Automated Generation of Mechanistic Models for Chemical Process Digital Twins using Reinforcement Learning

Part I: Conceptual Framework and Equation Generation

Mathis Heyer[a,b], Jiyizhe Zhang[a,c*], Naoto Sugisawa[a,d], Jan-Frederic Laub[a,b], and Alexei A. Lapkin[a,c,e*]

[a] Department of Chemical Engineering and Biotechnology, University of Cambridge, Cambridge CB3 0AS, United Kingdom

[b] Process Systems Engineering (AVT.SVT), RWTH Aachen University, 52074 Aachen, Germany

[c] Innovation Centre in Digital Molecular Technologies, Yusuf Hamied Department of Chemistry, University of Cambridge, Lensfield Road, Cambridge CB2 1EW, United Kingdom

[d] Department of Basic Medicinal Sciences, Graduate School of Pharmaceutical Sciences, Nagoya University, Nagoya, 464-8601, Japan

[e] Cambridge Centre for Advanced Research and Education in Singapore Ltd., 1 Create Way, CREATE Tower #05-05, Singapore 138602, Singapore

* Corresponding authors: jz596@cam.ac.uk and aal35@cam.ac.uk

Abstract

Deriving versatile and robust mechanistic models from experimental data is a key challenge in engineering and natural sciences. This is especially true in chemical reaction engineering, where reactor manufacturers and operators increasingly pursue the development and maintenance of digital twins that rely on frequent model updates and ask for automation of this modeling process. In this work, we propose an automated workflow that generates accurate mechanistic reactor models from experimental concentration data of a given reactor. At the core of this workflow, a reinforcement learning agent assembles an interpretable reactor model by iteratively simplifying general differential balance equations and fitting the resulting candidate model to experimental data. We demonstrate the performance of our workflow in two case studies. An *in silico* case study shows that the workflow correctly reconstructs the model underlying a synthetic data set, is robust against noise in the input data, and has favorable scaling properties. The agent accelerates the model derivation process significantly compared to an exhaustive enumerative search. Secondly, an experimental case study is conducted employing a Taylor-Couette prototype reactor. A liquid-phase esterification reaction of (2-bromophenyl)methanol and acetic anhydride was used as a test system. Based on the experimental data, the workflow derives meaningful mechanistic models, with the most accurate model showing a normalized root mean squared error of 2.4%. Future work encompasses the integration of automated

experiments into the workflow and the transfer of our workflow to process units beyond chemical reactors.

Keywords: Digital twins; Mechanistic model; Reinforcement learning; Taylor-Couette vortex reactor

1. Introduction

The transition from traditional chemical and pharmaceutical processes to sustainable manufacturing has become more urgent than ever. To address the growing demands of competitive markets and sustainability goals, it is essential to accelerate process development and scale-up, reducing the time to market for new products. This urgency has driven significant advancements in synthetic chemistry and process engineering, including the development of scalable synthetic routes (Horn et al., 2016; Politano and Oksdath-Mansilla, 2018), the shift from batch to continuous-flow processes (Capaldo et al., 2023), and the design of intensified reactors with enhanced efficiency (Plutschack et al., 2017).

Digitalization has emerged as a transformative approach to tackle these challenges and digital twins is one of the key enabler for accelerating process development (Fantke et al., 2021). Digital twins create a virtual representation of physical processes, allowing for simulation, testing, and optimization in a digital environment. For instance, a digital twin enables reactor manufacturers to evaluate their novel designs for new chemical reactions, identify optimal operating conditions, or train operators for practical applications. Despite their promising applications, the widespread adoption of digital twins remains limited by the challenge of quickly developing models that effectively integrate data from physical systems.

A critical challenge lies in the development of mechanistic models for digital twins. While mechanistic models are highly interpretable and provide more detail and better extrapolation behavior than data-driven models, developing mechanistic models for digital twins is a specialized and time-consuming task. The process typically involves skilled modelers to analyze systems, formulate assumptions, construct models, solve equations, and compare predictions with experimental data. If the model predictions fail to capture the trends of experimental data, models need to be revised through repeated iterations, which can be time-consuming.

Although software tools such as ASPEN, gPROMS and Scale-up Suite have simplified the model generation process, these tools often require expensive licenses and lack the modularity needed for novel processes and reactor designs. These tools are designed for a human model developer, and at present do not include automation that may allow to scale the development of digital twins. The role of model development is restricted to

highly experienced process engineers and these skills are typically not widely accessible within the organisations. Consequently, there is a need for accessible, automated workflows that democratize model development, streamline the modeling process, and facilitate the broader adoption of digital twins in process development.

With recent advancements in artificial intelligence, data-driven agents have emerged as a promising approach to assist in process development and design. In terms of the modeling task, automated model generation has been investigated on both the process level and the kinetic level. For example, Khan and Lapkin (2020) proposed the use of reinforcement learning to automate the generation of process flowsheets. Instead of exhaustively enumerating all possible process pathways from reactant to product, an agent learns which sequence of process units results in the most cost-efficient process. The authors later extend their work by introducing a hierarchical approach where a higher-level reinforcement learning agent decides on the arrangement of "process sections" whereas a lower-level agent specifies the process units within each section (Khan and Lapkin, 2022; Seidenberg et al., 2023). In this way, reward sparsity is addressed leading to a modular, more stable learning process. A similar approach is presented by Stops et al. (2023) who enhance hierarchical reinforcement learning for process design by representing the process flowsheets as graph convolutional neural networks and employing a modified proximal policy agent, instead of Q-learning as used by Khan and Lapkin (2020). For kinetic models, Neumann et al. (2020) identify kinetic models as well as the relationship between shear stress and shear rate from robotically collected, noisy data using a globally optimal symbolic regression approach. Carvalho Servia et al. (2024) employ genetic programming to detach the kinetic model generation process from model structure assumptions and to improve scalability. However, a clear gap remains in the automation of reactor-level model generation – a crucial area that links with scalability in process development. Reactor models span a spectrum of complexity, from simple idealized representations to detailed computational fluid dynamics (CFD) simulations (Haag et al., 2018). Automation of reactor-level model generation therefore requires the development of a workflow capable of accommodating diverse levels of complexity and adapting to a wide range of processes.

This work aims to bridge this gap by developing an automated workflow for mechanistic reactor model generation for digital twin applications, leveraging hierarchical reinforcement learning agents. In this work, Part I of the 2-part study, we focus on: (1) proposing a structured workflow for automated model generation using reinforcement learning agents, (2) designing an equation generation module, which is the centrepiece of the workflow, and (3) testing the workflow in an *in silico* and an experimental case studies, where the latter involves data collection from a prototype Taylor-Couette reactor with an esterification reaction. Part II of this work will focus on generalizing the workflow towards digital twin applications (Laub et al., 2025).

2. Methodology

In this section, we first outline the proposed workflow for automated reactor model generation based on experimental data. Then, we discuss the implementation details of the central, reinforcement learning-based model equation generator module. Finally, we introduce the two case studies where the workflow has been implemented and tested in an *in silico* case study of a one-dimensional dispersion model, and an experimental case study where data is collected from a novel Taylor-Couette flow reactor.

2.1. Workflow design

2.1.1. Design principles

The workflow is designed to satisfy six core requirements either pertaining to the workflow itself or the models generated by the workflow.

(1) Interpretability. The workflow derives a differential-algebraic system of mechanistic equations that resembles traditional, mechanistic model equations and is therefore interpretable. Interpretable models are usually better verifiable through measurements, adaptable to changing reactor geometries, chemistries, and operating conditions, and valid beyond the bounds of the data used to infer the model. Interpretable mechanistic models can be used as a store of knowledge, where knowledge enhancement can be tracked.

(2) Generality. The workflow needs to be capable of capturing any continuous flow reactor type with any chemistry at any operating conditions in a model. This is ensured by including: (i) a compartmentalization module as a sufficiently fine model discretization captures the most complex hydrodynamic flow patterns or multiphase processes, and (ii) integrating a comprehensive ontologically organized database of constitutive equations.

(3) Transferability of workflow and models. Although the workflow is developed to generate reactor models, it is easily adaptable to create models for other chemical process units, e.g., heat exchangers. Moreover, the workflow should capture the physical system well enough so that similar physical systems, i.e., systems with minor changes in the geometry, operating conditions, and chemistry, can be modelled with minimal or no additional experimental data but rather via inference from the original system. For that, a mapping between problem profiles and models can be developed.

(4) Computation time. A fast workflow is key to using this technology to update digital twins or use it in inline applications. The proposed workflow can be executed on a laptop computer in a matter of minutes and thus outpaces human model development and testing.

(5) Adjustable accuracy. The workflow is designed to create models that capture the physical system at any desired level of accuracy. For this, the compartmentalization module plays a key role as a sufficiently fine discretization captures phenomena with

arbitrary accuracy. Moreover, the estimation of the residual error can be integrated into the post-processing module.

(6) Modularity. The workflow is designed in a modular way allowing for parallel development and testing of the modules, and quick updates and refinements.

2.1.2. Workflow module design

Based on the outlined design principles, the proposed automated workflow is structured into seven modules that adapt the traditional modeling process (Preisig, 2010; Andres Mahecha-Botero et al., 2007), see Figure 1.

The **input module** queries the human modeler for known *a priori* information about the reactor and compiles a "problem profile" that is passed to the compartmentalization module. The problem profile may include information on the reactor geometry and type, the chemical system, the operating conditions, and available experimental data, e.g., concentration curves. All information that is not provided to the input module will be established in the remaining process; however, the provision of experimental data is mandatory.

The task of the **compartmentalization module** is to identify a compartmentalization of the physical system that is located on the continuum between a single ideal reactor (one compartment) and a topological model (e.g., a CFD-like model with millions of compartments) so that the desired model accuracy is achieved at minimal computational cost (Haag et al., 2018; Jourdan et al., 2019; Stegehake et al., 2018). The output of the compartmentalization module is a graph that captures the chosen topological structure of the model. Here, nodes represent balance volumes and edges represent interfaces between the sub-volumes. Part II of this work discusses the compartmentalization module in detail (Laub et al., 2025).

For each balance volume and each balance interface, the **equation generation module** uses a reinforcement learning approach to identify a set of equations that reflect the behavior of the sub-volumes and their interconnections. As input, the module receives the topological graph from the compartmentalization module as well as the problem profile. The module starts from general differential balance equations for extensive properties such as mass (or moles), energy, or momentum for the present species and gradually specifies it. For that, the module queries an expansive, ontologically organized **library of constitutive equations**. A core challenge in designing the equation generation module is to identify a computational representation of a system of equations that allows mathematical manipulation of the equations and the substitution of terms by constitutive equations, Section 2.2.3. The output of the equation generation module is a machine-readable model that can be executed using software such as Pyomo.DAE (Nicholson et al., 2018), GAMS (GAMS, 2021), OpenModelica (Fritzson, 2020), or Julia.JuMP (Lubin et al., 2023).

The **test module** receives as input the mechanistic model derived by the equation generation module. The module's task is to infer a suitable solver for the given model and test its validity (e.g., solvability, structural identifiability, and observability). If the model is identified as invalid, the model is passed back to the compartmentalization and equation generation modules to improve the formulation.

If a model passes the test module, it is sent to the **solver module**. Together with a **parameter estimation module**, the task of the solver module is to fit the model parameters so that the model's behavior matches the experimental, possibly time-variant and multidimensional, concentration data provided in the problem profile. Conceptually, this corresponds to a bilevel optimization problem where the repeated solution of the partial-differential algebraic model is embedded into a parameter estimation optimization. The output of the solver module is the parameterized model along with its accuracy. If the accuracy falls short of expectations, a new model is developed by the compartmentalization and equation generation modules until the accuracy is met. To accelerate this iterative process, the performance of previously developed models is made available to the equation generation module. This notion motivates the design of the model equation generation module as a reinforcement learning agent as described in Section 2.2.

Finally, the **post-processing module** integrates tasks that help to further improve the quality of the determined model, e.g., by learning the residual modeling error using a hybrid machine learning architecture, by performing uncertainty quantification, or by developing a computationally inexpensive surrogate model.

While this work focuses on the equation generation module and compartmentalization module respectively, more detailed descriptions of the other modules are provided in the Supplementary Materials.

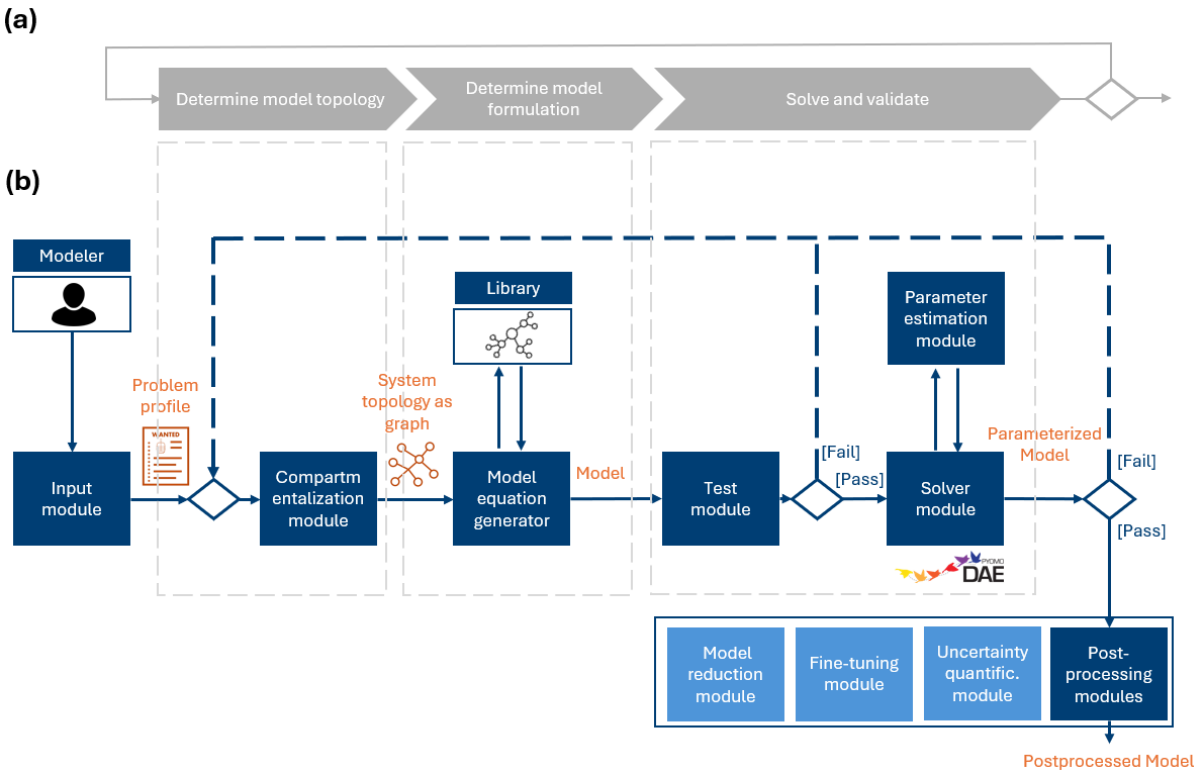


Figure 1: An illustration of the overall model generation workflow. (a) Steps of the manual system-theoretic workflow for the generation of reactor models. (b) Proposed automated workflow. The grey-dashed boxes relate the modules of the proposed workflow to the steps of the manual workflow.

2.1.3. Equation generation module

As the centrepiece of the proposed workflow, the equation generation module derives a mechanistic mathematical model for each reactor compartment as well as the interfaces between the compartments. In the most general case, such a model is a partial-differential algebraic system of equations including the necessary initial and boundary conditions to solve the model. Solving this system of equations yields, for instance, the reactor temperature and concentration profiles that chemical reaction engineers are interested in.

The module derives the reactor model by starting from the most general form of the differential mass, energy, and momentum balance equations for volumes and interfaces. Equations 1 and 2 show the differential volume and surface balances where ψ represents the intensive balance quantity. The five terms in the differential volume equation reflect accumulation, convective transport, diffusive transport, production, and supply. In the differential surface equation, Γ denotes surface properties, $+$ and $-$ distinguish the two sides of the surface, and $\mathbf{n}$ is the normal vector. Substituting ψ with ρ, ρh, ρs or ρv yields the differential mass, energy, entropy, and momentum balance respectively.

$$0 = \frac{\partial \psi}{\partial t} + \boldsymbol{\nabla} \cdot \psi \mathbf{v} + \boldsymbol{\nabla} \cdot \phi_\psi - \sigma_\psi^P - \sigma_\psi^F \tag{1}$$

$$0 = -\left[\psi^+[\mathbf{v}^+ - \mathbf{u}^\Gamma] + \phi_\psi^+ - \psi^-[\mathbf{v}^- - \mathbf{u}^\Gamma] - \phi_\psi^+\right] \cdot \mathbf{n} + \sigma_\psi^\Gamma \tag{2}$$

In this work, we automate the derivation of the reactor model using a reinforcement learning approach. This is based on the notion, that the derivation process can be understood as a sequence of decisions. Specifically, for each variable in the differential balance equation one of four actions must be chosen:

(1) Neglection of the variable. When a variable represents a phenomenon not relevant to the physical system at hand, the variable can be neglected in the system of equations. Common examples are radiative energy supply, which rarely makes a measurable contribution to the energy balance in most chemical reactor systems, or terms with time derivatives in stationary processes.

(2) Substitution with a known parameter. When modeling a reactor, often key quantities of the process are known, such as the input flow rate, input concentrations, geometric dimensions, or kinetic constants that were measured in separate experiments. These values are provided to the equation generation module via the problem profile. The module can include this prior knowledge by setting the corresponding variables to the provided values.

(3) Declaration as a parameter that will be estimated. Other parameters are not known *a priori*, for instance, diffusion coefficients, heat transfer coefficients, or the reaction order. These internal properties cannot be measured directly and are usually unknown. However, they can be inferred from the experimental data provided to the workflow. The module therefore sets these variables as "to-be-estimated parameters" whose values will be identified in the parameter estimation module.

(4) Substitution with a constitutive equation. Constitutive equations exist that further specify the behavior of the variable. This domain knowledge is made available to the equation generation module via the ontological database. The module queries the database for suitable constitutive equations and adds them to the system of equations. Notably, the added constitutive equation likely contains new variables and terms on which a decision has to be made.

These decisions are made successively for each variable in the differential equation and all subsequently added constitutive equations until no undecided variables remain. This sequential decision-making process is illustrated in Figure 2. Note that each decision for one of the actions corresponds to an assumption in the modeling process. The module can therefore conveniently provide a detailed list of the assumptions made.

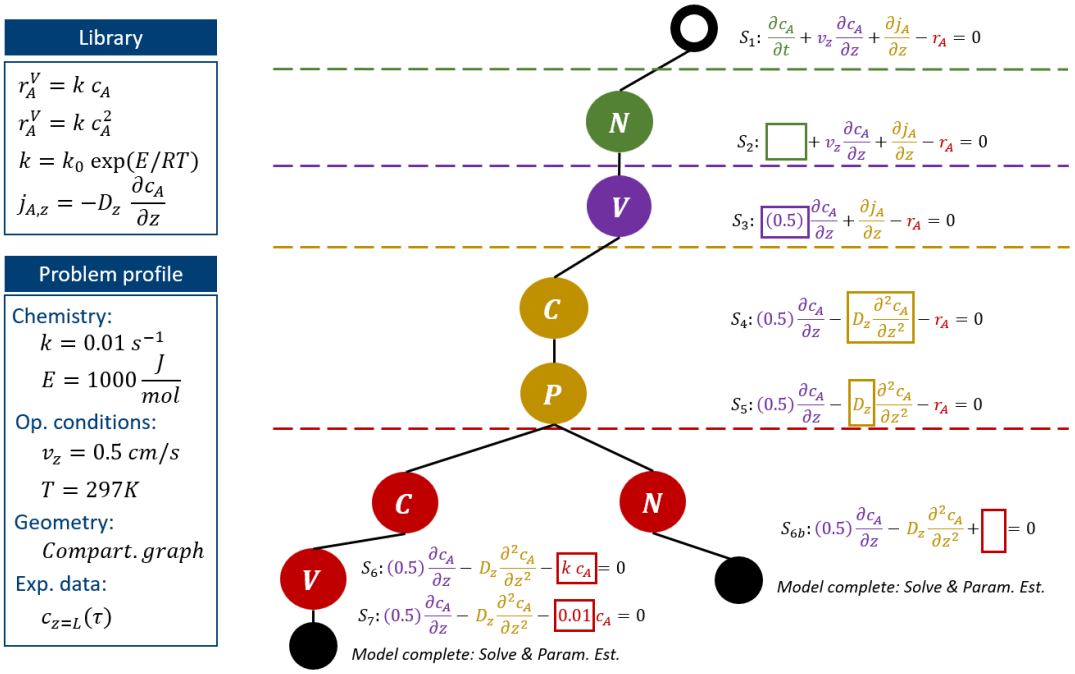


Figure 2: Sequential decision-making for model derivation: The derivation of a reactor model from a general differential balance equation is a sequential decision-making process, where for each term successively one of four actions is taken: Neglecting (N), Substitution with a known value from the problem profile (V), Declaration as to-be-estimated parameter (P), Substitution with a constitutive equation (C). The figure shows an exemplary state-action trajectory starting at the differential mass balance (State S1). The latest decision that has resulted in the current state is visualized with a box. Colours are used to match terms and decisions made on them. Actions are taken until no undecided variables remain. Terminal nodes therefore represent fully specified reactor models. Note that in this illustration no actions are taken on the partial derivative $\frac{\partial c_A}{\partial z}$ for clarity, however, in our implementation actions can be performed on differential variables like on any other variable.

The generated complete model is passed on to the solver and parameter estimation module. After fitting the to-be-estimated parameters to the experimental data, the remaining error between the fitted model and experimental data indicates whether a good model was developed. If the model does not satisfy the accuracy requirements, the equation generation module is tasked with developing a new model. For this, the module makes different decisions in the model derivation process, i.e., it chooses a different trajectory of actions. However, due to the complexity of reactor models and the large number of available constitutive equations, it is computationally infeasible to test all possible decision trajectories to determine which one delivers the best model.

Instead, we use reinforcement learning to identify the sequence of actions yielding the best model in as few attempts as possible. Reinforcement learning is a machine-learning technique that is particularly well-suited for identifying the optimal trajectory in such a

sequential decision-making problem or similar combinatorial optimization problems (Mazyavkina et al., 2021).

2.2. Reinforcement learning agent

In reinforcement learning, an "agent" is subjected to an "environment". The agent perceives the state of the environment and uses his internal decision-making rule, the "policy", to decide on an action to take. The action leads to a change in the state of the environment. A "reward function" quantifies whether the incurred change was favorable and returns a reward value to the agent, who adapts his policy based on the received reward. Then, the agent perceives the new state and decides on the next action using the updated policy. This iterative process generates a state-action trajectory and is repeated until the agent has achieved its application-dependent goal in the environment. The agent is then subjected to a fresh environment to further tailor its policy function to the problem at hand.

2.2.1. Environment

The proposed workflow formulates the reactor model generation process as a sequential decision-making process. For solving sequential decision-making problems with reinforcement learning, the problems are mathematically described as Markov Decision Processes (MDP). An MDP is defined by a tuple $M = \{S, A, R, T\}$, where S denotes the state space, A is the action space, R is the reward function, and T is the transition function.

Our decision-making process for automated model generation can be represented as an MDP as follows:

(1) The **state space $\boldsymbol{S}$** is the set of all states $s_i \in S$ that the environment can be in. The initial state is the system of differential mass, energy, and momentum balance equations. When, for example, the agent decides to take action on the first variable in the first equation, e.g., by substituting it with a known parameter value from the problem profile, this results in a new state: The first variable is "decided" to be a "known parameter", whereas all other variables remain "undecided". In summary, the state of the environment is the system of equations alongside the information, which variables have been specified and by which actions.

(2) The **action space $\boldsymbol{A}$** is the set of all actions $a_i \in A$ that the agent can take on undecided variables. As specified before, this includes neglecting the variable, substituting the variable with a known parameter from the problem profile, declaration of a variable as to-be-estimated parameter, and substituting the variable with a constitutive equation from the ontological database. Note that the number of available actions for an undecided variable can be larger than four when multiple candidate constitutive equations exist for the variable.

(3) The **reward function** $\boldsymbol{R(a_i, s_i)}$: $\boldsymbol{S \times A \rightarrow \mathbb{R}}$ quantifies the success or value of an action a_i in a given state s_i. In reinforcement learning, two classes of problems exist: Firstly, problems where a separate reward value can be computed for every state-action pair, and secondly, problems where only at the terminal state, i.e., the end of the trajectory of state-action pairs, the success of the actions leading to this terminal state can be determined. The latter is the case for our automated model generation problem: The quality of the derived model can only be established once all variables are specified so that the model is executable. A comparison of the model's predicted concentration or temperature curves with experimental data yields an error which is a measure of the model's quality and the actions' value. Specifically, in this work, two reward functions are tested: Firstly, Equation 3 shows the squashing function, as used by Petersen et al. (Petersen et al., 2021) mapping the sum squared error (SSE) between model prediction and experimental data to a value between 0 and 1. A low SSE translates to a high reward and vice versa. Secondly, Equation 4 shows an alternative reward function that additionally rewards simpler models over complex models (Bassenne and Lozano-Durán, 2019). In other words, the agent is encouraged to generate simpler models, if the error is comparable. We quantify the complexity of a model as the length of its representation vector, see Section 2.2.3. The parameter q is chosen to the value of $q = 0.125$ based on the average length of states in our problem to achieve favorable scaling.

$$R(a_i, s_i) = \frac{1}{1 + SSE} \tag{3}$$

$$R(a_i, s_i) = \frac{1}{p+SSE} + \frac{q^{-1}}{length(s_i)} \qquad \text{with } p = 1, q = 0.125 \tag{4}$$

Finally, a penalty is added for trajectories that result in unsolvable models. To avoid sudden major perturbations during the learning process, a reward of 0.1 is chosen for unsolvable models as opposed to a reward of 0. The complete formulation of the reward function is shown in Equation 5.

$$R(a_i, s_i) = \begin{cases} \text{Eq. 3 or 4}, & \text{if model is solvable} \\ 0.1, & \text{otherwise} \end{cases} \tag{5}$$

(4) The **transition function** $\boldsymbol{T(s_{i+1} | s_i, a_i)}$ in MDPs expresses the probability that action a_i in state s_i will lead to a transition of the environment to state s_{i+1}. In our case, these transitions are deterministic, i.e., taking action a_i in state s_i will always result in the same state s_{i+1}. Therefore, no transition function has to be specified.

2.2.2. Agent

The agent uses received rewards to learn an optimal decision policy $\pi(a_i|s_i)$ that determines which action a_i to take being a state s_i. In our implementation, we choose a value-based Q-learning approach where the agent relies on an action value function

$Q(s, a)$ that estimates the expected reward of taking action a in state s. The agent's optimal policy in state s is choosing the action a that maximizes the value function $Q(s, a)$ (Mazyavkina et al., 2021).

Compared to other value-based and policy-based approaches, the Q-learning approach best accommodates two particularities of our problem: Firstly, the number of available actions is not the same at each state but rather depends on how many constitutive equations are available for the next considered variable. Secondly, the states are not represented as a vector of real numbers, as will be explained in Section 2.2.3. Such a representation, however, is needed for policy-based agents. At the heart of Q-learning, a tabular data structure, the Q-table, records the expected reward $Q(s, a)$ for all available actions a in state s. We compute $Q(s, a)$, the Q-values, as the average reward of those trajectories that passed through the state-action pair (s, a) during the training process, as is customary in the field of Monte Carlo tree search methods (Browne et al., 2012). As initialization values, we choose $Q(s, a) = 0.5$ as the rewards range from 0 to approximately 1.

A fully trained agent makes decisions by always choosing the action a with the highest $Q(s, a)$ value for a given state. The agent follows the "optimal policy". However, while training, the agent adopts a "training policy" to balance exploration and exploitation rather than keep moving along the same trajectory in each episode. In this work, we test an ϵ-policy and a dynamic ϵ-policy and compare their impact on training performance. In the former, given a state and the available actions in that state, the agent makes a random decision among the available actions with a probability of ϵ, whereas with a probability of $1 - \epsilon$ the action with the best Q-value is chosen, see Equation 6. A high ϵ therefore favors exploration, while a low ϵ increases exploitation. Accordingly, $\epsilon = 1$ corresponds to an agent that guesses models entirely randomly, whereas in the other edge case $\epsilon = 0$ the agent would get stuck in the same trajectory. The dynamic ϵ-policy enhances this concept by gradually decreasing ϵ throughout the training process, thereby reducing exploration and increasing exploitation over time.

$$a_i = \begin{cases} \text{argmax Q(s, a)}, & \text{with probability } 1 - \epsilon \\ \text{random action}, & \text{with probability } \epsilon \end{cases} \quad (6)$$

2.2.3. States

A key challenge is to develop a meaningful computational representation of the state. The agent operates in a state space where each state is a partial-differential algebraic system of equations with variables that have either been "decided" (i.e., an action was performed on them) or are still "undecided".

The computational representation of these complex states must fulfil two requirements. Firstly, the state must be easy to process for the agent. In particular, the agent must be able to identify the next undecided variable in the system of equations. Secondly, the

state must be easy to translate into a machine-readable optimization model, e.g., a Pyomo model.

In this work, we represent the state, i.e., the partial differential system of equations, as a sequence of Python objects in postfix notation. This is illustrated in Figure 3.

The object-oriented implementation allows us to store additional information for each element of the state vector. We define four classes of elements, analogue to the types of nodes commonly used in expression trees (Angelis et al., 2023; Luo and Liu, 2018; Petersen et al., 2021): numerical values, symbolic values, unary mathematical operators that take one argument (e.g., $exp(\,)$, $sin(\,)$), and binary mathematical operators that take two arguments (e.g., $+$, $-$, $\cdot$, $/$). Most importantly, all objects of the class symbolic value possess an attribute "status" which is initialized to "undecided". The agent steps from symbolic element to symbolic element. For the first encountered symbolic element with the status "undecided" the agent performs one of the four actions and changes the element's status to "neglected", "problem-profile", "parameter-estimation", or "constitutive-equation", depending on the action. Conveniently, the object nature of the symbolic elements ensures that the neglection of a variable will be reflected in the statuses of all appearances of that variable throughout the system of equations. Our classification of operators into unary and binary operators aligns with the definitions by Lample and Charton (2019) with one exception: we classify differential terms, such as $\frac{\partial c}{\partial t}$, as a single variable, i.e., a symbolic element not a series of operators, since in Pyomo.DAE (Nicholson et al., 2018) they are handled as such.

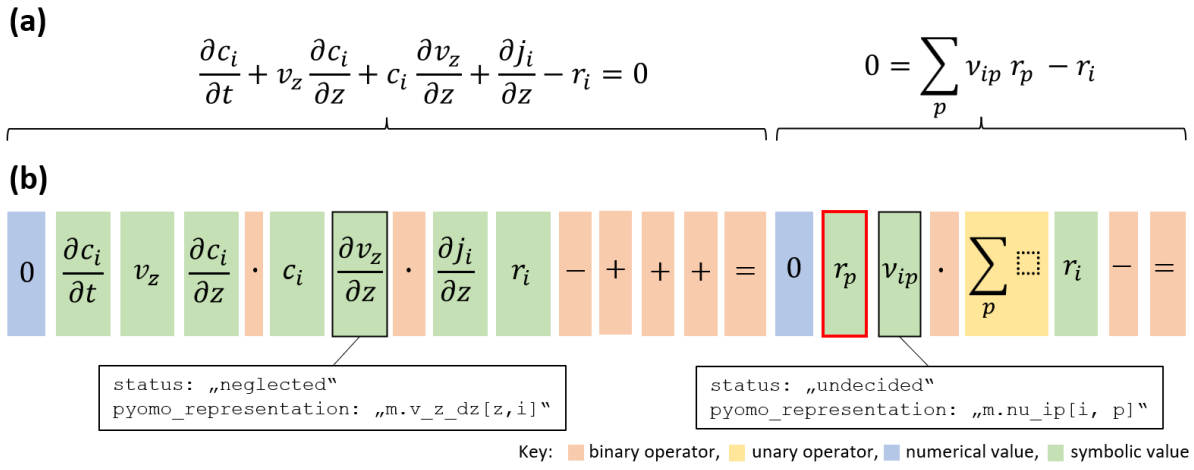


Figure 3: Infix and object-oriented postfix representation of systems of equations: (a) A differential mass balance equation and a constitutive equation for the reaction rate in infix notation. Together they form an exemplary system of equations. (b) The same system of equations as a sequence of objects in postfix notation. This state representation was developed for our workflow. The colours denote the object class of the element (red: binary operator, yellow: unary operator, blue: numerical value, green node: symbolic value). When deriving a model, the agent iterates through the symbolic elements (green) and performs one of the four actions on them. In the depicted example, the element

boxed in red is the first still ”undecided” term. Therefore, all symbolic elements before have a specified status, whereas all symbolic elements following behind are still ”undecided”. From the depicted state it can also be inferred that only for r_i the action ”substitute with constitutive equation” was chosen, as no other constitutive equations have been added to the state. Boundary and initial conditions are omitted in this example. Besides the ”status” attribute, symbolic elements possess a ”pyomo_representation” attribute that is accessed when parsing the model into pyomo code.

The postfix notation comes with three advantages. Firstly, it allows us to omit any parentheses in the state as postfix notation is unambiguous. Secondly, any equation that is rearranged to zero has in postfix notation the equal operator as the last element, see Figure 3. This conveniently signals the end of an equation and therefore paves the way for the representation of all equations of a system of equations in a single sequence. The equations are simply concatenated to each other with the equal elements functioning as natural separators. Similarly, initial and boundary conditions are treated as additional equations that are concatenated to the state. If a variable is substituted by a constitutive equation, this constitutive equation is added to the system of equations by concatenating it to the state.

The final advantage of the postfix notation is its easy evaluation using the reverse shunting yard algorithm (details in the Supplementary Materials, A.2.). While in computational mathematics this algorithm is used to determine the numerical value of a longer term, we use it to parse the terminal state, i.e., the state at the end of a state-action trajectory where all symbolic variables are specified, into an executable Pyomo model. For that, each element in the state possesses a second attribute "pyomo-representation" that stores the string representation of that element as it appears in a Pyomo model, see Figure 3. The compiled equations are inserted into a template file of a Pyomo model, as illustrated in Supplementary Materials (A.2.). The implementation of the equation generation module is summarized as pseudo-code in Scheme 1.

Scheme 1: Pseudo-code for the implementation of the equation generation module as a reinforcement learning agent.

Algorithm: Reinforcement learning agent for model equation generation

Data: Input profile, Library of constitutive equations
Return: Best-performing fitted Pyomo model
Parameters: $n_{epsiodes}$, initial $Q(s,a)$, $initial\_state$
For $episode$ in $n_{epsiodes}$ **do**:
    $state \leftarrow initial\_state$
    while $is\_model\_fully\_specified$ = False **do**:
        $action \leftarrow$ Decide on action according to training policy (Eq. 6);
        $new\_state, is\_model\_fully\_specified \leftarrow$ Apply $action$ to $state$;
        $state \leftarrow new\_state$;
    end

Parse terminal $state$ into a pyomo model;
Fit model parameters to experimental data;
$reward \leftarrow$ Compute from accuracy, model complexity, and solvability (Eq. 5);
$Q(s,a) \leftarrow$ Update Q-values of visited states using the $reward$;
end
return Optimal Pyomo model corresponding to terminal $state$ with best Q-value;

2.3. Implementation details

The focus of this work lies on the equation generation module, which is the core module of the proposed workflow. The other modules have been implemented in simplified forms to allow for the demonstration of the overall workflow. The simplified compartmentalization module approximates the reactor with a single balance volume that is passed to the equation generation module. Note that Part II of this work focuses on the implementation of the compartmentalization module (Laub et al., 2025). The equation database is implemented as a Python dictionary that is queried by the equation generation module for constitutive equations. Instead of implementing a comprehensive test module analyzing solvability, structural identifiability, and observability, the action space is constrained so that only valid models are passed to the solver. The size of the reduced search space comprises 52 distinct models that can result from the agent's actions. The solver and parameter estimation module is implemented using Pyomo's DAE extension (Nicholson et al., 2018) in conjunction with the Parmest extension (Klise et al., 2019). Derivative variables are discretized using backward finite differences. As a solver, Ipopt 3.11.1 is employed with standard settings (tol: $10 \cdot 10^{-8}$, max iter: 3000). No postprocessing modules are implemented at the moment.

Moreover, the following assumptions have been made in the equation generation module without loss of generality. Since the chosen reaction exhibits good conversions at room temperature and is neither strongly exothermic nor endothermic, the effect of temperature changes can be neglected. Accordingly, the reactor models are derived only from the differential mass balance. Moreover, the models are constrained to time-invariant models with a 0- or 1-dimensional spatial resolution modelled in cartesian coordinates. This is done, to eliminate partial-differential equations from the search space in the light of the not yet implemented testing module. Note, that the implementation of the equation generation module is not restricted to ordinary DAE models and Pyomo.DAE (Nicholson et al., 2018) is also capable of handling partial differential equations.

2.4. *In silico* case study

For the *in silico* case study, a known mechanistic model of a chemical reactor is used to generate concentration data which are fed into the workflow. The workflow derives a reactor model based on the concentration profiles. To test the functionality of the

workflow, the generated reactor model is compared to the original true model that was used to generate the synthetic data.

As a true model we use a one-dimensional dispersion model as shown in Equations 7-12. We subject the hypothetical reactor to a second-order reaction of type $A + 2B \rightarrow C$ and assume a reactor length of $L = 0.1\ \mathrm{m}$, an axial flow velocity of $v_z = 0.001\ \mathrm{m\ s^{-1}}$, $k_{01} = 0.1\ \mathrm{m\ mol^{-1}\ s^{-1}}$ ($T_{01} = 300\ \mathrm{K}$), $E_1 = 100\ \mathrm{J\ mol^{-1}}$ and $D_z = 1 \times 10^{-5}\ \mathrm{m^2\ s^{-1}}$ at ambient pressure and a temperature of $T = 298\ \mathrm{K}$. Moreover, inlet concentrations of $1\ \mathrm{mol\ m^{-3}}$, $1\ \mathrm{mol\ m^{-3}}$ and $0\ \mathrm{mol\ m^{-3}}$ are assumed for species *A*, *B*, and *C* respectively.

$$0 = v_z \frac{\partial c_i}{\partial z} + \frac{\partial j_i}{\partial z} - r_i \quad \forall i \in \{A, B, C\} \tag{7}$$

$$j_i = -D_z \frac{\partial c_i}{\partial z} \quad \forall i \in \{A, B, C\} \tag{8}$$

$$r_i = \sum_p \nu_{ip} r_p \quad \text{with } p = 1, \nu_{i1} = [-1, -2, 1] \quad \forall i \in \{A, B, C\} \tag{9}$$

$$r_p = k_p \prod_i c_i^{\lambda_{ip}} \quad \text{with } \lambda_{i1} = [1,1,0] \quad \forall p \in \{1\} \tag{10}$$

$$k_p = k_{0p} \exp\left(\frac{E_p}{R}\left(\frac{1}{T_{0p}} - \frac{1}{T}\right)\right) \quad \forall p \in \{1\} \tag{11}$$

$$0 = c_i(0) - c_{i0} \quad \text{and} \quad 0 = \frac{\partial j_i}{\partial z}(z = L) \quad \forall i \in \{A, B, C\} \quad (\text{as B. C.}) \tag{12}$$

2.5. Experimental case study

To test the workflow and the equation generation module in a real application setting, we selected a novel type of Taylor-Couette flow reactor as our “physical twin”. Experimental concentration data were recorded at the outlet of the reactor. The concentration results serve as inputs to the workflow that generates a mechanistic model of the reactor.

Specifically, a prototype ribbed Taylor-Couette reactor (from Autichem Ltd.) was set up and commissioned. Taylor-Couette reactors are characterized by a fixed, cylindrical outer shell with a rotating inner cylinder, creating an annular gap in which the reaction mixture flows. This design makes it a particularly versatile reactor type, as different flow patterns can be achieved by varying the speed and flow rate, ranging from a CSTR-like behavior with high dispersion to a PFR-like behavior with high flow segregation (Schrimpf et al., 2021), which has found to be useful for many types of reactions, such as photocatalytic reactions (Wang et al,. 2024), electrochemical reactions (Bouchon et al.,

2023), enzymatic reactions (Matsumoto et al., 2021). Further details on the Taylor-Couette reactor are provided in the Supplementary Materials (A.3.).

The experimental setup to obtain concentration data from the reactor is shown conceptually in Figure 4 and a labelled image of the setup is provided in Figure 5. The reactant solutions are prepared and stored in two bottles at room temperature and pressure. The solutions are fed to the Taylor-Couette reactor using two pumps (Vapourtec R series). The reactants enter the reactor through the bottom inlets and are pushed upwards to an outlet at top of the reactor. Samples from the head outlet are collected manually. The samples are analyzed using the high-performance liquid chromatography (HPLC) (Shimadzu).

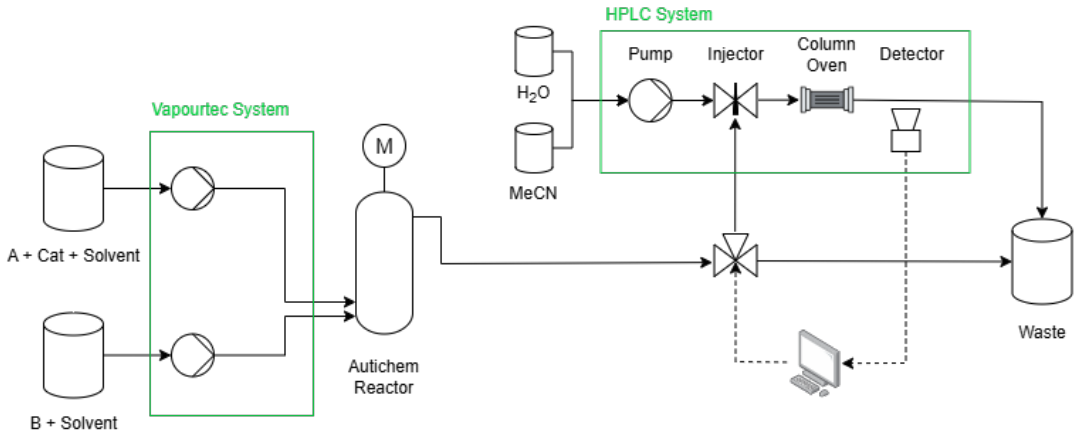
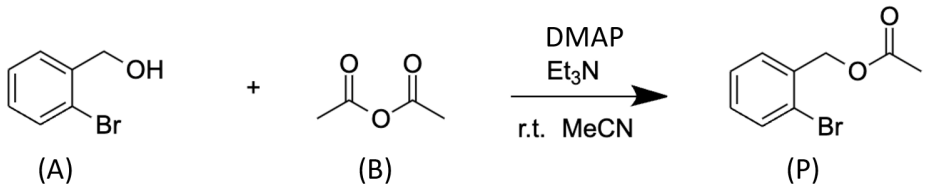


Figure 4: A flowsheet of the experimental setup for data collection from Autichem's Taylor-Couette reactor.

Figure 5: A labelled image of the experimental setup for data collection from Autichem's Taylor-Couette reactor.

As a test reaction we chose esterification of (2-bromophenyl)methanol (A) and acetic anhydride (B) to 2-bromobenzyl acetate (P) in acetonitrile (MeCN), with triethylamine

(Et_3N) as a base and 4-dimethylaminopyridine (DMAP) as a catalyst (Scheme 2). Details of the reaction system, synthesis procedures, equipment and experimental setup can be found in the Supplementary Materials (A.4., A.5.).



Scheme 2: Liquid-phase reaction of (2-bromophenyl)methanol (A) and acetic anhydride (B) to 2-bromobenzyl acetate (P) in acetonitrile (MeCN), using triethylamine (Et_3N) as base and 4-dimethylaminopyridine (DMAP) as catalyst.

In total, four sets of different experiments were conducted where the experiments differ in terms of mixing rate, and solvent as shown in Table 1. For each experiment, the outlet concentration is measured for six inlet single flow rates between 1.7 and 17 mL min^{-1} at a reactor volume of 17 mL. In the base case, two stock solutions are prepared: (1) A 0.2 M solution of (2-bromophenyl)methanol (A) (1.0 equiv) in MeCN with catalysts DMAP (0.01 equiv) and base Et_3N (1 equiv); (2) A 0.2 M solution of acetic anhydride (1 equiv) in solvent MeCN. The highest rotating speed of the Taylor-Couette reactor (360 rpm) is chosen.

Table 1: A list of experiments for four scenarios are conducted. For each, the total volumetric flowrate is adjusted between 1.7-17 mL min^{-1} to collect concentrations at the reactor outlet at the following residence times 1, 2, 4, 6, 8.5 and 10 min.

No	Scenario	Reactor	Solvent	Rotating speed [rpm]
1	Base case	Taylor-Couette	MeCN	360
2	Mixing medium	Taylor-Couette	MeCN	60
3	Mixing low	Taylor-Couette	MeCN	0
4	Solvent	Taylor-Couette	Toluene	360

3. Results and Discussion

3.1. *In silico* case study

As for the *in silico* case study, all computational work including the execution of the workflow for model generation was performed on a laptop computer with an Intel Core i5-8250U CPU, 1.60 GHz, 8 GB RAM. Python 3.10.12, Pyomo 6.6.1, Ipopt 3.11.1, and Pandas 1.5.3 were used among others.

3.1.1. Model generation results

The workflow is first tested *in silico* using synthetic data. We observe that the workflow successfully keeps and neglects the correct terms in the differential mass balance equation, includes the correct constitutive equations, and performs well in parameter estimation. Table 2 compares the true model with the best models identified by the

workflow. The first best model is a perfect reconstruction of the true model. For instance, the importance of the diffusion term $\frac{\partial j_i}{\partial z}$ is recognized by keeping the term in the system of equations and specifying it by including Fick's law as a constitutive equation, see (8. Similarly, the correct reaction order is identified, and the spatial dependency of the axial flow velocity is correctly neglected. The NRMSE between the data generated by the true model and the identified model is $\mathrm{NRMSE} = 2.598 \cdot 10^{-11}$ which can be attributed to numerical effects.

Table 2: A comparison of the best identified models with the true model (Equations 7-12), *in silico* case study). For each symbolic term in the differential mass balance, the actions leading to the true model are compared to the actions leading to the best models. Checkmarks indicate agreement with true model. ”N.A.” indicates that no decision had to be made on a term, as the term occurs in a constitutive equation not included in the model. The actions are abbreviated as N: Neglect variable, V: Substitute variable with known value from problem profile, P: Declare variable as to-be-estimated parameter, C: Substitute variable with constitutive equation (Section 2.2.1).

Symbolic Term	True model	1st best model	2nd best model	3rd best model	4th best model	Worst model
$\frac{\partial c}{\partial t}$	N	✓	✓	✓	✓	✓
v_z	V	✓	✓	✓	✓	✓
$\frac{\partial v}{\partial z}$	N	✓	✓	✓	✓	✓
$\frac{\partial j_i}{\partial z}$	C	✓	✓	N	✓	P
r_i	C	✓	✓	✓	✓	✓
D_z	V	✓	✓	N.A. (since neglecting $\frac{\partial j_i}{\partial z}$)	✓	N.A. (since fitting $\frac{\partial j_i}{\partial z}$)
r_p	C	✓	✓	✓	✓	✓
ν_{ip}	V	✓	✓	✓	✓	✓
k_p	C	✓	P	✓	✓	✓
λ_{ip}	$\{1,1,0\}$	✓	✓	✓	$\{1,2,0\}$	✓
k_p^{ref}	P	✓	N.A. (since fitting k_p)	✓	✓	N.A. (since fitting k_p)
E_p	V	✓	N.A. (since fitting k_p)	✓	✓	N.A. (since fitting k_p)
R	V	✓	N.A. (since fitting k_p)	✓	✓	N.A. (since fitting k_p)

T^{ref}	V	✓	N.A. (since fitting k_p)	✓	✓	N.A. (since fitting k_p)
T	V	✓	N.A. (since fitting k_p)	✓	✓	N.A. (since fitting k_p)
NRMSE [-]	N.A.	$2.598 \cdot 10^{-11}$	$5.813 \cdot 10^{-11}$	$8.468 \cdot 10^{-3}$	$5.116 \cdot 10^{-2}$	23.297

As the second-best model (NRMSE: $5.813 \cdot 10^{-11}$), the workflow identifies a model with high similarity to the true model that, however, does not include Arrhenius equation (Equation 11) as constitutive relationship for the rate constant k_p. Instead, k_p is declared a to-be-fitted parameter and found through parameter estimation. The high accuracy of this simpler model is explained by the absence of temperature changes over the length of the reactor. Fitting k_p instead of k_p^{ref} yields the same model, as all other terms in Arrhenius equation are constant.

The agent identifies two other models that achieve NRMSE values below 0.1. One corresponds to the optimal model but neglects the diffusion term as well as all related constitutive equations (NRMSE: $8.468 \cdot 10^{-3}$). This observation illustrates the influence of the diffusion term on the experimental results but also demonstrates that the hypothetical true reactor can still be accurately approximated with models that neglect diffusion. The fourth-best model corresponds to the first-best model with third-order reaction kinetics (NRMSE: $5.116 \cdot 10^{-2}$).

3.1.2. Influence of training policy

In the *in silico* case study, the impact of reinforcement learning becomes evident when analyzing the number of iterations needed until the true model (Equations 7-12) is identified. Note that in the experimental case study, the best-performing model is not known *a priori* and the workflow is repeated until the first model undercuts a chosen error threshold.

Figure 6 summarizes how different training policies result in faster or slower model identification. Specifically, Figure 6 compares three static and a dynamic training policy, see Equation 6, as well as for reward functions with and without a penalty term for equation complexity, see Equations 3, 4. The figure shows the probability of finding the optimal true model in a given number of iterations. The probability is obtained by executing the workflow 100 times and recording for each execution, how many iterations were necessary to identify the true model. Note that for high number of iteration, effectively corresponding to an exhaustive enumeration of the search space, the optimal model is always identified. This visualization is proposed in the work of Bassenne and Lozano-Durán (2019).

A comparison of the static ϵ strategies shows that the workflow performance has an optimum on the interval $\epsilon \in [1,0]$. The green curve in Figure 6 represents the behavior of

an agent with $\epsilon = 1$, i.e., an agent that takes random actions in every state and does not learn promising decisions through the results of previous iterations. Accordingly, even after 500 iterations only in 52% of the experiments, the correct model was discovered. However, when decreasing ϵ and thereby gradually enabling the agent to learn, model identification accelerates. For $\epsilon = 0.7$, which is found to be the optimal static epsilon for our use case, the probability of detecting the correct model after 500 iterations rises to 60%. A further decrease of ϵ leads to inhibition of the agent's exploration and impairs the performance.

The dynamic ϵ-training policy provides significant further improvements in workflow performance, see Figure 6. Gradually decreasing ϵ from one to zero corresponds to gradually increasing exploitation over exploration, as the agent takes random decisions less frequently. This training policy outperforms the static policies and achieves a model identification probability of 80% in fewer iterations (317 iterations). Compared to making random decisions, the reinforcement learning enhanced workflow increases the identification probability by a factor of 1.5 in 37% fewer iterations.

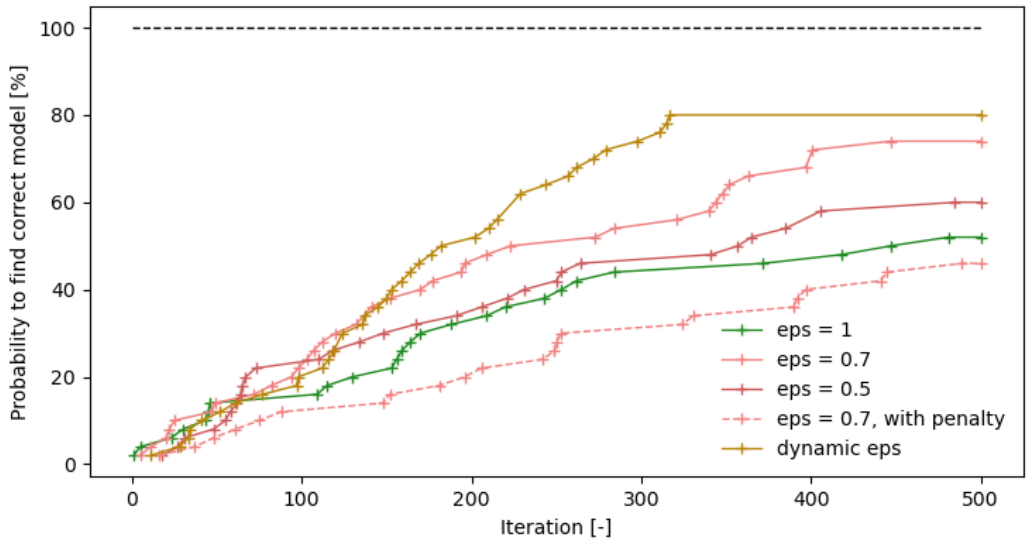


Figure 6: Performance of the agent under three static training policies ($\epsilon = 1, 0.7, 0.5$), a dynamic training policy ($\varepsilon = 1 \rightarrow 0$, linearly) and with and without a complexity penalty in the reward function (Equations 3, 4). If $\epsilon = 1$, the agent performs random decisions. The smaller ε, the more the agent favours exploitation over exploration. The figure shows the probability of finding the optimal true model in a given number of iterations. Note that for high number of iteration, effectively corresponding to an exhaustive enumeration of the search space, the optimal model is always identified. The figure can be read as 'After performing 317 iterations, the correct model was identified in 80% of the workflow executions when using the dynamic epsilon training policy.'

Finally, the effect of a complexity penalty in the reward function is investigated. Figure 6 includes the learning curve for an agent with constant $\epsilon = 0.7$ and a complexity penalty, see Equation 4. The modified agent takes significantly more iterations to discover the

optimal true model. The reason for this behavior is the high complexity of the true model in our case study compared to most other models in the search space. In the search process, smaller models are successfully prioritized over the more complex true model causing slower detection. In the Supplementary Materials (A.7), we provide additional evidence indicating that a complexity penalty in the reward function can be an effective measure to detect simpler models with similar accuracy earlier.

3.1.3. Influence of noisy data

The experimental data input into the workflow typically contains inherent levels of error and noise. To test the robustness of the workflow against such noise, we use the controlled setup of the *in silico* case study and perturb the input data with Gaussian noise at various standard deviations ($\sigma \in [0.001\ \mathrm{mol\ m^{-3}}, 0.1\ \mathrm{mol\ m^{-3}}] \approx [0.2\%, 20\%]$).

Figure 7 summarizes the sensitivity of the workflow performance to added noise. The four best-performing models in the *in silico* case study, see Table 2, all demonstrate errors of NRMSE $\leq 10\%$ for Gaussian noises up to $\sigma = 0.05\ \mathrm{mol\ m^{-3}} \approx 10\%$ which is above the common laboratory measurement uncertainties. Moreover, the order of the four best models does not change, i.e., the workflow still identifies the correct model as optimal model despite the addition of substantial synthetic noise.

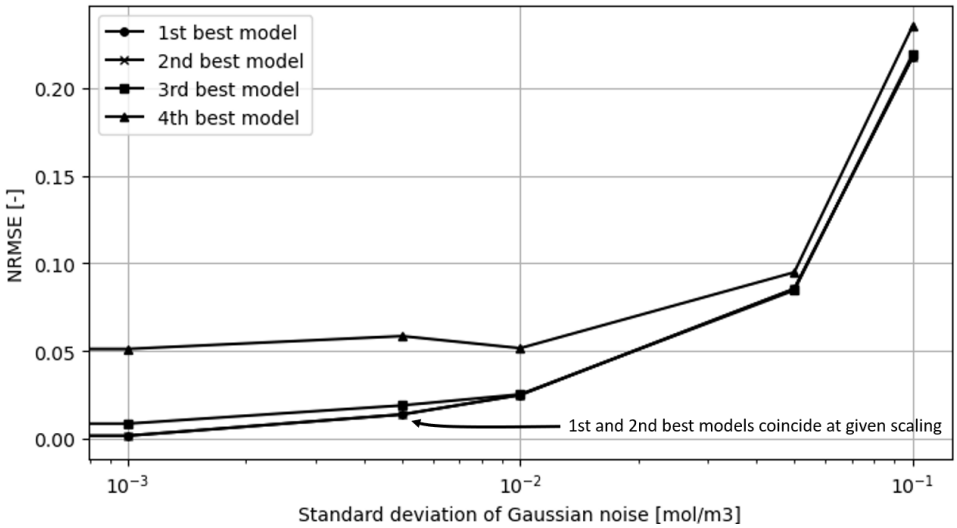


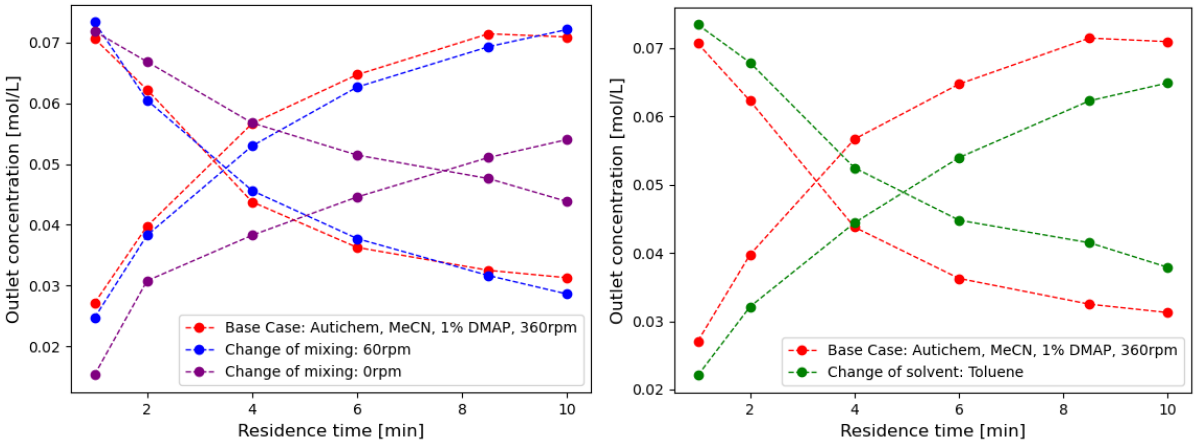
Figure 7: Sensitivity analysis of workflow results to Gaussian noise in the input data ($\sigma \in [0.001\ mol\ m^{-3}, 0.1\ mol\ m^{-3}] \approx [0.2\%, 20\%]$). The workflow continues to identify the true model successfully.

In summary, the *in silico* case study demonstrates that the developed workflow is capable of generating mechanistic models that closely match synthetic reactor data. Moreover, the case study shows how the reinforcement learning-based equation generation module significantly accelerates the model generation process compared to a trial-and-error approach and is robust against noise. In the Supplementary Materials (A.6.), an analysis of the runtime is provided.

3.2. Experimental case study

3.2.1. Data collection results

Figure 8(a) shows the effect of the rotation speed of the inner cylinder on the conversion. For higher revolutions, the conversion increases. This behavior can be attributed to the well-established vortex flow at high rotation speeds that increases mixing and prevents stagnant zones. Figure 8(b) visualizes the influence of the solvent choice on the reaction. When Toluene is used instead of MeCN, the reaction progresses more slowly.



(a) (b)

Figure 8: Experimental data from Taylor-Couette reactor for scenarios from Table 1. Concentration curves for reactant (2-bromophenyl)methanol (A) and product 2-bromobenzyl acetate (P). (a) Effect of mixing: Comparison of the base case (high mixing) with medium mixing and low mixing scenarios. (b) Effect of solvent: Comparison of the base case (solvent: MeCN) with a scenario where toluene is used.

3.2.2. Model generation using experimental data

In the experimental case study, the input data to the workflow is obtained from a Taylor-Couette reactor for four different operating condition scenarios.

For the base case scenario, the workflow identifies a set of candidate models where four models stand out in terms of accuracy (NRMSE$\leq 13.3\%$). The best model (NRMSE $= 2.4\%$) is presented in Table *3*. The agent neglects axial diffusion and chooses a third-order reaction rate with a fitted reaction constant of $k_p = 1.52 \cdot 10^{-5}\ \mathrm{m^6\ mol^{-2}\ s^{-1}}$. The second-, third- and fourth-best models closely align with the best model but either include the diffusion term, pick a first-order reaction, or a combination of both (NRMSE: $= 6.9, 8.1 \text{ and } 13.3\%$). The four models all represent reasonable approximations of the modelled system, and the second and third models, in particular, closely align with the dispersion model commonly proposed in the literature for modeling Taylor-Couette reactors. For validating the estimated reaction constant, the experimental data obtained

from a tube reactor is used, as the tube reactor can be assumed to exhibit an ideal PFR behavior. When fitting the data of the tube reactor to an ideal PFR model, a reaction constant of $k_p = 0.057\ \mathrm{m^6\ mol^{-2} s^{-1}}$ is obtained for the reaction, significantly higher than the reaction constant determined by the parameter estimation. This is a hint that the true reaction order may be more intricate and lie beyond the explored search space. In future implementations, insights like this can be used to systematically expand the equation library and search space.

Table 3: A comparison of the best models identified by the workflow for each of the four experimental scenarios in Table 1. "=" indicates identical choice as in the base case scenario. The actions are abbreviated as N: Neglect variable, V: Substitute variable with known value from problem profile, P: Declare variable as to-be-estimated parameter, C: Substitute variable with constitutive equation.

Symbolic Term	Base case	Case 2 (Med mixing)	Case 3 (Low mixing)	Case 4 (Toluene)
$\frac{\partial c}{\partial t}$	N	=	=	=
v_z	V	=	=	=
$\frac{\partial v}{\partial z}$	N	=	=	=
$\frac{\partial j_i}{\partial z}$	N	=	=	P
r_i	C	=	=	=
D_z	N.A. (since neglecting $\frac{\partial j_i}{\partial z}$)	=	=	=
r_p	C	=	=	=
ν_{ip}	V	=	=	=
k_p	$P\ (1.52 \cdot 10^{-5}\ m^6 mol^{-2} s^{-1})$	$P\ (1.33 \cdot 10^{-5}\ m^6 mol^{-2} s^{-1})$	$P\ (0.82 \cdot 10^{-5}\ m^6 mol^{-2} s^{-1})$	$P\ (0.67 \cdot 10^{-5}\ m^6 mol^{-2} s^{-1})$
λ_{ip}	$\{1,2,0\}$	=	=	=
k_p^{ref}	N.A. (since fitting k_p)	=	=	=
E_p	N.A. (since fitting k_p)	=	=	=
R	N.A. (since fitting k_p)	=	=	=
T^{ref}	N.A. (since fitting k_p)	=	=	=
T	N.A. (since fitting k_p)	=	=	=

NRMSE [-]	0.024	0.019	0.044	0.053

When providing the workflow with the concentration data of scenarios 2-4 (Table 1) as input data, new models are found that better approximate the Taylor-Couette system at the changed operation conditions. The updated models are summarized in Table *3*. In particular, when reducing the rotating rate of the mixer from 360 rpm to 60 rpm and 0 rpm, the workflow updates the model by estimating lower kinetic constants for the reaction ($k_{p,60rpm} = 1.33 \cdot 10^{-5}\ \mathrm{m^6\ mol^{-2}\ s^{-1}}$, $k_{p,60rpm} = 0.82 \cdot 10^{-5}\ \mathrm{m^6\ mol^{-2}\ s^{-1}}$). Domain knowledge, however, suggests that the slower reaction progress due to reduced mixing should be reflected in modifications to the axial diffusion term of the model rather than changes to the reaction constant, which is typically independent of mixing. This highlights two critical aspects of model interpretability. First, the observed behavior indicates that multiple models and combinations of parameter values may yield excellent fits to the data, underlining the need for structural identifiability analysis prior to model fitting (cf. test module in Section 2.1.2). While this feature lies beyond the current scope, we refer to Massonis et al. (2021) as a valuable foundation for future work. Second, interpretability can be enhanced by incorporating domain knowledge into the agent's reward function to discourage physically implausible model adjustments. In Part II of this study, we demonstrate this approach through reward shaping, guiding the agent toward model topologies that align with known reactor characteristics (Laub et al., 2025).

In summary, the experimental case study shows that the workflow is capable of generating realistic reactor models that emulate the concentration profiles with low error. The integrated parameter estimation gives a good indication of the order of magnitude of the parameters but is currently too imprecise to withstand validation by estimation equations from the literature for this reactor type. The scenario analysis demonstrates that the workflow effectively adapts the model to new operating conditions, while further integration of domain knowledge and incorporation of additional experimental data, such as residence time distributions, are needed to ensure complete physical plausibility. In contrast to models obtained using symbolic regression approaches such as in Narayanan et al. (2022), Kronberger et al. (2021), or Neumann et al. (2020), our models are derived from first-principles balance equations, thus abiding by conservation principles and known physical laws. Moreover, the workflow's modeling decisions, e.g., neglecting terms or substituting constitutive equations, directly correspond to assumptions traditionally made by the human modeler, increasing transparency and interpretability of the model. Furthermore, symbolic regression models are, while flexible and fast to evaluate, at risk of overfitting, particularly when data is noisy or sparse (De Franca, 2025) and unsuited for extrapolation (Kronberger et al., 2021). In Part II of this work, we furthermore demonstrate that the workflow is able to interpret new input data faster exploring closely related model structures first, whereas for symbolic regression

approaches small deviations in the input data commonly lead to significantly different model structures (Laub et al., 2025; Kammerer, 2024).

4. Conclusions

The automated development of insightful mechanistic models from experimental data is a challenging task that has been tackled for applications on the process and kinetic scale as well as outside of the chemical engineering domain. However, on the process unit level, and in particular for novel reactor designs, little research has been done and contributions in this field may accelerate the transition to widespread continuous chemical production. More precisely, further research can support reactor manufacturers in the urgently needed development of accurate and flexible digital twins for chemical reactors.

In this work, a modular workflow is presented that employs a Q-learning agent to efficiently derive mechanistic reactor models from concentration data. Internally, systems of equations are represented as a postfix sequence of python objects that the agent operates on. In an *in silico* case study, we find that the workflow is capable of identifying the correct model underlying the data with high accuracy (NRMSE $= 2.6 \cdot 10^{-11}$). For a search space restricted to one-dimensional DAE models with one compartment, the reinforcement learning agent increases the model identification probability by a factor of 1.5 in 37% fewer iterations compared to exhaustive search. For larger search spaces, this effect is expected to be even more significant. The workflow identifies the optimal model with a probability of 80% in the first 500 iterations, taking less than 5 minutes to execute on a laptop computer. Moreover, a dynamic ϵ-policy with $\epsilon = 1 \rightarrow 0$ was found to result in the fastest model identification. In the experimental case study, the Taylor-Couette reactor is successfully set up and operated to obtain concentration curves for four different reactor configurations. When providing the workflow with outlet concentration curves of the Taylor-Couette reactor, plausible reactor models are suggested that fit the experimental data with an NRMSE of 2.4%. However, a scenario analysis shows that additional domain knowledge needs to be incorporated to reliably ensure the physical validity of the proposed models.

A key opportunity for improving the workflow lies in enhancing the compartmentalization module, which is tasked with identifying an optimal model topology on the continuum between one-compartment ideal reactors and CFD models. The compartmentalization module will allow testing the workflow on more complex reactor geometries and phase systems. For example, for the Taylor-Couette reactor, more accurate predictions are expected from multi-zonal models based on the presented findings from the literature (Richter et al., 2008). There also lies large potential in learning a mapping between the problem profiles and the state representation of the optimal models. In this way, the effect of small changes in the reactor geometry, the chemical system, or the operating conditions on the model structure could be predicted, thus further eliminating the need

for experiments. Both these improvements are tackled in Part II of this research (Laub et al., 2025). In addition, future research should also enhance the test module to prevent ill-posed models from entering into the time-consuming parameter estimation step, thus further accelerating the workflow. Finally, the integration of automated experiments into the workflow can be explored to support the agent's decision-making with tailored online experiments.

Supplementary Materials

Supplementary Materials contains: Detailed explanation of the workflow modules; Reverse shunting yard algorithm for parsing states to pyomo models; Additional results demonstrating the effect of the complexity penalty; Background information on Taylor-Couette reactors; Technical specifications of the experimental equipment; Details on the synthesis of 2-bromobenzyl acetate and the calibration of the HPLC analysis; Analysis of computation time. The code and data are available upon reasonable request.

Acknowledgements, Funding Sources, Competing Interests

The research team is grateful to Autichem Ltd for providing a reactor system for this study. The author thanks the chair of Process Systems Engineering at RWTH Aachen, the Innovation Center in Digital Molecular Technologies at the University of Cambridge, Wolfson College, as well as the ERASMUS+ program for this research opportunity. The author thanks Dr. Dogancan Karan, and Dr. Ahmad Khan for the fruitful discussions throughout this research project. This work was in part supported by EPSRC EP/W031019/1 "Bio-derived and Bio-inspired Advanced Materials for Sustainable Industries (VALUED)". The authors declare no competing interests.

CRediT author statement

MH: Conceptualization, Methodology, Software, Validation, Formal Analysis, Investigation, Visualization, Writing – Original Draft, Writing – Review & Editing; JZ: Conceptualization, Methodology, Validation, Formal Analysis, Investigation, Writing – Review & Editing, Supervision, Project administration; NS: Methodology, Validation, Investigation, Data Curation, Writing – Review & Editing; JL: Validation, Writing – Review & Editing; AL: Conceptualization, Resources, Writing – Review & Editing, Supervision, Project administration, Funding Acquisition.

References

Angelis, D., Sofos, F., Karakasidis, T.E., 2023. Artificial Intelligence in Physical Sciences: Symbolic Regression Trends and Perspectives. Arch. Comput. Methods Eng. https://doi.org/10.1007/s11831-023-09922-z.

Bassenne, M., Lozano-Durán, A., 2019. Computational model discovery with reinforcement learning. arXiv preprint. http://arxiv.org/pdf/2001.00008v1.

Bouchon, F., Bergel, A., Filali, A., Bouchez, T., Fayolle, Y., 2023. First electrochemical Couette-Taylor reactor for studying the influence of transport phenomena on electrochemical kinetics. Chem. Eng. Sci. 281, 119103. https://doi.org/10.1016/j.ces.2023.119103.

Browne, C.B., Powley, E., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfshagen, P., 2012. A Survey of Monte Carlo Tree Search Methods. IEEE Trans. Comput. Intell. AI Games. 4, 1, 1–43. https://doi.org/10.1109/TCIAIG.2012.2186810.

Capaldo, L., Wen, Z., Noël, T., 2023. A field guide to flow chemistry for synthetic organic chemists. Chem. Sci. 14, 16, 4230–4247. https://doi.org/10.1039/d3sc00992k.

Carvalho Servia, M., Sandoval, I.O., Hii, K.K., Hellgardt, K., Zhang, D., Del Antonio Rio Chanona, E., 2024. The automated discovery of kinetic rate models – methodological frameworks. Digit. Discov. 3, 5, 954–968. https://doi.org/10.1039/D3DD00212H.

De Franca, F.O., 2025. Alleviating Overfitting in Transformation-Interaction-Rational Symbolic Regression with Multi-Objective Optimization. arXiv preprint. https://doi.org/10.48550/arXiv.2501.01905 .

Fantke, P., Cinquemani, C., Yaseneva, P., Mello, J., Schwabe, H., Ebeling, B., Lapkin, A.A., 2021. Transition to sustainable chemistry through digitalization. Chem. 7, 11, 2866–2882. https://doi.org/10.1016/j.chempr.2021.09.012.

Fritzson, P., 2020. The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development. Model. Identif. Control. 41, 4, 241–295. https://doi.org/10.4173/mic.2020.4.1.

GAMS Development Corporation, 2021. General Algebraic Modeling System v36.1. https://www.gams.com/.

Haag, J., Gentric, C., Lemaitre, C., Leclerc, J.-P., 2018. Modelling of Chemical Reactors: From Systemic Approach to Compartmental Modelling. Int. J. Chem. React. Eng. 16, 8. https://doi.org/10.1515/ijcre-2017-0172.

Horn, E.J., Rosen, B.R., Chen, Y., Tang, J., Chen, K., Eastgate, M.D., Baran, P.S., 2016. Scalable and sustainable electrochemical allylic C-H oxidation. Nature. 533, 7601, 77–81. https://doi.org/10.1038/nature17431.

Jourdan, N., Neveux, T., Potier, O., Kanniche, M., Wicks, J., Nopens, I. et al., 2019. Compartmental Modelling in chemical engineering: A critical review. Chem. Eng. Sci. 210, 115196. https://doi.org/10.1016/j.ces.2019.115196.

Kammerer, L., Kronberger, G., Winkler, S., 2024. Bias and Variance Analysis of Contemporary Symbolic Regression Methods. Appl. Sci. 14(23), 11061. https://doi.org/10.3390/app142311061.

Khan, A., Lapkin, A.A., 2020. Searching for optimal process routes: A reinforcement learning approach. Comput. Chem. Eng. 141, 107027. https://doi.org/10.1016/j.compchemeng.2020.107027.

Khan, A.A., Lapkin, A.A., 2022. Designing the process designer: Hierarchical reinforcement learning for optimisation-based process design. Chem. Eng. Process. Process Intensif. 180, 108885. https://doi.org/10.1016/j.cep.2022.108885.

Klise, K.A., Nicholson, B.L., Staid, A., Woodruff, D.L., 2019. Parmest: Parameter Estimation Via Pyomo. Comput. Aided Chem. Eng. 47, 41–46. https://doi.org/10.1016/B978-0-12-818597-1.50007-2.

Kronberger, G., De Franca, F.O., Burlacu, B., Haider, C., Kommenda, M., 2021. Shape-Constrained Symbolic Regression - Improving Extrapolation with Prior Knowledge. Evol. Comp. 30(1), 75-98. https://doi.org/10.1162/evcoa00294.

Lample, G., Charton, F., 2019. Deep Learning for Symbolic Mathematics. arXiv preprint. http://arxiv.org/pdf/1912.01412v1.

Laub, J.-F., Zhang, J., Heyer, M., Lapkin, A.A., 2025. Automated Generation of Mechanistic Models for Chemical Process Digital Twins using Reinforcement Learning. Part II: Compartmentalization and Learning-Based Recalibration. ChemRxiv preprint. https://doi.org/10.26434/chemrxiv-2025-83x6v

Lubin, M., Dowson, O., Garcia, J.D., Huchette, J., Legat, B., Vielma, J.P., 2023. JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. Math. Program. Comput. 15, 581-589. https://doi.org/10.1007/s12532-023-00239-3.

Luo, M., Liu, L., 2018. Automatic Derivation Of Formulas Using Reforcement Learning. arXiv preprint. http://arxiv.org/pdf/1808.04946v1.

Mahecha-Botero, A., Grace, J.R., Elnashaie, S.S.E.H. and Lim, C.J., 2007. A Comprehensive Approach to Reaction Engineering. Int. J. Chem. React. Eng. 5,1. https://doi.org/10.2202/1542-6580.1399.

Massonis, G., Banga, J.R., Villaverde, A.F., 2021. AutoRepar: a method to obtain identifiable and observable reparametrizations of dynamic models with mechanistic insights. Int. J. Robust Nonlinear Control. 33, 5039-5057. https://doi.org/10.1002/rnc.5887.

Matsumoto, M., Masuda, H., Hubacz, R., Horie, T., Iyota, H., Shimoyamada, M., Ohmura, N., 2021. Enzymatic starch hydrolysis performance of Taylor-Couette flow reactor with ribbed inner cylinder. Chem. Eng. Sci. 231, 116270. https://doi.org/10.1016/j.ces.2020.116270.

Mazyavkina, N., Sviridov, S., Ivanov, S., Burnaev, E., 2021. Reinforcement learning for combinatorial optimization: A survey. Comput. Oper. Res. 134, 105400. https://doi.org/10.1016/j.cor.2021.105400.

Narayanan, H., Bournazou, M.N.C., Gosalbez, G.G., Butte, A., 2022. Functional-Hybrid modeling through automated adaptive symbolic regression for interpretable mathematical expressions. Chem. Eng. J. 430, 133032. https://doi.org/10.1016/j.cej.2021.133032.

Neumann, P., Cao, L., Russo, D., Vassiliadis, V.S., Lapkin, A.A., 2020. A new formulation for symbolic regression to identify physico-chemical laws from experimental data. Chem. Eng. J. 387, 123412. https://doi.org/10.1016/j.cej.2019.123412.

Nicholson, B., Siirola, J.D., Watson, J.-P., Zavala, V.M., Biegler, L.T., 2018. pyomo.dae: a modeling and automatic discretization framework for optimization with differential and algebraic equations. Math. Program. Comput. 10, 2, 287-223. https://doi.org/10.1007/s12532-017-0127-0.

Petersen, B.K., Landajuela, M., Mundhenk, T.N., Santiago, C.P., Kim, S.K., Kim, J.T., 2021. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. Int. Conf. Learn. Represent. http://arxiv.org/pdf/1912.04871v4.

Plutschack, M.B., Pieber, B., Gilmore, K., Seeberger, P.H., 2017. The Hitchhiker's Guide to Flow Chemistry. Chem. Rev. 117, 18, 11796–11893. https://doi.org/10.1021/acs.chemrev.7b00183.

Politano, F., Oksdath-Mansilla, G., 2018. Light on the Horizon: Current Research and Future Perspectives in Flow Photochemistry. Org. Process Res. Dev. 22, 9, 1045–1062. https://doi.org/10.1021/acs.oprd.8b00213.

Preisig, H.A., 2010. Constructing and maintaining proper process models. Comput. Chem. Eng. 34, 9, 1543–1555. https://doi.org/10.1016/j.compchemeng.2010.02.023.

Richter, O., Hoffmann, H., Kraushaar-Czarnetzki, B., 2008. Effect of the rotor shape on the mixing characteristics of a continuous flow Taylor-vortex reactor. Chem. Eng. Sci. 63, 13, 3504–3513. https://doi.org/10.1016/j.ces.2008.04.003.

Schrimpf, M., Esteban, J., Warmeling, H., Färber, T., Behr, A., Vorholt, A.J., 2021. Taylor-Couette reactor: Principles, design, and applications. AIChE J. 67, 5. https://doi.org/10.1002/aic.17228.

Seidenberg, J.R., Khan, A.A., Lapkin, A.A., 2023. Boosting autonomous process design and intensification with formalized domain knowledge. Comput. Chem. Eng. 169, 108097. https://doi.org/10.1016/j.compchemeng.2022.108097.

Stegehake, C., Riese, J., Grünewald, M., 2018. Aktueller Stand zur Modellierung von Festbettreaktoren und Möglichkeiten zur experimentellen Validierung. Chem. Ing. Tech. 90, 11, 1739–1758. https://doi.org/10.1002/cite.201800130.

Stops, L., Leenhouts, R., Gao, Q., Schweidtmann, A.M., 2023. Flowsheet generation through hierarchical reinforcement learning and graph neural networks. AIChE J. 69, 1, e17938. https://doi.org/10.1002/aic.17938.

Wang, C., Chen, W., Wei, J., Shi, W.-D., Yan, W.-C., 2024. Integrated Taylor–Couette Reactor Model for Heterogeneous Photocatalytic Degradation of AO7. Ind. Eng. Chem. Res. 63, 32, 14052–14063. https://doi.org/10.1021/acs.iecr.4c01479.