151.

Santana, V.V., Costa, E., Rebello, C.M., Ribeiro, A.M., Rackauckas, C., Nogueira, I.B., 2023. Efficient hybrid modeling and sorption model discovery for non-linear advection-diffusion-sorption systems: A systematic scientific machine learning approach. Chem. Eng. Sci. 282, 119223. http://dx.doi.org/10.1016/j.ces.2023.119223.

Schmidt-Traub, H., Schulte, M., Seidel-Morgenstern, A., 2020. Preparative Chromatography. John Wiley & Sons, Google-Books-ID: 7H_PDwAAQBAJ.

Schmölder, J., Kaspereit, M., 2020. A modular framework for the modelling and optimization of advanced chromatographic processes. Processes 8 (1), 65. http://dx.doi.org/10.3390/pr8010065, URL: https://www.mdpi.com/2227-9717/8/1/65, Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

SciML, 2021. Sciml ecosystem update: Improved QNDF outperforms CVODE on SciML benchmarks. URL: https://sciml.ai/news/2021/05/24/QNDF/.

Shampine, L.F., 2002. Solving 0=f(t,y(t),y'(t)) in matlab. J. Numer. Math. 10 (4), 291–310. http://dx.doi.org/10.1515/JNMA.2002.291.

Shampine, L., Reichelt, M., Shampine, L.F., Reichelt, M.W., 1997. The MATLAB ODE suite. SIAM J. Sci. Comput. http://dx.doi.org/10.1137/S1064827594276424\"i, URL: https://hal.science/hal-01333731.

Winters, A.R., Kopriva, D.A., Gassner, G.J., Hindenlang, F., 2021. Construction of modern robust nodal discontinuous Galerkin spectral element methods for the compressible Navier–Stokes equations. In: Kronbichler, M., Persson, P.-O. (Eds.), Efficient High-Order Discretizations for Computational Fluid Dynamics. In: CISM International Centre for Mechanical Sciences, Springer International Publishing, Cham, pp. 117–196. http://dx.doi.org/10.1007/978-3-030-60610-7_3.

Zafar, S., Perveen, S., Qamar, S., 2021. Discontinuous Galerkin scheme for solving non-isothermal and non-equilibrium model of liquid chromatography. J. Liq. Chromatogr. Relat. Technol. 44, 52–69. http://dx.doi.org/10.1080/10826076.2020.1867164.

Zafar, S., Perveen, S., Qamar, S., 2023. Discontinuous Galerkin finite element scheme for solving non-linear lumped kinetic model of non-isothermal reactive liquid chromatography. Korean J. Chem. Eng. 40 (3), 555–571. http://dx.doi.org/10.1007/s11814-022-1352-4.

Zhang, W., Przybycien, T., Schmölder, J., Leweke, S., von Lieres, E., 2024. Solving crystallization/precipitation population balance models in CADET, part I: Nucleation growth and growth rate dispersion in batch and continuous modes on nonuniform grids. Comput. Chem. Eng. 183, 108612. http://dx.doi.org/10.1016/j.compchemeng.2024.108612, URL: https://www.sciencedirect.com/science/article/pii/S0098135424000309.

Zydney, A.L., 2016. Continuous downstream processing for high value biological products: A review. Biotechnol. Bioeng. 113, 465–475. http://dx.doi.org/10.1002/bit.25695.