Digital Chemical Engineering 17 (2025) 100276

Contents lists available at ScienceDirect

Digital Chemical Engineering

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

MIDDoE: An MBDoE *Python* package for model identification, discrimination, and calibration

Z. Tabrizi [a,b], E. Barbera [a], W.R. Leal da Silva [b], F. Bezzo [a,*]

[a] CAPE-Lab – Computer-Aided Process Engineering Laboratory, Department of Industrial Engineering, University of Padova, via Marzolo 9, 35131 Padova PD, Italy
[b] FLSmidth Cement A/S, Green Innovation, Denmark

ARTICLE INFO

Keywords:
MBDoE
Model identification
Design of experiments
Model discrimination
Estimability analysis
Open-source software

ABSTRACT

Mathematical modelling plays a critical role in the design, optimisation, and control of dynamic systems in the process industry. While mechanistic models offer strong explanatory and predictive power, their effectiveness depends on informed model selection and precise parameter calibration. Model-based design of experiments (MBDoE) provides a framework for addressing these challenges by designing experiments that accelerate model discrimination and parameter precision tasks. However, its practical application is frequently constrained by fragmented digital tools that lack integration and make MBDoE implementation a task for expert users. To address that – thus supporting the widespread use of MBDoE – *MIDDoE*, a modular and user-friendly *Python*-based framework centred on MBDoE is introduced. *MIDDoE* supports both model discrimination and parameter precision design strategies, incorporating physical constraints and non-convex design spaces. To provide a comprehensive MBDoE digital tool, the framework integrates numerical techniques such as Global Sensitivity Analysis, Estimability Analysis, parameter estimation, uncertainty analysis, and model validation. Its architecture decouples simulation from analysis, enabling compatibility with both built-in and external simulators, which allows *MIDDoE* to be applied across different systems. *MIDDoE* practical application is demonstrated through two case studies in bioprocess and pharmaceutical systems for model discrimination and parameter precision tasks.

1. Introduction

Mechanistic models are cornerstones in the development, optimisation, and control of engineered systems, offering predictive capabilities that extend beyond experimental observations (Buede, 2024; Pistikopoulos et al., 2021). Based on first principles, their reliability depends on a representative model structure and precisely estimated parameters. However, achieving predictive reliability is challenging: sparse data, parameter correlations, and structural ambiguities often compromise early-stage modelling. These limitations highlight the need for targeted data acquisition and rigorous model identification strategies to reduce uncertainty and improve predictive performance, a challenge that calls for Model-Based Design of Experiments (MBDoE).

MBDoE is tailored for mechanistic models, explicitly accounting for physical laws and meaningful parameters (Franceschini and Macchietto, 2008). It frames experiment planning as an optimisation problem, aiming to maximise the information content of data – either to increase the predictive divergence between candidate models and accelerate model discrimination (MBDoE-MD) (Hunter and Reiner, 1965), or to improve the precision of parameter estimates (MBDoE-PP) (Espie and Macchietto, 1989). This approach proved effective in reducing experimental burden, material usage, and measurement costs, while maximising mechanistic insight and valuable in applications involving complex, nonlinear dynamic systems, where experimentation is resource-intensive (Geremia et al., 2026).

Despite these demonstrated benefits, MBDoE remains a niche methodology in practice. Its application is limited by the need for computational literacy and familiarity with optimisation strategies. Additionally, the lack of accessible and physics-aware digital tools limits broader implementation among experimentalists and industrial practitioners (Geremia et al., 2026). Moreover, the practical effectiveness of MBDoE depends on its application following a rigorous pre-experimental analysis – particularly in terms of parameter significance and estimability – which requires holistic workflows for parameter ranking and identifiability assessment prior to experiment design.

Transforming MBDoE from a niche approach into a widely practical methodology requires more than simply addressing an optimization problem: it demands integrated workflows that begin with Global

* Corresponding author.
E-mail address: fabrizio.bezzo@unipd.it (F. Bezzo).

https://doi.org/10.1016/j.dche.2025.100276

Available online 22 November 2025

Sensitivity Analysis (GSA) to identify influential inputs (Saltelli et al., 2006) and Estimability Analysis (EA) to determine which parameters can be reliably estimated to assist MBDoE in correctly assigning the explorative budget (Cobelli and DiStefano, 1980). These steps are critical in nonlinear dynamic systems, where parameter identifiability may vary across the design space, and uninformed experiments risk targeting insensitive or weakly identifiable parameters. For an effective MBDoE use, the method must be embedded within a holistic modelling workflow that integrates initial diagnostic steps with complementary tasks such as parameter estimation, uncertainty analysis, and validation. Digital tools that are physics-aware, constraint-capable, and flexible enough to adjust to real-world process specifications are also necessary for the practical implementation of such a comprehensive MBDoE usage, particularly in industrial environments (Geremia et al., 2026).

While the conceptual foundation of MBDoE is well established, available software remain limited. In the area of MBDoE-MD, digital implementations are rare. A notable exception is *GPdoemd* (Olofsson et al., 2019), an open-source *Python* package that employs Gaussian Process (GP) surrogates to represent mechanistic models as black boxes, enabling gradient-free optimisation (Harris et al., 2020; Olofsson et al., 2019). It supports a variety of divergence-based design criteria – from Hunter–Reiner to Jensen–Rényi divergence – and includes modules for parameter estimation, model selection, and post-processing (Rényi and Renyi, 1965; Vanlier et al., 2014). However, some issues are not solved yet, including the absence of sampling time optimisation, limited support for Control Vector Parameterisation (CVP), insufficient enforcement of physical constraints, and elevated uncertainty in GP predictions for highly nonlinear systems.

In contrast, DoE-SINDy (Lyu and Galvanin, 2025) consists in a recently introduced Python-based framework that targets automated kinetic model discovery when first-principles formulations are unavailable or uncertain as a complementary problem. It uses the Sparse Identification of Nonlinear Dynamics (SINDy) approach to construct interpretable differential equations directly from time-series data (Brunton et al., 2016). The current version iteratively improves model quality by augmenting the training dataset with new experiments and validating structural and statistical adequacy at each step. It does not yet include MBDoE capabilities, but these are expected to be added in future work.

In contrast, several tools have been developed to support MBDoE-PP. At present, the commercial platform *gPROMS* by Siemens (https://www.siemens.com/global/en/products/automation/industry-software/gproms-digital-process-design-and-operations.html) offers the most comprehensive implementation, supporting various optimality criteria, constant and linear piecewise CVPs, and advanced solvers for nonlinear and non-convex problems. However, as a proprietary software, it may restrict user customisation (which can be a drawback when adapting to experimental constraints), lacks support for MBDoE-MD, and does not offer features for defining uncontrollable and forced control regions, and signal perturbation constraints.

EFCOSS, Pydex, and *PYOMO.DOE* are the three prominent open-source *Python* tools that support MBDoE-PP. *EFCOSS* (Rasch and Bücker, 2010) is a modular framework based on the Cobra platform, enabling distributed coupling between simulators and optimisers. Despite its flexibility, it requires considerable programming skills from the user, lacks support for alternative MBDoE-PP criteria, and delegates data handling and post-processing to the user. *Pydex* (Kusumo et al., 2022) employs a continuous-effort approach that improves robustness and mitigates non-convexity, but may compromise optimality where sequential updating is feasible, or may demand increased computational resources for larger design spaces. *PYOMO.DOE* (Wang and Dowling, 2022) integrates experiment design within the *PYOMO* algebraic modelling framework and supports differential-algebraic systems (DAEs) through *PYOMO.DAE*, along with parameter estimation via *PYOMO.Parmest* (Klise et al., 2019; Nicholson et al., 2018). *PYOMO* has been extended to support external solvers and to interface with external models, including input-output (GreyBox) models via PyNumero (Laky et al., 2022; Rodriguez et al., 2020). However, setting up such interfaces may still be challenging for inexperienced users.

Although each tool offers distinct capabilities, the most complete ecosystems are currently found in *gPROMS* and *PYOMO.DOE*, since they partially integrate several elements of a model identification workflow. Both provide parameter estimation, uncertainty analysis, and post-processing capabilities, but neither includes built-in EA or support for MBDoE-MD. *gPROMS* includes built-in GSA and proprietary solvers, whereas *PYOMO.DOE* requires external solvers for optimisation and simulation tasks, as well as third-party libraries – such as SALib (Iwanaga et al., 2022) – to perform GSA. Both frameworks are tightly coupled to their native environments which can add complexity in enforcing advanced constraints within the design space. This is a challenge for users with limited programming experience and increases the computational overhead required for model integration and tool interoperability.

To overcome the shortcomings of current tools in offering a comprehensive platform for MBDoE, an open-source *Python* package called *MIDDoE* (Model-(based) Identification, Discrimination, and Design of Experiments) is presented here. *MIDDoE*, recently introduced in a preliminary form for parameter estimation (Tabrizi et al., 2025), integrates the core components of a model identification workflow required to support MBDoE in dynamic systems. It includes GSA and EA for early-stage evaluation of the model structure and design space, and also includes modules for parameter estimation, uncertainty analysis, validation, and post-analysis. These features collectively support the core functionalities of MBDoE for model discrimination and parameter precision within a single, cohesive, open-source, and common *Python* environment. *MIDDoE* adopts a modular architecture for identification process organised into kernel, logic, and client conceptual layers, as illustrated in Fig. 1.

In contrast to existing tools, *MIDDoE* aims at taking a step forward by not only integrating the full range of functionalities needed to support MBDoE at varying levels of user expertise, but also embedding a dedicated optimisation core. This core enables the incorporation of physical and operational constraints as well as flexible solver configurations for tackling non-convex optimisation problems. Additionally, the package facilitates seamless integration with external simulators – such as *gPROMS* – increasing its applicability across diverse scientific and engineering domains.

This article is structured as follows. Section 2, methods and mathematical framework, describes the theoretical basis of model identification as well as the numerical techniques implemented in *MIDDoE* to support it. Section 3, package architecture, outlines the modular structure of the software, along with the execution capabilities and technical specifications of the implemented methods. Section 4 presents two case studies are used to demonstrate the practical use of *MIDDoE* to address model identification problems in practical case studies. Finally, the conclusions summarise the core features, main findings, and future directions of this work in Section 5.

2. Methods and mathematical framework

MIDDoE integrates various numerical techniques. This section outlines the mathematical foundation upon which these techniques are implemented. Only the techniques implemented within *MIDDoE* are discussed.

2.1. Definition of dynamic models

The formulation is grounded in the theory of dynamic systems and adopts a general nonlinear parametric model structure, consistent with a multiple-input, multiple-output (MIMO) system, denoted as $M(\boldsymbol{\theta})$:

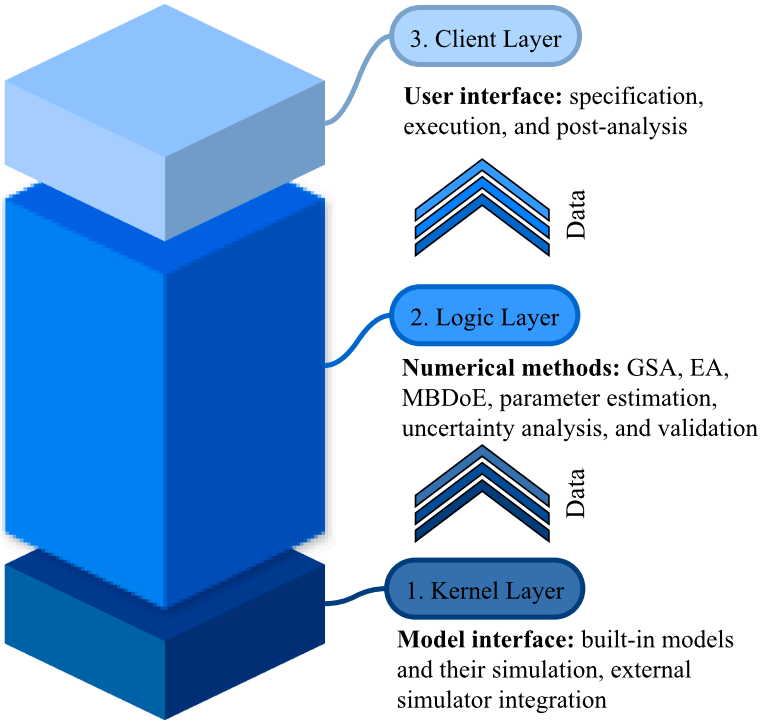

Fig. 1. Conceptual layers of *MIDDoE*: (1) the kernel layer handles model simulation (built-in or externally interfaced) and internal data flow; (2) the logic layer implements MBDoE alongside essential identification steps including GSA, EA, parameter estimation, uncertainty analysis, and validation; (3) the client layer enables user-level configuration, execution, and post-analysis.

$$M(\boldsymbol{\theta}, \mathbf{x}(t), \mathbf{u}(t), \mathbf{z}(t), \mathbf{w}) : \begin{cases} f(\dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{u}(t), \mathbf{z}(t), \mathbf{w}, \boldsymbol{\theta}) = 0 \\ g(\mathbf{x}(t), \mathbf{u}(t), \mathbf{z}(t), \mathbf{w}, \boldsymbol{\theta}) = 0 \\ y(t) = h(\mathbf{x}(t), \mathbf{u}(t), \mathbf{z}(t), \mathbf{w}, \boldsymbol{\theta}) \end{cases} \quad (1)$$

- $\boldsymbol{\theta} \in \mathbb{R}^{N_\theta}$: Vector of model parameters to be estimated, indexed by the set $\{1, ..., N_\theta\}$.
- $\mathbf{x}(t) \in \mathbb{R}^{N_x \times 1}$: Time-variant state variables governed by the differential equation f, represented over time as a matrix $\mathbf{X} \in \Re^{N_x \times N_t}$.

This formulation is characterised by the following components:

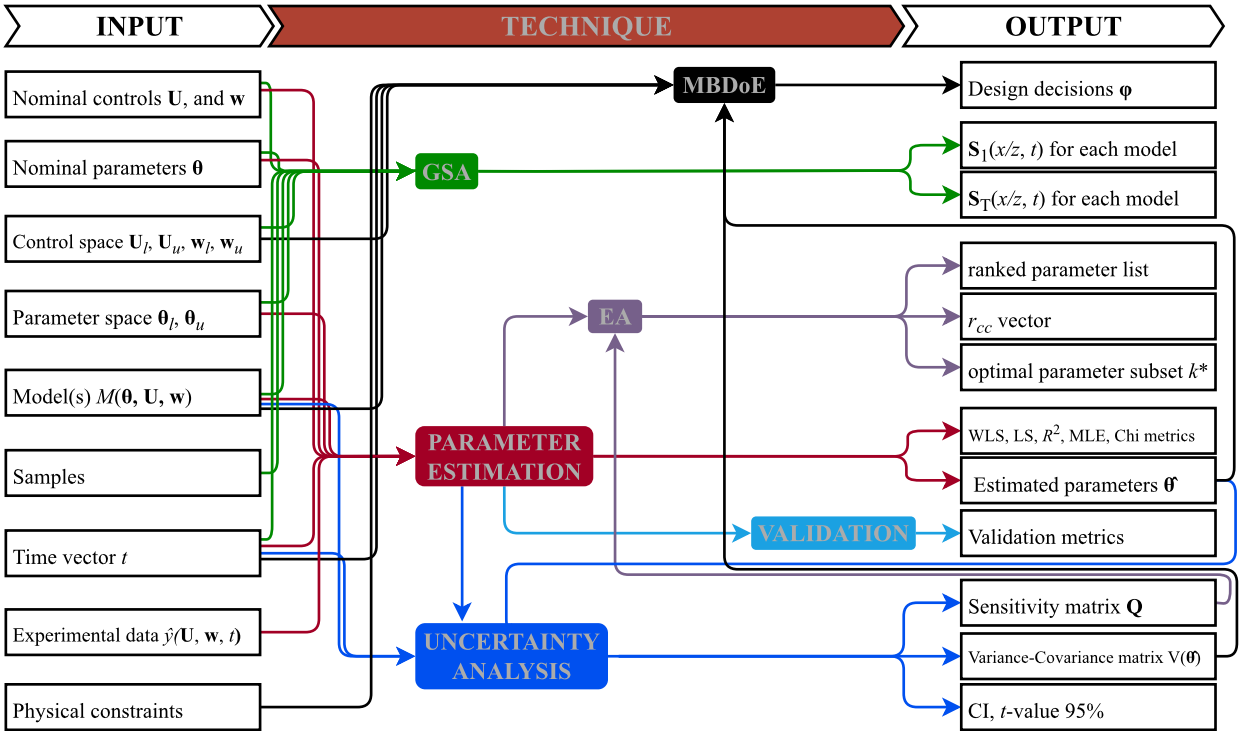

Fig. 2. Input/output structure of the core techniques implemented in the *MIDDoE* logic layer, including: (1) GSA; (2) parameter estimation and uncertainty analysis; (3) EA; (4) MBDoE; and (5) model validation.

- $\mathbf{z}(t) \in \mathbb{R}^{N_z \times 1}$: Time-variant state variables governed by the algebraic equation g, represented over time as a matrix $\mathbf{Z} \in \Re^{N_z \times N_t}$.
- $\mathbf{u}(t) \in \mathbb{R}^{N_u \times 1}$: Time-dependent control inputs (manipulated variables), represented over time as a matrix $\mathbf{U} \in \Re^{N_u \times N_t}$.
- $\mathbf{w} \in \mathbb{R}^{N_w}$: Time-invariant control inputs.
- $\mathbf{y}(t) \in \mathbb{R}^{N_r \times 1}$: Measured state variables, indexed by the set $\{1,\ldots,N_r\}$, and related to the model predictions by: $\mathbf{y}(t) = \hat{\mathbf{y}}(t) + \boldsymbol{\epsilon}(t)$ where $\hat{\mathbf{y}}(t)$ is the corresponding predicted value and $\boldsymbol{\epsilon}(t)$ represents the measurement noise or modelling error. It is represented over time as a matrix $\mathbf{Y} \in \Re^{N_r \times N_t}$.
- $t \in \Re^{N_t \times 1}$: Time vector encompassing all control and measurement time points.

With $M(\boldsymbol{\theta})$ representing the generic structure of candidate models, the implemented techniques – forming the backbone of the previously introduced logic layer (Fig. 1) for identification of such systems – are outlined in Fig. 2. This flowchart illustrates the theoretical inputs and outputs of the methods supported by *MIDDoE*, which are subsequently detailed in the following sections.

2.2. Global sensitivity analysis

GSA methods are developed to evaluate the influence of model inputs – parameters $\boldsymbol{\theta}$, time-variant controls $\mathbf{u}(t)$, time-invariant controls $\mathbf{w}$ – on the state variable trajectories $\mathbf{x}(t)$ and $\mathbf{z}(t)$ across the entire feasible input space. These approaches account for nonlinear dependencies and higher-order interactions, thereby offering more comprehensive and informative insights into system behaviour (Saltelli et al., 2010). Among GSA methods, Sobol's method (Sobol' IM, 1990) is one of the most robust and widely used variance-based GSA techniques. It enables the estimation of first-order (S_1) and total-order (S_T) Sobol indices for each model output across time and model instances, making it particularly suitable for complex, nonlinear, and dynamic systems.

This technique employs a Saltelli-type quasi–Monte Carlo strategy for global sensitivity quantification (Saltelli et al., 2010). The sampling approach is based on Sobol' sequences, which are deterministic low-discrepancy sequences in $[0,1]^d$ (where d is the number of input variables) that uniformly cover the unit hypercube. The sequences are scrambled using a mix of Left Matrix Scrambling (LMS) and a digital random shift, which is referred to as LMS+shift (Owen, 1998), in order to improve uniformity and mitigate structural correlations. This transforms the deterministic Sobol's sequence into a randomized quasi–Monte Carlo sample, preserving low-discrepancy characteristics while allowing for unbiased estimation and robust variance reduction. Further details on this transformation, as well as the evaluation of Sobol' first-order and total-effect indices ($\mathbf{S_1}$, and $\mathbf{S_T}$) are provided in the Supplementary Material S1.

As illustrated in Fig. 2, this technique requires the nominal values of fixed inputs, the sampling space of studied inputs, the time vector, the model(s), and the number of samples in order to compute the sensitivity indices for each response across time and model instances.

2.3. Parameter estimation and statistical metrics

Identification of model parameters $\boldsymbol{\theta}$ that best reproduce experimental observations, along with the evaluation of uncertainty in this stage, plays a pivotal role in guiding the model identification workflow. It forms the basis of MBDoE optimisation, and the metric to accept the adequacy and precision of identified model. Common cost functions employed in parameter estimation include Least Squares (LS), Weighted Least Squares (WLS), Maximum Likelihood Estimation (MLE), and Chi-Square (CS). These formulations are discussed in detail in the Supplementary Material S2.

After calibration of the candidate models, P-test is applicable, which quantifies the relative likelihood that each candidate model provides the best fit to the observed data. This method assigns a probability score P_i to each model i, reflecting its relative quality based on goodness-of-fit metrics.

The calculation is based on the Chi-Square values (χ^2) obtained after parameter estimation for each model. Models with lower χ^2 values are considered better descriptors of the system. The probability of the i-th model being the best among N_m candidates is defined following the formulation by (Galvanin et al., 2016):

$$P_i = 1 - \frac{1/\chi_i^2}{\sum_{i=1}^{N_M} 1/\chi_i^2} \cdot 100 \tag{2}$$

This normalised probability score allows direct comparison of model quality. A higher P_i indicates stronger evidence in favour of model i and supports effective model discrimination based on fit performance.

2.4. Uncertainty analysis and statistical metrics

Once the parameter estimates ($\widehat{\boldsymbol{\theta}}$) are obtained, the uncertainty in these estimates is quantified using either frequentist or asymptotic methods. The choice of method depends on the assumed model linearity and the degree of parameter identifiability. Regardless of the approach, the first step involves constructing a local sensitivity matrix $\mathbf{Q}_r$ for each measured response r, defined as:

$$\mathbf{Q}_r = \begin{bmatrix} \left.\frac{\partial \widehat{y}_r}{\partial \widehat{\theta}_1}\right|_{t_1} & \cdots & \left.\frac{\partial \widehat{y}_r}{\partial \widehat{\theta}_{N_\theta}}\right|_{t_1} \\ \vdots & \ddots & \vdots \\ \left.\frac{\partial \widehat{y}_r}{\partial \widehat{\theta}_1}\right|_{t_{N_{sp}}} & \cdots & \left.\frac{\partial \widehat{y}_r}{\partial \widehat{\theta}_{N_\theta}}\right|_{t_{N_{sp}}} \end{bmatrix} \tag{3}$$

where $\widehat{y}_{(r,t)}$ denotes model predictions, N_{sp} is the number of sampling points, and N_θ is the number of active parameters. This matrix lets construction of the Fisher information matrix $\mathbf{I}(\widehat{\boldsymbol{\theta}},\ \mathbf{U},\ \mathbf{w})$ as:

$$\mathbf{I}(\widehat{\boldsymbol{\theta}}, \mathbf{U}, \mathbf{w}) = \sum_{r}^{N_r} \sum_{r'}^{N_r} \widetilde{\sigma}_{(r,r')} \mathbf{Q}_r^T \mathbf{Q}_{r'} + \boldsymbol{\Sigma}_\theta(\widehat{\boldsymbol{\theta}})^{-1} \approx \sum_{r}^{N_r} \sum_{r'}^{N_r} \widetilde{\sigma}_{(r,r')} \mathbf{Q}_r^T \mathbf{Q}_{r'} \tag{4}$$

where prior information of parameters can be neglected (i.e., $\boldsymbol{\Sigma}_\theta(\widehat{\boldsymbol{\theta}})^{-1} \approx 0$), and $\widetilde{\sigma}_{(r,r')}$ are elements from the inverse of the variance-covariance matrix of measurements errors. The measurement noise structure across all responses is characterised by matrix $\boldsymbol{\Sigma}_y$:

$$\boldsymbol{\Sigma}_y = \begin{bmatrix} \sigma^2_{y_1 y_1} & \cdots & \sigma^2_{y_1 y_{N_r}} \\ \vdots & \ddots & \vdots \\ \sigma^2_{y_{N_r} y_1} & \cdots & \sigma^2_{y_{N_r} y_{N_r}} \end{bmatrix} \tag{5}$$

$\mathbf{I}$ serves two key purposes:

(i) Diagnosing the presence of sloppy parameters and ill-conditioning in the estimation problem, and selecting the uncertainty analysis method.
(ii) Enabling the approximation of the parameter variance–covariance matrix by inverting it ($\mathbf{V}_\theta = \mathbf{I}^{-1}$) if an asymptotic method is selected.

In particular, sloppy parameters correspond to directions in parameter space where the model output is only weakly sensitive, resulting in very small eigenvalues of $\mathbf{I}$. If the condition number of the Fisher matrix, defined as $\kappa(\mathbf{I}) = \lambda_{max}/\lambda_{min}$, exceeds a critical threshold (typically $\kappa > 10^3$), the model is considered sloppy and if this number is extremely high, the matrix is ill-conditioned and the inversion of it creates significant errors in evaluation of the variance and covariance terms (Gutenkunst et al., 2007). This indicates possible non-identifiability of parameters with lower eigenvalues and reduced reliability of asymptotic

uncertainty estimates.

In cases of strong nonlinearity or poor identifiability, a frequentist approach such as bootstrapping is preferred for more robust uncertainty quantification (Efron, 1979). In such a technique, bootstrap resampling generates multiple synthetic datasets by drawing with replacement from the original data. Parameter estimation is repeated on each resampled set, and the resulting distribution of estimates reflects the underlying variability without requiring assumptions about linearity or normality.

The resulting covariance matrix $\mathbf{V}_\theta(\widehat{\boldsymbol{\theta}}, \mathbf{U}, \mathbf{w})$ from any of the discussed methods is used to compute Confidence Interval (CI_i) and t-value of estimated parameter $\widehat{\theta}_i$ at a confidence level of $(1-\alpha) = 95\%$:

$$CI_i = \sqrt{v_{\theta,ii}} \cdot t\left(\frac{1-\alpha}{2},\ DoF\right) \text{ for } i = 1,\ \ldots,\ N_\theta \tag{6}$$

$$t_i = \frac{\widehat{\theta}_i}{\sqrt{v_{\theta,ii}}},\ i = 1,\ \cdots,\ N_\theta \tag{7}$$

where $v_{\theta,ii}$ is the i-th diagonal element of $\mathbf{V}_\theta$, and $t\left(\frac{1-\alpha}{2},\ DoF\right)$ is the upper $\frac{(1-\alpha)}{2}$ critical value of a Student's t-distribution with $DoF = N_{sp} - N_\theta$ degrees of freedom.

The t-test provides a quantitative measure of estimation precision by statistically assessing whether each estimated parameter $\widehat{\theta}_i$ is significantly different from the parameter noise with a zero mean. This is verified by checking whether the computed t_i-value exceeds the reference threshold t^{ref}, corresponding to a Student t-value at the same confidence level (Galvanin, 2010).

When uncertainty is quantified using the Fisher information matrix, the analysis is inherently local because it relies on local sensitivities. In contrast, when uncertainty is assessed using bootstrapping, the procedure is global.

Fig. 2 summarises the required inputs for these techniques, including data, model structures, the time vector, control input values, and the feasible parameter space. These inputs are used to estimate parameters and evaluate fitting quality metrics, followed by quantification of uncertainty through the variance-covariance matrix, 95% CIs, and corresponding t-values.

2.5. Estimability analysis

Model parameters can be assessed in terms of their capability to be uniquely and precisely estimated by performing an identifiability analysis. It includes structural identifiability, which examines models under ideal conditions (infinite data, no noise), and practical identifiability (called estimability in some contexts), which considers real-world data limitations (Cobelli and DiStefano, 1980; Jacquez and Greif, 1985; Miao et al., 2011; Raue et al., 2009). Estimability analysis and parameter subset selection are key techniques to support MBDoE to assess the influence of parameters, and fix the ones that cannot be estimated. Among different EA methods, the orthogonalisation method has emerged as a powerful, systematic, and computationally efficient local technique for ranking and selecting estimable parameters (Wu et al., 2011; Yao et al., 2003). This method removes dependencies among parameters to ensure that each selected parameter contributes independently, and then, based on the influence on predictability, proposes the optimal active parameter trade-off (top-k* ranked parameters) between biases caused by fixing of parameters and uncertainty introduced by over fitting. These methods are further explained in detail in the Supplementary Material S3.

The input and output structure of these techniques is summarised in Fig. 2, where the scaled sensitivity matrix is used for parameter ranking, and the parameter estimation procedure – along with its required inputs – is employed for parameter subset selection.

2.6. Model-Based design of experiments

The MBDoE-MD and MBDoE-PP are central techniques to accelerate the model discrimination and calibration by designing more informative experiments. The core of each MBDoE task is an optimisation, which solves for the decision vector $\boldsymbol{\varphi} = [\mathbf{u}(\mathbf{t}), \mathbf{w}, \mathbf{t_{sp}}]$, representing time-variant, -invariant controls, as well as the sampling times of measured state variables.

MBDoE-MD implements the T-optimality criteria, which seeks to maximise the divergence between competing model predictions. Two of the most used formulations are the Hunter and Reiner (HR) (Hunter and Reiner, 1965), and an extension on Buzzi-Ferraris and Forzatti method (BFF) (Chen and Asprey, 2003). The general form of the T-optimal design problem is expressed as:

$$\boldsymbol{\varphi}_{OPT} = \arg\max_{\boldsymbol{\varphi} \in \boldsymbol{\Phi}} \sum_{k=1}^{N_{sp}} \sum_{l=1}^{N_m} \sum_{l'=1\pm1}^{N_m} T_{l,l'}(\boldsymbol{\varphi}, \widehat{\boldsymbol{\theta}}, \widehat{\boldsymbol{\theta}}', t)\ dt \tag{8}$$

$$T_{l,l'}(\boldsymbol{\varphi}, \widehat{\boldsymbol{\theta}}, \widehat{\boldsymbol{\theta}}', t) = (\widehat{y}_l(\boldsymbol{\varphi}, \widehat{\boldsymbol{\theta}}, t) - \widehat{y}_{l'}(\boldsymbol{\varphi}, \widehat{\boldsymbol{\theta}}', t))^T \times \mathbf{F}_{l,l',t}(\widehat{y}_l(\boldsymbol{\varphi}, \widehat{\boldsymbol{\theta}}, t) - \widehat{y}_{l'}(\boldsymbol{\varphi}, \widehat{\boldsymbol{\theta}}', t)) \tag{9}$$

where N_m denotes the number of competing models, $\widehat{\boldsymbol{\theta}}$ and $\widehat{\boldsymbol{\theta}}'$ are the respective preliminary estimated parameter vectors for models l, and l', $\boldsymbol{\Phi}$ defines the feasible design space, and $\mathbf{F}_{l,l',t}$ is a weight factor based on the uncertainties in observation and parameter estimation at different sampling times, $t \in \mathbb{R}^{N_{sp} \times 1}$ which is 1 for 'HR'. For the BFF, $\mathbf{F}_{l,l',t}$ incorporates model sensitivity and covariance terms, and is defined at each sampling time t as:

$$\mathbf{F}_{l,l',t} = \boldsymbol{\Sigma}_y + \mathbf{W}_{l,t} + \mathbf{W}_{l',t} \tag{10}$$

where $\boldsymbol{\Sigma}_y$ is the variance-covariance matrix of measurement errors, and $\mathbf{W_{l,t}}$ and $\mathbf{W}_{l',t}$ are representing the modeling error contributions, derived from the sensitivity of each model to its parameters. Each $\mathbf{W}_{l,t}$ is constructed as:

$$\mathbf{W}_{l,t} = \mathbf{Q}_{l,t}\ \boldsymbol{\Sigma}_{\theta,l}^{-1}\ \mathbf{Q}_{l,t}^T \tag{11}$$

where $\mathbf{Q}_{l,t}$ is the sensitivity matrix of model l with respect to its estimated parameters $\widehat{\boldsymbol{\theta}}_\mathbf{l}$ from the preliminary estimations, evaluated at time t:

$$\mathbf{Q}_{l,t} = \begin{bmatrix} \frac{\partial \widehat{y}_{l,1}(t,\ \widehat{\boldsymbol{\theta}}_\mathbf{1})}{\partial \widehat{\theta}_{l,1}} & \cdots & \frac{\partial \widehat{y}_{l,1}(t,\ \widehat{\boldsymbol{\theta}}_\mathbf{1})}{\partial \widehat{\theta}_{l,N_\theta}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \widehat{y}_{l,M}(t,\ \widehat{\boldsymbol{\theta}}_\mathbf{1})}{\partial \widehat{\theta}_{l,1}} & \cdots & \frac{\partial \widehat{y}_{l,M}(t,\ \widehat{\boldsymbol{\theta}}_\mathbf{1})}{\partial \widehat{\theta}_{l,N_\theta}} \end{bmatrix} \tag{12}$$

This formulation also applies analogously to $\mathbf{W}_{l',t}$ for model l'.

Conversely, MBDoE-PP aims to minimise parameter uncertainty by reducing the size of the confidence hyperellipsoid through optimisation of a scalar metric $\psi[\mathbf{V}_\theta(\widehat{\boldsymbol{\theta}}, \mathbf{U}, \mathbf{w})]$. The MBDoE-PP problem is formulated as:

$$\boldsymbol{\varphi}_{OPT} = \arg\min_{\boldsymbol{\varphi} \in \boldsymbol{\Phi}} \psi[\mathbf{V}_\theta(\boldsymbol{\theta}, \mathbf{U}, \mathbf{w})] \tag{13}$$

Several objective functions $\psi[\mathbf{V}_\theta(\widehat{\boldsymbol{\theta}}, \mathbf{U}, \mathbf{w})]$, commonly referred to as alphabetical criteria are widely used in this context and listed in Table 1 (Franceschini and Macchietto, 2008).

These formulations are described in detail with their geometric interpretation, advantages, and disadvantages in the Supplementary Material S4.

Fig. 2 summarises the inputs and outputs of this method, which relies on the estimated parameters and their associated variance–covariance matrix, along with the model(s), time vector, design space, and relevant physical constraints.

Table 1
MBDoE-PP alphabetical criteria.

Criterion	Definition	Mathematical form	Equation
D	Reducing the confidence region size by minimising the determinant of the $\mathbf{V}_{\boldsymbol{\theta}}(\widehat{\boldsymbol{\theta}}, \mathbf{U}, \mathbf{w})$.	$\psi_D = det\ [\mathbf{V}_{\boldsymbol{\theta}}(\widehat{\boldsymbol{\theta}}, \mathbf{U}, \mathbf{w})]$	(14)
A	Lowering the average variance across all parameter estimates by minimising the trace of $\mathbf{V}_{\boldsymbol{\theta}}(\widehat{\boldsymbol{\theta}}, \mathbf{U}, \mathbf{w})$.	$\psi_A = tr\ [\mathbf{V}_{\boldsymbol{\theta}}(\widehat{\boldsymbol{\theta}}, \mathbf{U}, \mathbf{w})]$	(15)
E	Reducing the variance of the most uncertain parameter by minimising the biggest eigenvalue of $\mathbf{V}_{\boldsymbol{\theta}}(\widehat{\boldsymbol{\theta}}, \mathbf{U}, \mathbf{w})$.	$\psi_E = \lambda_{\max}\ [\mathbf{V}_{\boldsymbol{\theta}}(\widehat{\boldsymbol{\theta}}, \mathbf{U}, \mathbf{w})]$	(16)
ME	Ensure uniform precision across parameters by minimises the condition number of the $\mathbf{V}_{\boldsymbol{\theta}}(\widehat{\boldsymbol{\theta}}, \mathbf{U}, \mathbf{w})$.	$\psi_{ME} = \kappa[\mathbf{V}_{\boldsymbol{\theta}}(\widehat{\boldsymbol{\theta}}, \mathbf{U}, \mathbf{w})]$	(17)

2.7. Model validation

To assess the reliability and generalisation capability of estimated parameters, leave-one-out cross-validation (LOOCV) procedure (Stone, 1974) is a widely used method. In each fold of this method, a part of data is withheld for validation, while the rest is used for model recalibration. The process is repeated, and performance metrics (e.g., R^2, mean square error) are compared for both calibration and validation sets. Fig. 2 summarises the demanded inputs and expected outputs, as the technique iteratively uses the parameter estimation and returns both calibration and validation predictive metrics.

3. Package architecture

3.1. Overview of the package architecture

MIDDoE, available at https://pypi.org/project/middoe and compatible with Python versions 3.9 and above, is designed to support the discussed model identification tasks through two service levels. The first targets end-users with limited programming experience, offering a wrapper-based execution pathway that follows a logical, stepwise sequence. The second provides a flexible interface for advanced users who wish to customise the logic of each method for complex experimental campaigns or methodological investigations. To ensure computational efficiency – particularly in inner optimisation loops and numerical solvers – *MIDDoE* adopts a modular procedural programming approach. This structure reduces runtime overhead by eliminating unnecessary object instantiations and attribute lookups, while retaining the structural benefits of object-oriented design, such as abstraction and modularity (Calder et al., 1994).

The following sections elaborate on the conceptual architecture of *MIDDoE* (Fig. 1) and describe the role of each executive module in the layered structure.

3.2. Logic layer

Fig. 3 illustrates a structured model identification workflow, incorporating the executive modules and their interdependencies. This application-oriented structure allows systematic execution of tasks such as GSA, parameter estimation, uncertainty analysis, EA, MBDoE and model validation. The workflow is initiated by specifying model structures and available experimental data. In the absence of initial observations, GSA (Fig. 3, Step 1) can be performed to explore the design

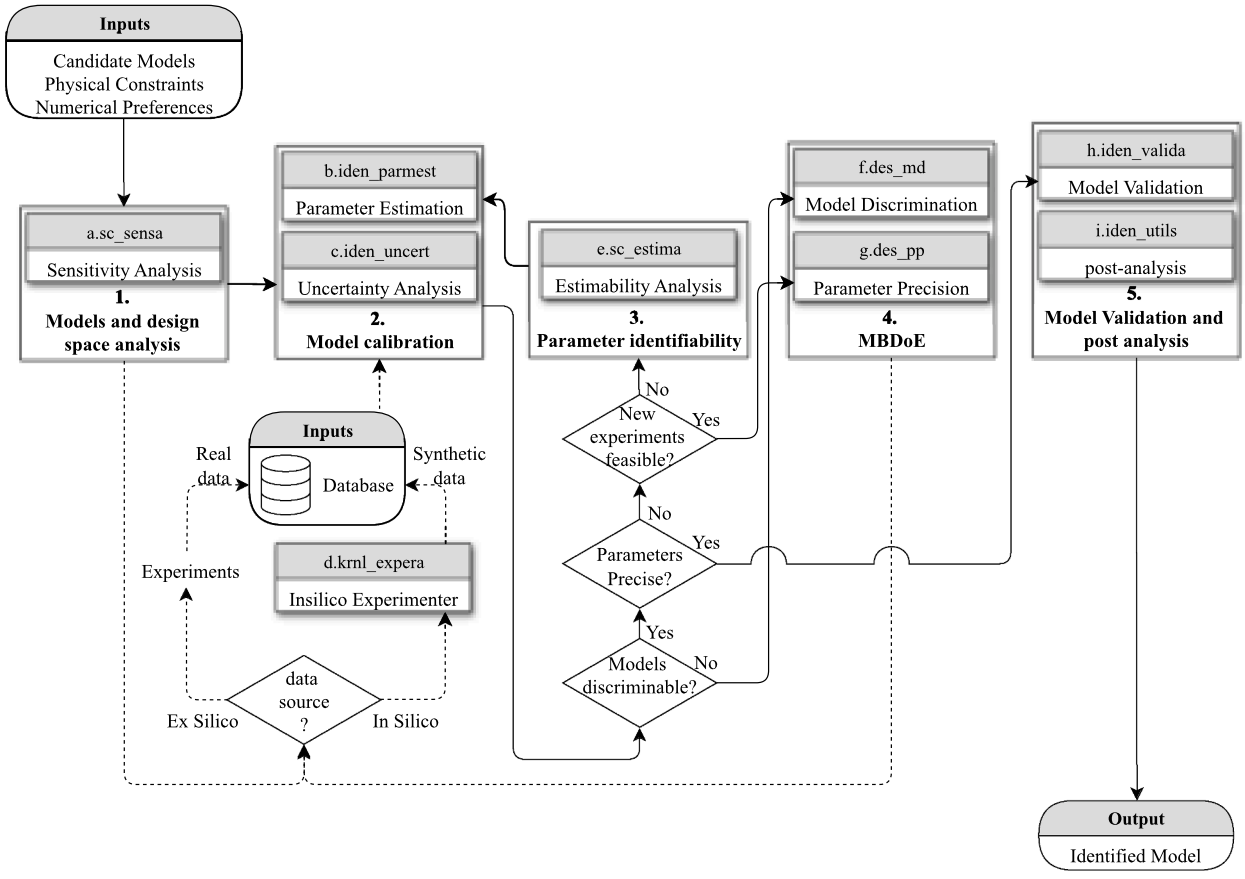

Fig. 3. Model identification workflow implemented by *MIDDoE*. The key identification steps are indicated by numbers, while the executive software modules are marked with alphabetic labels. Solid arrows represent the model flow, and dashed arrows represent the data flow.

space, inform the configuration of preliminary experiments, or rank the parameter significance.

Once measurements are available, preliminary model calibration and uncertainty quantification are performed (Fig. 3, Step 2). If multiple candidate models remain indistinguishable, MBDoE-MD is applied (Fig. 3, Step 4) to design experiments that maximise the divergence in their predictions. This process may be iterated alongside recalibration until a single model is preferred. If parameter uncertainty remains high, two complementary strategies are available:

(i) EA (Fig. 3, Step 3), which reduces model complexity by fixing poorly identifiable parameters.
(ii) MBDoE-PP (Fig. 3, Step 4), which designs experiments to improve parameter precision.

These strategies may be applied separately or in tandem. Once a single, well-defined model is identified, it is validated (Step 5) through cross-validation and post-analysed for reporting.

3.3. Client layer

Data, models, and settings are the three main inputs, which each *MIDDoE* module depends on. To guarantee compatibility with experimental protocols, data are either imported as Excel files or supplied as Pandas DataFrames. Models are implemented as *Python* functions or as interfaces to external simulators, with configuration options supplied via user-provided dictionaries.

Two fundamental dictionaries – `system` and `models`, which serve as configuration hubs across all modules in *MIDDoE*, regulate how models and data are interpreted. The `system` dictionary establishes the model internal structure, specifying the state variables, determining which are measured, and setting measurement constraints as needed. It also distinguishes between time-invariant and time-variant controllable inputs and classifies the output variables, each constrained within user-defined feasible domains. The `models` dictionary determines how the model is loaded, identifies the parameters involved along with their corresponding bounds, and specifies the model interface – whether internal, externally defined, or connected to third-party simulators – used to evaluate the model.

3.4. Kernel layer

MIDDoE is designed for the analysis of lumped dynamic systems (DAE systems), with time typically treated as the independent variable. However, for steady-state applications or models governed by alternative independent variables, this convention can be adapted accordingly. Furthermore, *MIDDoE* supports certain classes of models involving Partial Differential Algebraic Equations (PDAEs), provided that the additional independent variable can be coarsely discretised and each segment response treated as discretised individual outputs (e.g., this occurs in some solution methods of population balance models). This flexibility enables the package to accommodate a broad spectrum of process systems and mechanistic model structures.

MIDDoE supports simulation of MIMO systems, denoted as $\boldsymbol{M}(\boldsymbol{\theta})$, by referencing models through the `models` dictionary using one of four available methods:

1. As a built-in DAE model selected from the `krnl_models` module;
2. As an externally defined DAE available in the global namespace;
3. As a user-defined black-box model in *Python* (or wrapped around external simulators);
4. As a *gPROMS*-based model accessed through an integrated simulation interface.

Regardless of the simulation backend, all models must conform to a standardised interface. The model function must accept the following five inputs:

- `t`: A list of time points at which the model is evaluated;
- `y0`: The initial conditions for the differential state variables $\mathbf{x}(t)$;
- `tii`: A dictionary of time-invariant input controls representing $\mathbf{w}$;
- `tvi`: A dictionary of time-variant input controls $\mathbf{u}(t)$, defined as time series;
- `theta`: A list of model parameters $\boldsymbol{\theta}$ to be estimated.

The output must be a dictionary in which each key corresponds to a model response, and each value is a list representing the variable trajectory over the specified time vector.

This modular separation of simulation as an internal kernel enables *MIDDoE* to function as an extensible, modular framework. The same simulation engine can be called uniformly across sampling, parameter estimation, and input optimisation tasks, facilitating flexible experimentation across a wide range of dynamic systems.

3.5. Executive modules

The executive modules of the *MIDDoE* logic layer (Fig. 3) are further explained here to include implementation details of the aforementioned methods (see Section 2).

Module (a): `sc_sensa` – *GSA*

This module implements Sobol's based-GSA (see Section 2.2) with support for parameter or control input sampling. It allows analysis over either the full feasible space or a damped region around nominal values. Parallel sampling and evaluation are supported to distribute samples across assigned logical CPUs, thereby reducing runtimes. Outputs include structured dictionaries of $\mathbf{S}_1$ and $\mathbf{S}_T$, indexed by model, time, and response.

Key features, and their corresponding configuration keywords are summarised in Table S2 together with additional implementation and execution details that are provided in the Supplementary Material S5.1.

Modules (b): `iden_parmest` – *parameter estimation*

Parameter estimation is defined as an optimisation problem employing various cost functions, including LS, WLS, MLE, and CS (see Section 2.3). Model responses are normalised by their maximum observed values across batches, and sample sizes are scaled by response type to promote consistency across outputs with differing dynamic ranges. The parameter space is normalised to $[0,1]^{N_\theta}$ to improve numerical stability and mitigate bias caused by differences in parameter magnitudes.

A variety of local optimisers, including Sequential Least Squares Programming (SLSQP), Nelder-Mead simplex (NMS), Broyden–Fletcher–Goldfarb–Shanno (BFGS), Limited-memory BFGS (LMBFGS), and trust-region constrained (TC) methods as well as Differential Evolution (DE) for global search, are available solvers (Jorge Nocedal, 2006; Virtanen et al., 2020). To overcome the restricted exploratory of local optimisers, they can be executed in multi-start routines and parallel computing can be employed to prevent excessive runtimes. Parallelisation assigns each optimisation task a randomly selected initial guess on a specific logical CPU, with the final solution chosen as the best among all converged tasks. These solvers operate within user-defined parameter bounds and can be initialised either randomly or from a nominal parameter vector, acting on the specified cost function.

Modules (c): `iden_uncert` – *uncertainty analysis*

This module constructs the sensitivity matrix with all terms expressed in the scaled parameter space to ensure numerical consistency. To balance non-linear errors from large perturbation steps with truncation errors from excessively small steps (especially in the presence of weakly varying responses), *MIDDoE* implements an adaptive step-size strategy. This strategy performs a mesh-independency test on the perturbation magnitude and selects a value near the plateau region, where the eigenvalues of the variance–covariance matrix $\mathbf{V}_\theta$ remain approximately constant.

To improve derivative accuracy, both forward (first-order) and central (second-order) schemes of Finite Difference Method (FDM) are available. Additionally, when the chosen optimiser supplies the Hessian (i.e., second derivatives of the objective), an asymptotic method leverages this information to estimate uncertainty directly. For enhanced robustness, a frequentist bootstrap approach is also implemented (see Section 2.4) and complemented with moment estimates based on truncated normal fits to the bootstrap parameter distributions (Casella and Berger, 2024).

An overview of estimation and uncertainty options, including solver types, objective functions, and output controls, is summarised in Table S3 together with executive details in the Supplementary Material S5.2.

Module (d): `iden_expera` – *In silico Experimentation*

This module generates synthetic experimental data for benchmarking numerical strategies. It can be used by supplying a settings dictionary that defines the true model, true parameters, type of error (absolute or relative), and batch setup controls. These values are treated as ground truth for the selected model. Simulations are conducted accordingly, after which synthetic noise is independently added to each measured response. The noise is sampled from a normal distribution with the absolute or signal relative standard deviations.

The generated data is automatically saved in `.xlsx` format for downstream analysis.

Module (e): `sc_estima` – *EA*

This module supports EA with orthogonalisation method (detailed in Section 2.5) to increase the effective degrees of freedom in mode identification by identifying and fixing practically insignificant parameters. The module outputs the parameter ranking (relative to their original positions), the optimal subset size, the complete r_{cc} profile (see the Supplementary Material S3), and the associated WLS cost function output values for diagnostic comparison.

Modules (f, g): `des_md`, *and*`des_pp` – *MBDoE-MD and MBDoE-PP*

Physically adaptive MBDoE modules plays central role in *MIDDoE* by providing a systematic framework to design informative experiments for model discrimination (MBDoE-MD) and parameter precision (MBDoE-PP) (see Section 2.6). These modules act as high-level interfaces to a uniform architecture that separates the workflow into variable indexing and scaling, objective evaluation, optimisation kernel, and result reporting. This modularity ensures numerical consistency and computational efficiency.

The MBDoE optimisation kernel of *MIDDoE* currently supports DE as a particle-based global optimiser, a Direct Search method (DS) (Hooke and Jeeves, 1961) as a gradient-free local optimiser, and a joint global-local strategy (DEPS) that refines global solutions through local search. These methods are embedded with hard constraints directly enforced within the optimisation, to address the strong non-convexities inherent in dynamic models and constraint formulations in sequential MBDoE problems. Computational efficiency and robustness against local minima are enhanced via multi-start strategies distributed across logical CPU cores with parallelisation. These approaches help by assigning each optimisation task a randomly selected initial guess to a specific logical CPU, and by selecting the best solution among all converged tasks.

These modules are designed to operate in iterative workflows, where model predictions, parameter estimates, and associated uncertainties are updated round by round.

MIDDoE accepts the specification of the decision vector $\boldsymbol{\varphi}$ through either scalar controls or CVPs, supporting piecewise-constant (CPF) and piecewise-linear (LPF) profiles, each subject to design space bounds and perturbation limits. The framework also supports physical and experimental constraints, such as the number and spacing of samples, inclusion and exclusion zones, forced sampling times, synchronisation of sampling across outputs, and switching constraints in control profiles.

An overview of configuration options for MBDoE-MD and MBDoE-PP and their executive details are summarised in Table S4 of the Supplementary Material S5.3.

Module (h): `iden_valida` – *Cross-Validation*

This module performs the LOOCV method (detailed in Section 2.7) treating each batch as a fold.

Module (i): `iden_utils` – *Post-Analysis and Reporting*

Designed to support result interpretation, this module generates visual and tabular outputs, including design decision plots, estimability profiles, model fits, sensitivity indices, validation metrics, inter-parametric confidence regions, and the propagation of *t*-values and P-values after each experimental round for all the candidate models. All outputs are stored in `.xlsx` format for external analysis and in `.jac` format for maximal information retention and portability.

3.6. Package dependencies

The *MIDDoE* library depends on several external packages beyond the standard Python library to support its core functionalities. These dependencies provide essential statistical, mathematical, and computational tools necessary for *MIDDoE* operation. All required packages are installed automatically with *MIDDoE*, ensuring a smooth and consistent setup process for users. Table 2 summarises these dependencies, including their specific versions and the roles they fulfil within the library.

3.7. Package limitations

MIDDoE has been developed with a focus on model identification and system analysis, which means that its simulation capabilities are intrinsically limited by those of SciPy when compared to mature modelling tools such as *PYOMO* and *gPROMS*. However, this limitation can be partially addressed by integrating *MIDDoE* with external simulators, even if at the cost of increased computational burden. Seamless integration with more advanced solvers as an internal model-solving capability could improve this computational bottleneck in future versions. Such improvements would also enable the solution of PDAE models. Furthermore, *MIDDoE* currently lacks techniques to identify the feasible design space (Geremia et al., 2023), which would enable the detection of meaningful subsets of operating conditions – an important direction for future developments.

4. Using *MIDDoE*: Case studies

MIDDoE has been successfully employed in previous experimental studies investigating the kinetics of mineral carbonation (Tabrizi et al., 2025). The following examples, executed with *MIDDoE* version 1.0.0, will be considered to provide more insight into the package applications:

- *Discrimination among candidate models:* when only a limited number of experiments can be conducted, and the selection of the most representative model is hindered by minor divergence of models in predicting the phenomena, MBDoE-MD offers a powerful strategy.

Table 2
List of *MIDDoE* package dependencies including version numbers and functionality descriptions.

Package	Version	Functionality	Reference
Numpy	2.0.2	Numerical computations, efficient array handling, and vectorised mathematical operations.	(Harris et al., 2020)
Scipy	1.13.1	ODE solving, integration, interpolation, statistics, and basic constrained optimisation features.	(Virtanen et al., 2020)
Pandas	2.2.3	Managing, manipulating, and analysing tabular data, CSV and Excel file I/O.	(McKinney, 2010)
Matplotlib	3.9.4	Visualisation of plots, charts, and graphs.	(Hunter, 2007)
Pymoo	0.6.1.3	Solving complex constrained optimisation problems.	(Blank and Deb, 2020)

By optimising the informativeness of each experiment, this technique enhances the ability to distinguish between candidate models. Once a model is selected, parameters are estimated and their precision is evaluated. The identified model is subsequently validated to confirm its predictive ability. The first case study addresses a challenging model discrimination scenario, in which four competing models are expected to explain the observed phenomena.

- *Parameter estimation: experimental data are available and no additional data can be obtained:* in cases where the obtained data are insufficient to precisely calibrate the model parameters and no further experiments can be conducted, EA is employed to fix the unidentifiable parameters and minimise uncertainty, while maintaining acceptable predictive accuracy based on the available data. The second case study initially discusses this challenge, i.e. the identification of a model once the experimental campaign has concluded.
- *Parameter estimation: designing new experiments under minimal experimental budget:* when no prior information is available and the goal is to minimise the size of the experimental campaign, local identifiability issues may arise due to an ill-conditioned **I**. In such cases, EA is first employed to stabilise the MBDoE-PP process against numerical instabilities. Subsequently, MBDoE-PP is used to sequentially improve the quality of the experiment design, maximising the number of precisely estimable parameters until the experimental or computational budget is exhausted, or no improvement is observed. The second case study tackles also this second challenge, i.e. designing experiments under a limited budget for a model that suffers from estimability issues.

4.1. Case study 1: Discrimination among candidate models

This case study focuses on model discrimination within a fed-batch fermentation process for intracellular enzyme production by baker's yeast. The system – described using an unstructured and unsegregated modelling framework – operates isothermally with a single growth-limiting substrate. The feed stream contains no product. The individual steps of this workflow are described in detail below. The *MIDDoE* modules applied to construct a discriminative model identification workflow are shown in Fig. 4 and detailed as follows.

4.1.1. Step 1: Definition of the system and candidate models

Dynamic behaviour is characterised by biomass concentration ($\mathbf{y}_1$, g·L^{-1}) and substrate concentration ($\mathbf{y}_2$, g·L^{-1}), manipulated through the dilution rate ($\mathbf{u}_1$, h^{-1}), feed substrate concentration ($\mathbf{u}_2$, g·L^{-1}), and the initial biomass concentration ($w_1 = y_1(0)$, g·L^{-1}). Model M_I (Eq. 18.a), a Monod-type formulation with maintenance, is assumed to represent the true system with true parameters as $\boldsymbol{\theta}_\mathbf{I} = [0.25, 0.25, 0.88, 0.09]$, and a feasible parameter space of ±50% for model calibration.

Three alternative kinetic models – M_{II} (Eq. 18b, Canoid kinetics), M_{III} (Eq. 18c, linear growth), and M_{IV} (Eq. 18d, Monod without maintenance) – are considered as candidate structures for discrimination. The goal is to evaluate whether experiment designs could effectively distinguish these candidates from the assumed true model, and if the *MIDDoE* is capable to perform the MBDoE-MD methods with following the physically constrained and bounded design space, represented in Table 3. The model formulations and parameter sets are adapted from (Chen and Asprey, 2003).

$$M_I(\boldsymbol{\theta},\mathbf{U},\mathbf{Y}): \begin{cases} \dfrac{dy_1(t)}{dt} = (r - u_1(t) - \theta_4)y_1(t) \\ y_1(0) = w_1 \\ \dfrac{dy_2(t)}{dt} = -\dfrac{ry_1(t)}{\theta_3} + u_1(t)(u_2(t) - y_2(t)) \\ y_2(0) = 0.01 \\ r = \dfrac{\theta_1 y_2(t)}{\theta_2 + y_2(t)} \end{cases} \tag{18a}$$

$$M_{II}(\boldsymbol{\theta},\mathbf{U},\mathbf{Y}): \begin{cases} \dfrac{dy_1(t)}{dt} = (r - u_1(t) - \theta_4)y_1(t) \\ y_1(0) = w_1 \\ \dfrac{dy_2(t)}{dt} = -\dfrac{ry_1}{\theta_3} + u_1(t)(u_2(t) - y_2(t)) \\ y_2(0) = 0.01 \\ r = \dfrac{\theta_1 y_2(t)}{\theta_2 y_1(t) + y_2(t)} \end{cases} \tag{18b}$$

$$M_{III}(\boldsymbol{\theta},\mathbf{U},\mathbf{Y}): \begin{cases} \dfrac{dy_1(t)}{dt} = (r - u_1(t) - \theta_3)y_1(t) \\ y_1(0) = w_1 \\ \dfrac{dy_2(t)}{dt} = -\dfrac{ry_1}{\theta_2} + u_1(t)(u_2(t) - y_2(t)) \\ y_2(0) = 0.01 \\ r = \theta_1 y_2(t) \end{cases} \tag{18c}$$

$$M_{IV}(\boldsymbol{\theta},\mathbf{U},\mathbf{Y}): \begin{cases} \dfrac{dy_1(t)}{dt} = (r - u_1(t))y_1(t) \\ y_1(0) = w_1 \\ \dfrac{dy_2(t)}{dt} = -\dfrac{ry_1}{\theta_3} + u_1(t)(u_2(t) - y_2(t)) \\ y_2(0) = 0.01 \\ r = \dfrac{\theta_1 y_2(t)}{\theta_2 + y_2(t)} \end{cases} \tag{18d}$$

To generate the in silico data, the assumed true model M_I is simulated using known true parameter values by `krnl_expera`. Two batch experiments are designed and simulated over a 40-hour process time under the following operational conditions, aligned with the defined design space:

- **Experiment 1:** $u_1(t) = 0.05$ h^{-1}, $u_2(t) = 30.00$ g·L^{-1}, $w_1 = 1.00$ g·L^{-1}
- **Experiment 2:** $u_1(t) = 0.10$ h^{-1}, $u_2(t) = 30.00$ g·L^{-1}, $w_1 = 1.00$ g·L^{-1}

For each output trajectory, 10 sampling points are uniformly distributed over the process duration. At each sampling time, measurement noise is introduced by adding normally distributed absolute

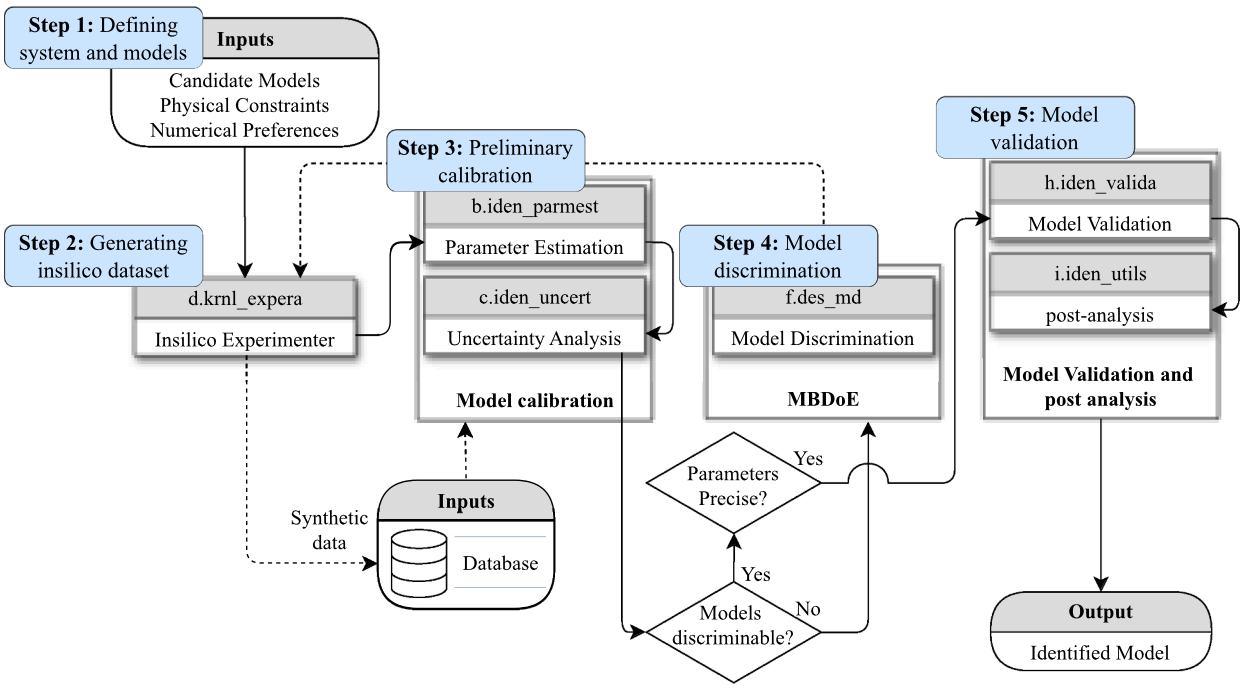

Fig. 4. Model identification workflow implemented by *MIDDoE* for model discrimination in Case Study 1. The workflow includes the following steps: (1) definition of the system and candidate models followed by generation of in silico data; (2) calibration of candidate models; and (3) design of experiments using MBDoE-MD. Generation of data and steps (2)-(3) are repeated until the models are successfully discriminated and calibrated.

Table 3
Design space and constraints.

Controls	$\mathbf{u}_1$	$\mathbf{u}_2$	w_1
Min	0.05	5	1
Max	0.2	35	5.5
CVP	CPF	LPF	-
Steps	4	4	-
Relative constraint	relaxed	relaxed	-
Minimum perturbed signal	0.01	1	-
Minimum switching time interval	0.5	0.5	
Measurements	$\mathbf{y}_1$		$\mathbf{y}_2$
Number of Sampling points	10		10
Minimum sampling time interval	0.5		0.5
Sampling point strategy	Synchronised		

errors, with standard deviations of 15% for $\mathbf{y}_1$ and 10% for $\mathbf{y}_2$. This simulated noise reflects the experimental variability typically encountered due to the different characterisation techniques employed.

4.1.2. Step 2: Calibration of candidate models

A WLS cost function and DE global optimisation algorithm are applied using `iden_parmest` to perform preliminary parameter estimation. DE is selected to reduce the likelihood of being trapped in local optima during parameter estimation and to avoid biases introduced by varying initial guesses.

Following calibration, parameter precision (*t*-value), and the probability of each model being the true one (P-value) are evaluated using `iden_uncert`. These values are reported in Table 4, which highlights the relative likelihood of model M_I being the correct representative, based on the highest P-value. Nonetheless, model M_{II} also exhibits considerable predictive capability, indicating that additional data are needed to effectively discriminate between the top two candidates. All remaining models are excluded from the model discrimination campaign due to negligible P-values.

The performance of the two remaining candidate models is illustrated against the observation points from the preliminary experiments in Fig. 5.

4.1.3. Step 3: Design of experiments using MBDoE-MD, followed by recalibration and model discrimination

To further discriminate between M_I, and M_{II}, a new experiment is designed using `des_md`, employing both the HR and BFF design criteria through joint optimisation strategy that combines DE and Direct Search for local refinement. This approach reduces the likelihood of the optimiser getting trapped in suboptimal local minima while refining the solution locally. The MBDoE-MD optimisation incorporates the design

Table 4
Preliminary calibration results.

Param-eter	$M_I(\boldsymbol{\theta},\mathbf{U},\mathbf{Y})$		$M_{II}(\boldsymbol{\theta},\mathbf{U},\mathbf{Y})$		$M_{III}(\boldsymbol{\theta},\mathbf{U},\mathbf{Y})$		$M_{IV}(\boldsymbol{\theta},\mathbf{U},\mathbf{Y})$	
	Estimate	*t*-value	Estimate	*t*-value	Estimate	*t*-value	Estimate	*t*-value
θ_1	0.25	14.91	0.26	7.49	0.06	2.81	0.15	41.03
θ_2	0.26	10.10	0.02	5.11	0.58	2.95	0.37	6.91
θ_3	0.88	11.31	0.94	5.92	0.02	0.69	0.45	25.67
θ_4	0.09	5.90	0.10	3.25	-	-	-	-
$t^{ref,95\%}$	2.02		2.02		2.02		2.02	
P-value	90.47%		9.53%		0.00%		0.02%	

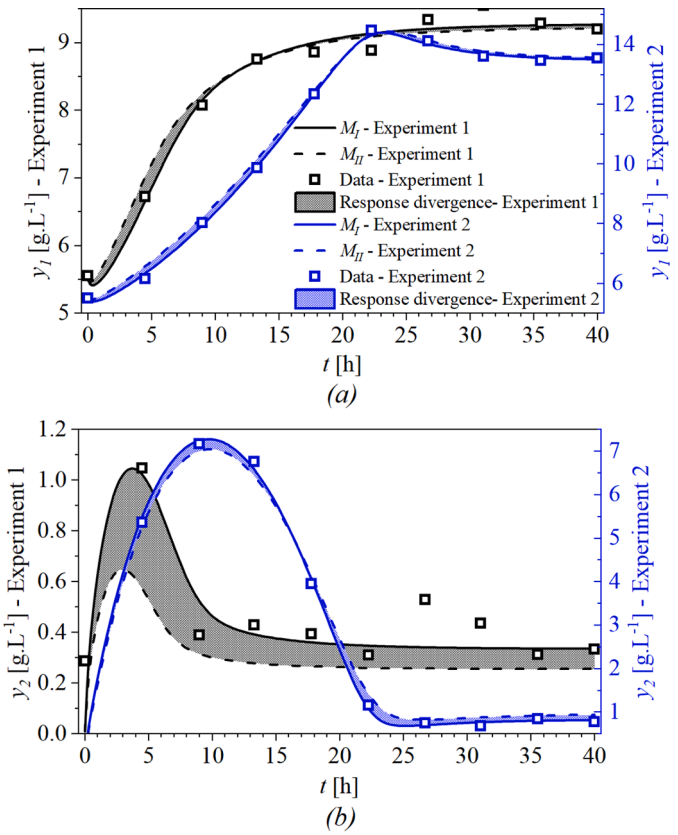

Fig. 5. Overlay plot of data and predictions of candidate models (M_I, M_{II}) after preliminary calibration for measurements of (a) $\mathbf{y_1}$, (b) $\mathbf{y_2}$ in experiments 1 and 2.

space and experimental constraints outlined in Table 3, ensuring feasibility and addressing practical limitations through the optimisation kernel of *MIDDoE*. The resulting experiment designs, along with the expected response divergences based on manipulated CVPs, are illustrated in Figs. 6 and 7 for the HR and BFF criteria, respectively.

All HR designs conform to the predefined design space and constraints; no optimisation constraints are violated. In contrast, some of the CVP switching times in the BFF designs violate the Minimum CVP switching signal constraint (see the Supplementary Materials S6). This violation occurs when the same computational budget is defined for the optimiser of both designs. However, it could have been avoided by allocating a higher number of maximum iterations to the optimisation of the BFF design.

Building upon the experiment designs outlined above, these approaches are compared in terms of their impact on decision-making and ultimate discriminative capability in systems characterised by absolute errors. Under such conditions, the BFF design approach can be misleading, as it tends to focus on regions with lower signal levels where noise dominates the measurements, thereby reducing their effectiveness on calibration of the resulting experimental data (further detailed in the Supplementary Material S6). The final discrimination decision is made following simulated experimental data generation and model recalibration, using the same procedures described in Sections 4.1.1 and 4.1.2. Specifically, new synthetic data are generated based on the optimised experiment design, appended to the original dataset, and used to recalibrate all models. Initial parameter values are updated using the preliminary estimations from Section 4.1.2. The final divergences between models are illustrated in Figs. 6 and 7, and the resulting P-values for HR and BFF designs are 99.9% and 99.2%, respectively.

In both scenarios, M_I achieves a high P-value of more than 99%, clearly confirming its status as the most representative model of the underlying process. This round also yields updated parameter estimates, 95% CIs, and corresponding *t*-values, summarised in Table 5. This table also reveals notable shifts in parameter estimates for model M_{II}. All parameter estimates of model M_I exhibit *t*-values greater than the reference threshold of 2.00, confirming their precision. Furthermore, the estimated values and their 95% CIs include the true parameters used for simulation, indicating high accuracy in both model calibration and discrimination.

4.2. Case study 2: Model calibration

This case study applies the *MIDDoE* framework to calibrate a mechanistic model describing a batch pharmaceutical synthesis process (Moshiritabrizi et al., 2024). The model captures the dynamic behaviour of reactions involved in the production of a quaternary chloride salt intermediate (QS1Cl), relevant to HIV drug manufacturing. The synthesis involves a reversible main reaction between the starting material (SM) and trimethylamine (TMA), and a parallel side reaction that irreversibly converts QS1Cl into an impurity (ClDMI) and a volatile by-product (MeCl):

$$SM + TMA \underset{k_r}{\overset{k_f}{\rightleftarrows}} QS1Cl, \quad QS1Cl \xrightarrow{k_{fs}} CIDMI + MeCl \tag{19}$$

In scenario 1, a concluded experimental campaign is assumed, where EA assists selection of estimable parameters, and thereby achieving an identified form of model with maximum possible precision of parameters and preserved predictive capability.

In scenario 2, a limited experimental budget is assumed without providing prior information, where GSA is used to tailor the model for first MBDoE run, by detecting and fixing unimportant parameters without prior information. After obtaining a first set of in silico data, EA is used to reassess the state of parameters for their capability of being identified, and fixing non-estimable parameters. At the end of each loop, parameters are estimated, and uncertainty evaluated. These workflows are illustrated in Fig. 8 and explained stepwise.

4.2.1. Step 1: Definition of the system and candidate model

This dynamic behaviour is characterised by the concentrations of the starting material ($C_{SM}(t)$ as $\mathbf{y}_1$), the main product ($C_{QS1Cl}(t)$ as $\mathbf{y}_2$), and the impurity ($C_{CIDMI}(t)$ as $\mathbf{y}_3$), all in mol·L^{-1}. These outputs are influenced by the initial concentrations of SM ($C_{SM}(0)$ as w_1), and TMA ($C_{TMA}(0)$ as w_2), as time-invariant controls, and a time-varying temperature profile ($T(t)$ as $\mathbf{u}_1$), serving as the manipulated variables in the model identification task. Model *M* (Eq. 20) represents the mass and energy balance of the QS1Cl production process with a true parameter vector $\boldsymbol{\theta}$ = [50000, 75000, 0.4116, 111900, 9905, 30000]. Fixed constants in the system are R=8.314 J·mol^{-1}·K^{-1} as universal gas constant, and T_{ref}=296.15 K as reference temperature. The reactions are studied for 16 h of process time. The design space and its associated constraints are summarised in Table 6, while the feasible parameter space and the initial guess are presented in Table 7.

$$M(\boldsymbol{\theta}, \mathbf{U}, \mathbf{Y}): \begin{cases} \frac{dC_{SM}(t)}{dt} = -k_f \cdot C_{SM}(t) \cdot C_{TMA}(t) + k_r \cdot C_{QS1Cl}(t), \quad C_{SM}(0) = w_1 \\ \frac{dC_{TMA}(t)}{dt} = -k_f \cdot C_{SM}(t) \cdot C_{TMA}(t) + k_r \cdot C_{QS1Cl}(t), \quad C_{TMA}(0) = w_2 \\ \frac{dC_{QS1Cl}(t)}{dt} = k_f \cdot C_{SM}(t) \cdot C_{TMA}(t) - k_r \cdot C_{QS1Cl}(t) - k_{fs} \cdot C_{QS1Cl}(t), \quad C_{TMA}(0) = 0 \\ \frac{dC_{ClDMI}(t)}{dt} = k_{fs} \cdot C_{QS1Cl}(t), \quad C_{ClDMI}(0) = 0 \\ \frac{dC_{MeCl}(t)}{dt} = k_{fs} \cdot C_{QS1Cl}(t), \quad C_{MeCl}(0) = 0 \\ k_f = \theta_1 \cdot \exp\left(\frac{-\theta_2}{R}\left(\frac{1}{T(t)} - \frac{1}{T_{ref}}\right)\right) \\ k_{fs} = \theta_3 \cdot \exp\left(\frac{-\theta_4}{R}\left(\frac{1}{T(t)} - \frac{1}{T_{ref}}\right)\right) \\ k_r = \frac{k_f}{k} \\ k = \theta \cdot \exp\left(\frac{\theta_6}{R}\left(\frac{1}{T(t)} - \frac{1}{T_{ref}}\right)\right) \end{cases} \tag{20}$$

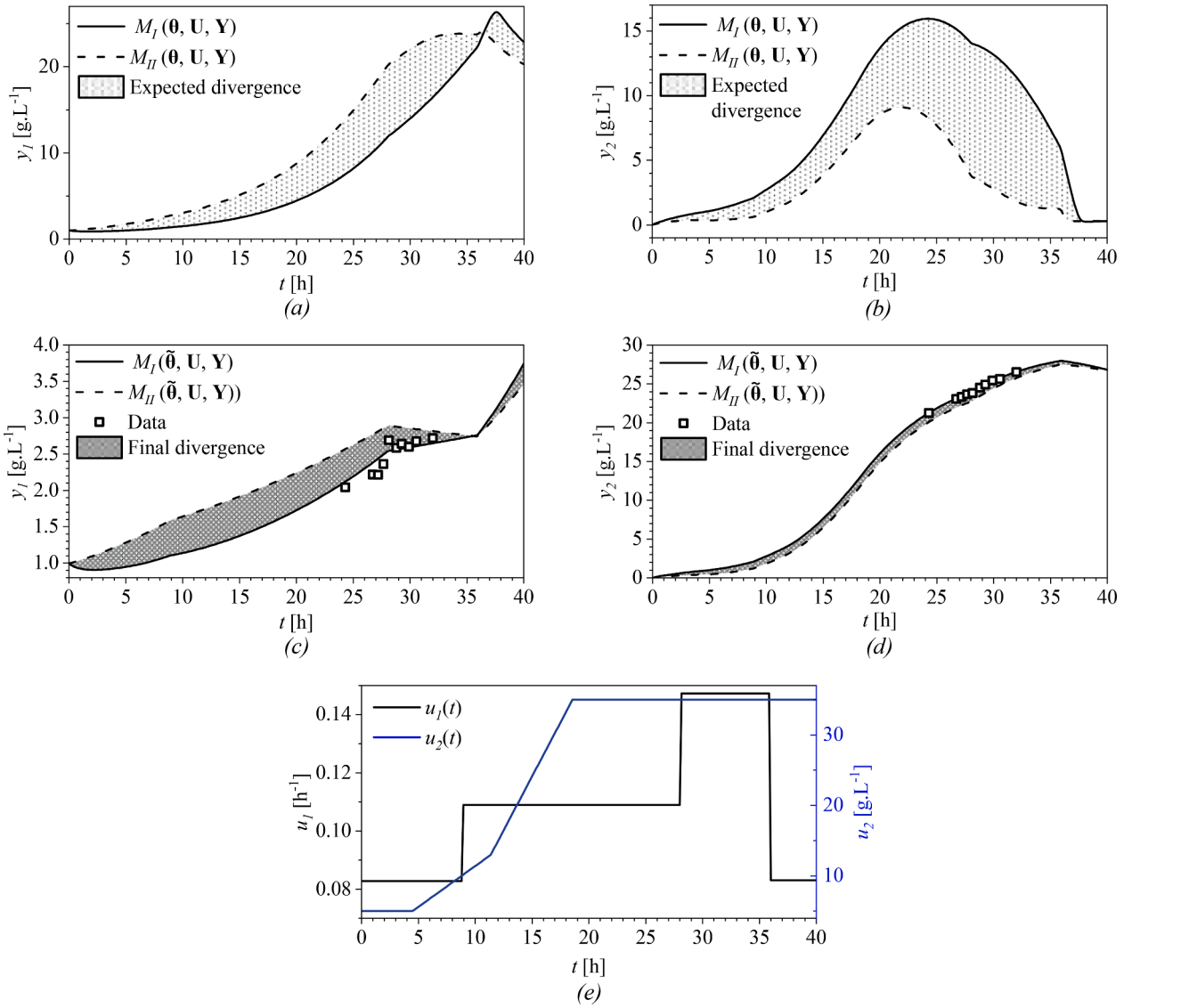

Fig. 6. Designed experiments for MBDoE-MD using the HR criterion, showing expected divergence for (a) y_1, (b) y_2, and divergence after recalibration for (c) $\mathbf{y_1}$, (d) $\mathbf{y_2}$, along with (e) the designed CVPs.

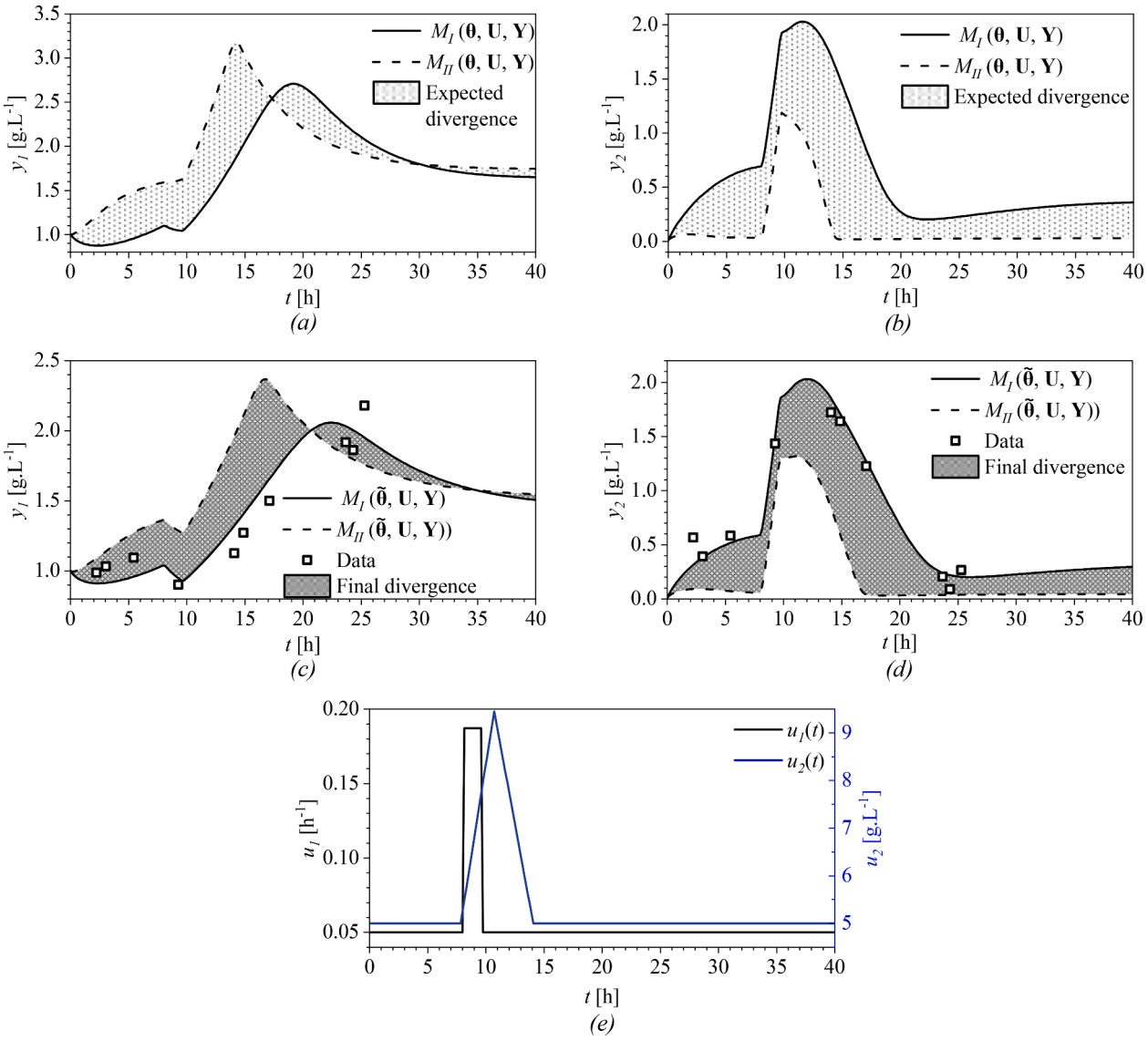

Fig. 7. Designed experiments for MBDoE-MD using the BFF criterion, showing expected divergence for (a) y_1, (b) y_2, and divergence after recalibration for (c) y_1, (d) y_2, along with (e) the designed CVPs.

Table 5
Parameters and their precision after recalibration.

Parameter	HR design						BFF design					
	$M_I(\boldsymbol{\theta}, \mathbf{U}, \mathbf{Y})$			$M_{II}(\boldsymbol{\theta}, \mathbf{U}, \mathbf{Y})$			$M_I(\boldsymbol{\theta}, \mathbf{U}, \mathbf{Y})$			$M_{II}(\boldsymbol{\theta}, \mathbf{U}, \mathbf{Y})$		
	Estimate	CI 95%	t-value	Estimate	CI 95%	t-value	Estimate	CI 95%	t-value	Estimate	CI 95%	t-value
θ_1	0.25	±0.01	23.08	0.22	±0.03	7.86	0.25	±0.02	15.82	0.26	±0.07	14.92
θ_2	0.25	±0.02	13.14	0.01	±0.01	1.72	0.25	±0.03	10.46	0.02	±0.02	3.14
θ_3	0.88	±0.06	17.88	0.78	±0.01	6.50	0.88	±0.10	12.04	0.95	±0.33	3.86
θ_4	0.09	±0.01	8.56	0.098	±0.03	2.80	0.09	±0.02	6.22	0.10	±0.06	2.16

4.2.2. Scenario 1: Available experimental data and no possibility for further experiments

Four batches of experimental data, each containing 17 uniformly distributed sampling points, are generated in silico using the `krnl_expera` module with 0.5% absolute normal noise (as described in Section 4.1.1). The experiments are configured as follows:

- **Experiment 1:** $u_1(t) =$ 296.15 K, $w_1 =$ 0.366 mol·L^{-1}, $w_2 =$ 0.19 mol·L^{-1}
- **Experiment 2:** $u_1(t) =$ 306.15 K, $w_1 =$ 0.366 mol·L^{-1}, $w_2 =$ 0.19 mol·L^{-1}
- **Experiment 3:** $u_1(t) =$ 296.15 K, $w_1 =$ 0.65 mol·L^{-1}, $w_2 =$ 0.595 mol·L^{-1}

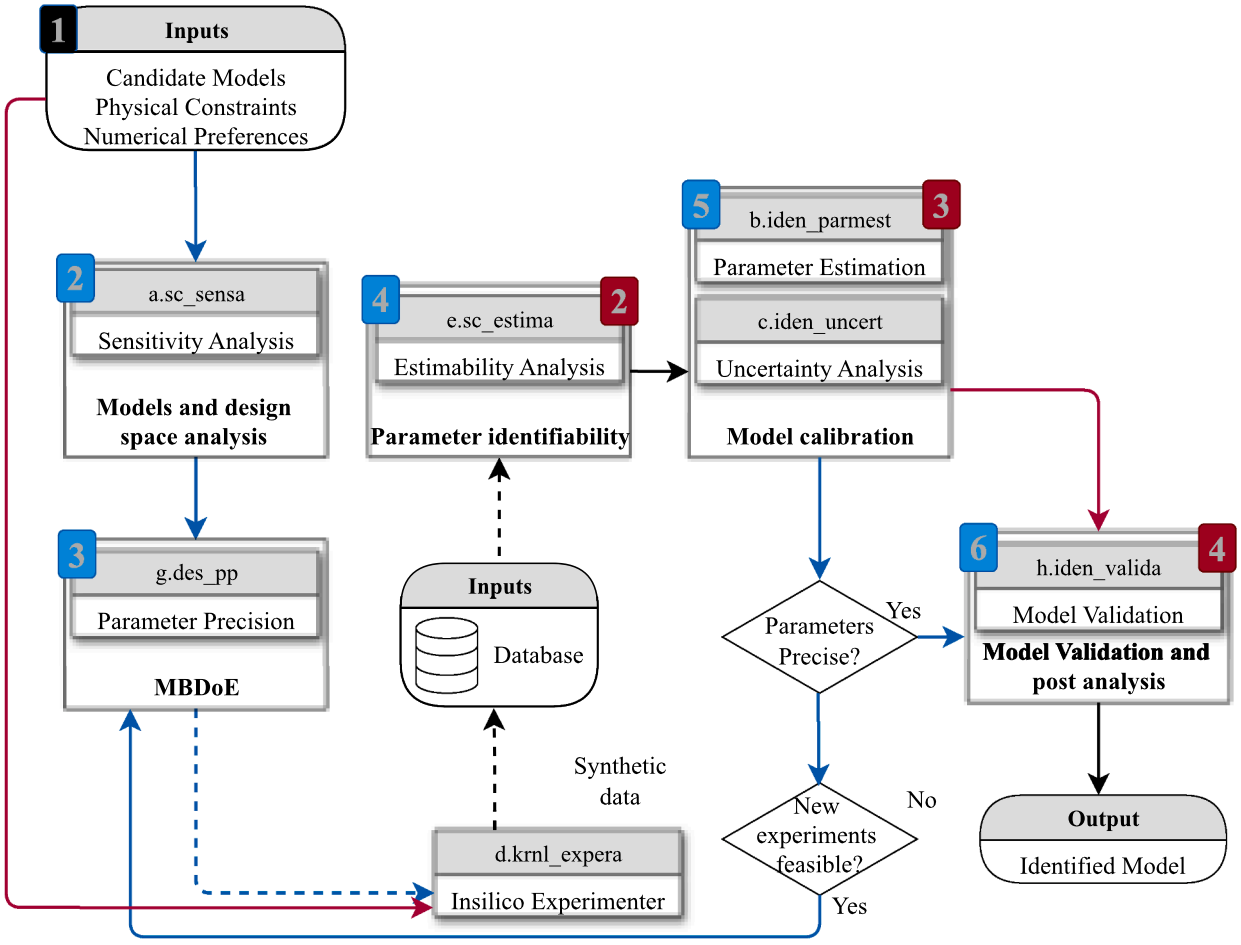

Fig. 8. Model identification workflow implemented by *MIDDoE* for Case Study 2. Arrows and step numbers in red indicate Scenario 1, blue indicate Scenario 2, and black represent shared steps. Following step (1), which defines the system and candidate models, in Scenario 1, the workflow generates in silico data and proceeds with (2) estimable parameter subset selection, and (3) model calibration. Generation of data and Steps (2)–(3) are repeated until the full observation matrix is evaluated, followed by model validation in step (4). In Scenario 2, the process begins with (2) GSA for preliminary estimable parameter selection, followed by (3) MBDoE-PP design and in silico data generation, (4) subset selection, and (5) model calibration. Steps (3)–(5) are repeated until the experimental budget ends, concluding with model validation in step (6).

Table 6
Design space and constraints.

Controls	$\mathbf{u}_1$	w_1	w_2
Min	296.15	0.05	0.1
Max	306.15	1.0	1.0
CVP	LPF	-	-
Steps	6	-	-
Relative constraint	increasing	-	-
Minimum perturbed signal	0.01	-	-
Minimum switching time interval	0.3	-	
Measurements	$\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3$		
Number of Sampling points	17		
Minimum sampling time interval	0.3		
Fixed sampling points	0, 16		
Sampling point strategy	synchronised		

Table 7
Feasible parameter space.

Parameter	θ^0	Lower bound	Upper bound
θ_1	100000	10000	1000000
θ_2	100000	0	200000
θ_3	1	0.1	10
θ_4	100000	50000	200000
θ_5	100	10	10000
θ_6	10000	10000	200000

- **Experiment 4:** $u_1(t) = 306.15$ K, $w_1 = 0.65$ mol·L^{-1}, $w_2 = 0.595$ mol·L^{-1}

These configurations constitute the full list of input settings used for the four experiments.

4.2.2.1. Step 2: Estimable parameter subset selection. Batch data are sequentially generated and appended to the observation matrix. At each round, EA is performed, followed by parameter subset selection and model calibration using the selected estimable parameters. The EA procedure is called by the `sc_estima` module and employs the LMBFGS optimiser with a multi-start strategy. The sensitivity matrix is constructed using the central finite difference method.

Throughout the four EA calls, parameter θ_3 consistently ranks as the most estimable, while the rankings of the remaining parameters varies across batches. When all experimental data are combined, the final ranking of parameters in descending order of estimability is: θ_3, θ_4, θ_5, θ_6, θ_1, θ_2. The parameter subset selection of EA indicates that only the first 3 parameters in the list are estimable.

Table 8
Model calibration results for Scenario 1.

	Round 1		Round 2		Round 3		Round 4	
Parameter	Estimate	*t*-value	Estimate	*t*-value	Estimate	*t*-value	Estimate	*t*-value
θ_1	–	–	–	–	–	–	–	–
θ_2	–	–	–	–	–	–	–	–
θ_3	0.4115	52.34	0.4120	55.97	0.4141	174.70	0.4155	205.25
θ_4	–	–	116038	14.69	116670	10.76	112017	66.30
θ_5	4210	1.73	3853	1.82	6141	2.99	7051	4.08
θ_6	–	–	–	–	–	–	–	–
$t^{ref,95\%}$		2.01		1.98		1.97		1.97

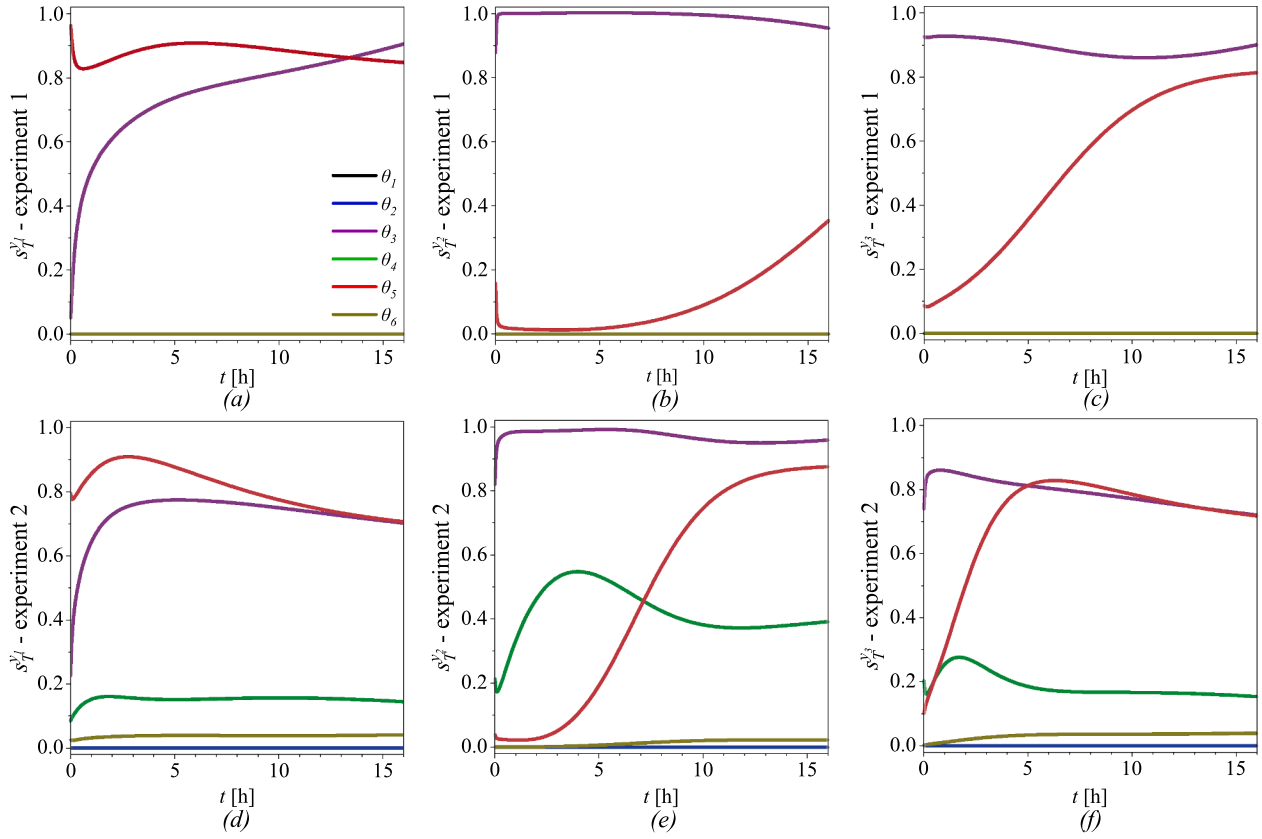

Fig. 9. Variation of the total Sobol index $S_{Ti}^{y_j}$ of parameters θ_i over process time for the measured responses: (a) $\mathbf{y}_1$, (b) $\mathbf{y}_2$, and (c) $\mathbf{y}_3$ in experiment 1, and (d) $\mathbf{y}_1$, (e) $\mathbf{y}_2$, and (f) $\mathbf{y}_3$ in experiment 2.

4.2.2.2. Step 3: Model calibration. After each EA run, selected parameters are estimated using the `iden_parmest` module with identical optimiser settings. The uncertainty of these estimations is assessed via the `iden_uncert` module, employing a bootstrap method[1]. Table 8 lists the estimated parameters alongside their *t*-values. Based on the comparison with the reference *t*-value at 95% confidence level, all the EA suggested parameters are statistically assessed to be precise after being estimated with the full dataset. Estimated values show progressive precision and accuracy with the sequential addition of information to the workflow. The final model is identified with a high predictive capability (R^2 of 0.99).

4.2.2.3. Step 4: Model validation. Cross-validation of the final state of identified model is performed using the `iden_valida` module. It yields a high validation R^2 of 0.9985 ± 0.0015, which together with the high precision of parameter estimates and strong predictive performance, confirms the model predictive ability.

4.2.3. Scenario 2: Sequential design of new experiments

4.2.3.1. Step 2: Preliminary ranking of parameters. A preliminary evaluation of the contribution of each parameter to the measured responses is performed using the `sc_sensa` module. This analysis is performed to assess the influence of parameters before designing any experiments in order to tune the model and to avoid the numerical instabilities of MBDoE. Sampling is conducted across the entire feasible parameter space, while fixing the input profiles to those of experiments 1 and 2 in Scenario 1. Fig. 9 illustrates the variation of the total Sobol index over time for each measured state variable under the respective input conditions.

The results indicate a descending order of average parameter influence, with θ_3, θ_5, θ_4, and θ_6 contributing most significantly. Based on this ranking, θ_1 and θ_2 are considered non-influential under the tested operational conditions and fixed to their initial guess before running the

[1] This method is selected due to the presence of sloppy parameters and ill-conditioning of the Fisher information matrix.

Table 9
Model calibration results for Scenario 2.

	Round 1		Round 2		Round 3		Round 4	
Parameter	Estimate	*t*-value	Estimate	*t*-value	Estimate	*t*-value	Estimate	*t*-value
θ_1	–	–	–	–	–	–	–	–
θ_2	–	–	–	–	–	–	–	–
θ_3	0.4170	125.12	0.4160	34.65	0.4162	53.50	0.4163	52.95
θ_4	–	–	110692	11.80	109686	17.08	109686	14.32
θ_5	4004	2.49	4225	3.19	4337	2.56	4908	3.05
θ_6	–	–	–	–	–	–	107093	2.42
$t^{ref,95\%}$		2.01		1.98		1.97		1.97

first MBDoE-PP.

4.2.3.2. Steps 3 and 4: Design of experiments and in silico data generation. The first model-based experiment focused on parameter precision is designed using the `des_pp` module. D-optimality is employed to maximise the initial information content, aiming to reduce the uncertainty of the four remaining parameters. The global-local joint optimiser (DEPS) in *MIDDoE* is used for this step. The same algorithmic settings are applied in the subsequent rounds, but with an E-optimality criterion to target the identification of the less influential parameters. The optimality criterion of MBDoE-PP is switched from D-optimality to E-optimality to shift the focus from enhancing the precision of the most influential parameter to improving that of the least precisely estimated one. This switch reallocates the optimisation budget toward designing experiments that minimise the largest eigenvalue of the variance–covariance matrix, thereby improving the precision of parameters with lower *t*-values. A similar scenario, where D-optimality is maintained throughout the workflow for comparison purposes, is presented in Supplementary Material S7. All designed experiments compile with the physical constraints of the process. These experiments are conducted entirely in silico, under the same error regime described in Scenario 1 (Section 4.2.2).

4.2.3.3. Step 5: Estimable parameter subset selection. This step is carried out immediately after each round of experiment design using the same specifications as in Scenario 1 (Section 4.2.2.2). Consistently, θ_3 is identified as the most estimable parameter throughout the sequence. However, by the end of the campaign, four parameters are suggested as estimable by EA. The final ranking of parameters is $\theta_3,\ \theta_5,\ \theta_4,\ \theta_6,\ \theta_1, \theta_2$ and the richer information acquired through MBDoE-PP has enabled the detection of θ_6 as an estimable parameter in this scenario.

The estimable subset of parameters is assessed using the same settings described in Scenario 1 (Section 4.2.2.2). Table 9 presents the sequential parameter calibration following each newly designed experiment, and the selection of estimable parameters. Comparing the results of this scenario with those obtained from non-MBDoE-designed experiments highlights a more substantial improvement in precision after the first design, attributable to D-optimality. Additionally, the number of statistically significant parameters increases from three to four. Switching to E-optimality in the second round improves the precision of the least estimable parameters, while slightly reducing it for the most precise ones. The final model exhibits a high predictive capability, with a R^2 of 0.99.

4.2.3.4. Step 6: Model validation. Leveraging the same approach as in Scenario 1 (Section 4.2.2.3), the model is validated and yields a high validation R^2 of 0.9996, confirming the identified model predictive ability.

5. Conclusions

This work presented *MIDDoE*, a modular and end-to-end *Python* package for comprehensive employment of MBDoE within model identification workflows. It is based on three conceptual layers – client, logic, and kernel – letting it operate independently of specific model structure syntaxes. Its architecture treats the model and simulator as part of a core kernel, with standardised data interfaces to ensure seamless integration. This design supports both built-in and externally defined simulation routines, offering flexibility and interoperability across different modelling environments.

As a simulation-driven framework for systems governed by ODEs, *MIDDoE*'s logic layer implements techniques for designing experiments and conducting (1) model discrimination and (2) model calibration. These techniques are further supported by GSA and EA to assess the parameter space to fix practically insignificant parameters. This improves the robustness of MBDoE in ill-conditioned systems, and guides selection of estimable parameters. Besides, it facilitates practical applications by supporting the enforcement of physical constraints and user-defined optimisation configurations. The MBDoE engine is built to handle non-convex design spaces and practical experimental limitations. It is further supported by integrated numerical techniques for input space exploration, parameter estimable subset selection and estimation, uncertainty quantification, and model validation.

Finally, *MIDDoE* is built with usability as a core principle. For experimentalists and users without programming experience, the client layer provides a clear and intuitive interface. Meanwhile, advanced users can customise workflows using modular tools within the logic layer. This dual approach, combined with the use of simple Python/Numpy arrays, ensures that *MIDDoE* can serve users seeking a plug-and-play tool without requiring familiarisation with a specific software ecosystem.

Although *MIDDoE* currently supports the most common sequential MBDoE techniques, several challenges remain that warrant further investigations both in terms of experimental strategies and model compatibility. These include detecting feasible design spaces and constraining MBDoE within them, enabling online or inline redesigning of experiments, and extending support to more complex PDAE structures.

List of acronyms and symbols (all acronyms and symbols refer to general sections only and not to the case studies)

Acronyms

BFF	Buzzi-Ferraris and Forzatti (a MBDoE-MD method)
BFGS	Broyden–Fletcher–Goldfarb–Shanno (an optimisation method)
CI	Confidence Interval
CPF	Constant Piecewise Profile (a CVP method)
CS	Chi-square (a fitting cost function)
CVP	Control Vector Parameterisation
DAE	Differential Algebraic Equation
DE	Differential Evolution (an optimisation method)
DEPS	Differential Evolution and Pattern Search (an optimisation method)
DoF	Degree of Freedom
DoE	Design of Experiments
EA	Estimability Analysis
GP	Gaussian Process
GSA	Global Sensitivity Analysis
HR	Hunter and Reiner (a MBDoE-MD method)
LMBFGS	Limited-memory BFGS (an optimisation method)

(*continued on next page*)

(*continued*)

LMS	Left Matrix Scrambling
LOOCV	leave-one-out cross-validation
LPF	Linear Piecewise Profile (a CVP method)
LS	Least Squares (a fitting cost function)
MBDoE	Model-Based Design of Experiments
MBDoE-MD	Model-Based Design of Experiments for Model Discrimination
MBDoE-PP	Model-Based Design of Experiments for Parameter Precision
MIMO	multiple-input, multiple-output (a model structure)
MLE	Maximum Likelihood Estimation (a fitting cost function)
NMS	Nelder-Mead simplex (an optimisation method)
PDAE	Partial Differential Algebraic Equation
PS	Pattern Search (an optimisation method)
SINDy	Sparse Identification of Nonlinear Dynamics
SLSQP	Sequential Least Squares Programming (an optimisation method)
TC	Trust-region constrained (an optimisation method)
WLS	Weighted Least Squares (a fitting cost function)

Latin symbols

CI_i	Confidence Interval of estimated parameter $\widehat{\theta}_i$
$\mathbf{F}_{l,l',t}$	Uncertainty weight factor of T-optimal design criteria at each sampling time t for models l, and l'
f	Differential equation in the MIMO system
g	Algebraic equation in the MIMO system
$\mathbf{I}$	Fisher information matrix $[N_\theta \times N_\theta]$
M	Model as a MIMO system
P_i	Probability of the i-th model being the best among N_m candidates
$\mathbf{Q}_r$	Local sensitivity matrix for each measured response r at all sampling times $t_{N_{tp}}$ $[N_{t_{N_{tp}}} \times N_\theta]$
$\mathbf{Q}_{l,t}$	Local sensitivity matrix of model l at each sampling time t for measured responses $[N_r \times N_\theta]$
R^2	Coefficient of determination
$\mathbf{S}_1$	Matrix of Sobol' first-order for model state variables $[N_r \times N_t]$
$\mathbf{S}_\mathbf{T}$	Matrix of Sobol' total-order for model state variables $[N_r \times N_t]$
t	Time vector encompassing all control and measurement time points $[N_t \times 1]$
t_i	t-value of estimated parameter $\widehat{\theta}_i$
t^{ref}	Reference threshold, corresponding to a Student t-value
$T_{l,l'}$	T-optimal design criteria for MBDoE-MD between models l and l'
$\mathbf{U}$	Matrix of time-dependent control inputs $[N_u \times N_t]$
$\mathbf{u}$	Vector of time-dependent control inputs at each time step $[N_u \times 1]$
$\mathbf{V}_\theta$	Variance-covariance matrix of model parameters $[N_\theta \times N_\theta]$
$v_{\theta,ii}$	The i-th diagonal element of $\mathbf{V}_\theta$
$\mathbf{W}_{l,t}$	Modeling errors contribution in MBDoE-MD of model l at each sampling time t
$\mathbf{w}$	Time-invariant control inputs $[N_w]$
$\mathbf{X}$	Matrix of time-variant state variables, governed by differential equation $[N_x \times N_t]$
$\mathbf{x}$	Vector of time-variant state variables at each time step, governed by differential equation $[N_x \times 1]$
$\dot{\mathbf{x}}$	Vector of time-variant derivatives of state variables at each time step, governed by differential equation $[N_x \times 1]$
$\mathbf{Y}$	Matrix of measured state variables $[N_r \times N_t]$
$\mathbf{y}$	Vector of measured state variables at each time step $[N_r \times 1]$
$\widehat{\mathbf{y}}$	Vector of model predictions at each time step $[N_r \times 1]$
$\widehat{y}_{(r,t)}$	Model prediction of response r at time step t
$\mathbf{Z}$	Matrix of time-variant state variables, governed by algebraic equation $[N_z \times N_t]$
$\mathbf{z}$	Vector of time-variant state variables at each time step, governed by algebraic equation $[N_z \times 1]$

Greek symbols

α	Significance level
χ^2	Chi-Square value
$\boldsymbol{\epsilon}$	Vector of errors $[N_r \times 1]$
ψ	Scalar metrics of MBDoE-PP problems (so called alphabetical optimal criterial)
$\widetilde{\sigma}_{(r,r')}$	Elements from the inverse of the measurement errors variance-covariance matrix
$\boldsymbol{\theta}$	Vector of true values of model parameters $[N_\theta]$
$\widehat{\boldsymbol{\theta}}$	Vector of estimated values of model parameters $[N_\theta]$
$\widehat{\theta}_i$	The i-th estimated parameter
$\boldsymbol{\Sigma}_\theta$	Prior variance-covariance matrix of model parameters $[N_\theta \times N_\theta]$
$\boldsymbol{\Sigma}_y$	Measurement errors variance-covariance matrix $[N_r \times N_r]$
$\boldsymbol{\varphi}$	Design vector of MBDoE $[N_\varphi]$
$\boldsymbol{\varphi}_{\text{OPT}}$	Optimal design vector $[N_\varphi]$
$\boldsymbol{\Phi}$	The feasible design space

(*continued on next column*)

Funding

This work is part of the CO2Valorize project that has received funding from the European Union's Horizon Europe research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 101073547.

Data availability statement

MIDDoE is available as a pip-installable package via PyPI at https://pypi.org/project/middoe, with the user and API documentation accessible through the project's documentation site at https://middoe.readthedocs.io.Additionally, case studies and source code can be accessed through the project's GitHub repository at https://github.com/zuhairblr/middoe.

CRediT authorship contribution statement

Z. Tabrizi: Conceptualization, Investigation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. **E. Barbera:** Supervision, Writing – review & editing. **W.R. Leal da Silva:** Supervision, Writing – review & editing. **F. Bezzo:** Conceptualization, Methodology, Supervision, Writing – review & editing.

Declaration of competing interests

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: W. R. Leal da Silva is an employee of FLSmidth Cement A/S.

Acknowledgements

The authors would like to thank the following people at Siemens Industry Software Limited for their support in coupling *MIDDoE* with *gPROMS*: Prof. Costas Pantelides for his supervision and theoretical guidance on the coupling approach; Mr. Bart de Groot for his supervision of the procedure and technical support; Dr. Xenia Kleniati and Mr. Nathan Samra for their assistance with the *gPAS* structure and *pygpas* syntax.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.dche.2025.100276.

References

Blank, J., Deb, K., 2020. Pymoo: multi-objective optimization in Python. IEEe Access. 8, 89497–89509. https://doi.org/10.1109/ACCESS.2020.2990567.

Brunton, S.L., Proctor, J.L., Kutz, J.N., 2016. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Natl. Acad. Sci. U S A 113, 3932–3937. https://doi.org/10.1073/pnas.1517384113.

Calder, B., Grunwald, D., Zorn, B., 1994. Quantifying behavioral differences between C and C++ programs. J. Progr. Lang. 2, 313–351.

Casella, G., Berger, R., 2024. Statistical Inference, 2nd ed. Chapman and Hall/CRC, Boca Raton. https://doi.org/10.1201/9781003456285.

Chen, B.H., Asprey, S.P., 2003. On the design of optimally informative dynamic experiments for model discrimination in multiresponse nonlinear situations. Ind. Eng. Chem. Res. 42, 1379–1390. https://doi.org/10.1021/ie0203025.

Cobelli, C., DiStefano, J.J., 1980. Parameter and structural identifiability concepts and ambiguities: a critical review and analysis. Am. J. Physiol.-Regul., Integr. Compar. Physiol. 239, R7–R24. https://doi.org/10.1152/ajpregu.1980.239.1.R7.

Buede, D.M., 2024. The engineering design of systems: models and methods, 4th ed. John Wiley and Sons, New York.

Efron, B., 1979. Bootstrap methods: another look at the jackknife. Ann. Stat. 7, 1–26. https://doi.org/10.1214/aos/1176344552.
Espie, D., Macchietto, S., 1989. The optimal design of dynamic experiments. AIChE J. 35, 223–229. https://doi.org/10.1002/aic.690350206.
Franceschini, G., Macchietto, S., 2008. Model-based design of experiments for parameter precision: state of the art. Chem. Eng. Sci. 63, 4846–4872. https://doi.org/10.1016/j.ces.2007.11.034.
Galvanin, F., 2010. Optimal model-based design of experiments in dynamic systems: novel techniques and unconventional applications.
Galvanin, F., Cao, E., Al-Rifai, N., Gavriilidis, A., Dua, V., 2016. A joint model-based experimental design approach for the identification of kinetic models in continuous flow laboratory reactors. Comput. Chem. Eng. 95, 202–215. https://doi.org/10.1016/j.compchemeng.2016.05.009.
Geremia, M., Bezzo, F., Ierapetritou, M.G., 2023. A novel framework for the identification of complex feasible space. Comput. Chem. Eng. 179, 108427. https://doi.org/10.1016/j.compchemeng.2023.108427.
Geremia, M., Macchietto, S., Bezzo, F., 2026. A review on model-based design of experiments for parameter precision – open challenges, trends and future perspectives. Chem. Eng. Sci. 319, 122347. https://doi.org/10.1016/j.ces.2025.122347.
Gutenkunst, R.N., Casey, F.P., Waterfall, J.J., Myers, C.R., Sethna, J.P., 2007. Extracting falsifiable predictions from sloppy models. Ann. N. Y. Acad. Sci. 203–211. https://doi.org/10.1196/annals.1407.003.
Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E., 2020. Array programming with NumPy. Nature 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2.
Hooke, R., Jeeves, T.A., 1961. Direct search" solution of numerical and statistical problems*. J. ACM (JACM) 8, 212–229. https://doi.org/10.1145/321062.3210.
Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. Comput. Sci. Eng. 9, 90–95. https://doi.org/10.1109/MCSE.2007.55.
Hunter, W.G., Reiner, A.M., 1965. Designs for discriminating between two rival models. Technometrics 7, 307–323. https://doi.org/10.1080/00401706.1965.10490265.
Iwanaga, T., Usher, W., Herman, J., 2022. Toward SALib 2.0: advancing the accessibility and interpretability of global sensitivity analyses. Socio-Environ. Syst. Model. 4, 18155. https://doi.org/10.18174/sesmo.18155.
Jacquez, J.A., Greif, P., 1985. Numerical parameter identifiability and estimability: integrating identifiability, estimability, and optimal sampling design. Math. Biosci. 77, 201–227. https://doi.org/10.1016/0025-5564(85)90098-7.
Jorge Nocedal, S.J.W., 2006. Numerical Optimization, 2nd ed. Springer New York. https://doi.org/10.1007/978-0-387-40065-5.
Klise, K.A., Nicholson, B.L., Staid, A., Woodruff, D.L., 2019. Parmest: parameter estimation via Pyomo. In: Muñoz, S.G., Laird, C.D., Realff, M.J. (Eds.), Proceedings of the 9th International Conference on Foundations of Computer-Aided Process Design, Computer Aided Chemical Engineering. Elsevier, pp. 41–46. https://doi.org/10.1016/B978-0-12-818597-1.50007-2.
Kusumo, K.P., Kuriyan, K., Vaidyaraman, S., García-Muñoz, S., Shah, N., Chachuat, B., 2022. Risk mitigation in model-based experiment design: A continuous-effort approach to optimal campaigns. Comput. Chem. Eng. 159. https://doi.org/10.1016/j.compchemeng.2022.107680.
Laky, D.J., Casas-Orozco, D., Laird, C.D., Reklaitis, G.V., Nagy, Z.K., 2022. Simulation–Optimization framework for the digital design of pharmaceutical processes using Pyomo and PharmaPy. Ind. Eng. Chem. Res. 61, 16128–16140. https://doi.org/10.1021/acs.iecr.2c01636.
Lyu, W., Galvanin, F., 2025. DoE-SINDy: an automated framework for model generation and selection in kinetic studies. Comput. Chem. Eng. 202. https://doi.org/10.1016/j.compchemeng.2025.109265.
McKinney, W., 2010. Data structures for statistical computing in Python, pp. 56–61. https://doi.org/10.25080/Majora-92bf1922-00a.
Miao, H., Xia, X., Perelson, A.S., Wu, H., 2011. On identifiability of nonlinear ODE models and applications in viral dynamics. SIAM Rev. 53, 3–39. https://doi.org/10.1137/090757009.
Moshiritabrizi, I., Abdi, K., McMullen, J.P., Wyvratt, B.M., McAuley, K.B., 2024. Parameter estimation and estimability analysis in pharmaceutical models with uncertain inputs. AIChE J. 70, e18168. https://doi.org/10.1002/aic.18168.
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, 187–223. https://doi.org/10.1007/s12532-017-0127-0.
Olofsson, S., Hebing, L., Niedenführ, S., Deisenroth, M.P., Misener, R., 2019. GPdoemd: A Python package for design of experiments for model discrimination. Comput. Chem. Eng. 125, 54–70. https://doi.org/10.1016/j.compchemeng.2019.03.010.
Owen, A.B., 1998. Scrambling Sobol' and NiederreiterrXing points. J. Complex.
Pistikopoulos, E.N., Barbosa-Povoa, A., Lee, J.H., Misener, R., Mitsos, A., Reklaitis, G.V., Venkatasubramanian, V., You, F., Gani, R., 2021. Process systems engineering – the generation next? Comput. Chem. Eng. https://doi.org/10.1016/j.compchemeng.2021.107252.
Rasch, A., Bücker, H.M., 2010. EFCOSS: an interactive environment facilitating optimal experimental design. ACM Trans. Math. Softw. 37. https://doi.org/10.1145/1731022.1731023.
Raue, A., Kreutz, C., Maiwald, T., Bachmann, J., Schilling, M., Klingmüller, U., Timmer, J., 2009. Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. Bioinformatics 25, 1923–1929. https://doi.org/10.1093/bioinformatics/btp358.
Rényi, A., Renyi, A., 1965. On the foundations of information theory. Revue de l'Institut International de Statistique /Rev. Int. Stat. Inst. 33, 1. https://doi.org/10.2307/1401301.
Rodriguez, J.S., Laird, C.D., Zavala, V.M., 2020. Scalable preconditioning of block-structured linear algebra systems using ADMM. Comput. Chem. Eng. 133, 106478. https://doi.org/10.1016/j.compchemeng.2019.06.003.
Saltelli, A., Annoni, P., Azzini, I., Campolongo, F., Ratto, M., Tarantola, S., 2010. Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index. Comput. Phys. Commun. 181, 259–270. https://doi.org/10.1016/j.cpc.2009.09.018.
Saltelli, A., Ratto, M., Tarantola, S., Campolongo, F., 2006. Sensitivity analysis practices: strategies for model-based inference. Reliab. Eng. Syst. Saf. https://doi.org/10.1016/j.ress.2005.11.014.
Sobol', I.M., 1990. Sensitivity estimates for nonlinear mathematical models. Mat. Model. 2, 112–118.
Stone, M., 1974. Cross-validatory choice and assessment of statistical predictions. Source: J. R. Stat. Soc. Ser. B (Methodol.). https://doi.org/10.1111/j.2517-6161.1974.tb00994.x.
Tabrizi, S.Z.B., Barbera, E., Silva, W.R.L.da, Bezzo, F., 2025. A Python/Numpy-based package to support model discrimination and identification. In: Van Impe, J.F.M., Léonard, G., Bhonsale, S.S., Polańska, M.E., Logist, F. (Eds.), Systems and Control Transactions, Proc. of the 35th European Symposium on Computer Aided Process Engineering, pp. 1282–1287. https://doi.org/10.69997/sct.192104.
Tabrizi, Z., Rodriquez, C., Barbera, E., Leal da Silva, W.R., Bezzo, F., 2025. Wet carbonation of industrial recycled concrete fines: experimental study and reaction kinetic modeling. Ind. Eng. Chem. Res. https://doi.org/10.1021/acs.iecr.5c02835.
Vanlier, J., Tiemann, C.A., Hilbers, P.A., van Riel, N.A., 2014. Optimal experiment design for model selection in biochemical networks. BMC. Syst. Biol. 8, 20. https://doi.org/10.1186/1752-0509-8-20.
Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., Vijaykumar, A., Pietro, Bardelli, A., Rothberg, A., Hilboll, A., Kloeckner, A., Scopatz, A., Lee, A., Rokem, A., Woods, C.N., Fulton, C., Masson, C., Häggström, C., Fitzgerald, C., Nicholson, D.A., Hagen, D.R., Pasechnik, D.V., Olivetti, E., Martin, E., Wieser, E., Silva, F., Lenders, F., Wilhelm, F., Young, G., Price, G.A., Ingold, G.L., Allen, G.E., Lee, G.R., Audren, H., Probst, I., Dietrich, J.P., Silterra, J., Webber, J.T., Slavič, J., Nothman, J., Buchner, J., 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods 17, 261–272. https://doi.org/10.1038/s41592-019-0686-2.
Wang, J., Dowling, A.W., 2022. Pyomo.DOE: an open-source package for model-based design of experiments in Python. AIChE J. 68. https://doi.org/10.1002/aic.17813.
Wu, S., McLean, K.A.P., Harris, T.J., McAuley, K.B., 2011. Selection of optimal parameter set using estimability analysis and MSE-based model-selection criterion. Int. J. Adv. Mechatron. Syst. 3, 188. https://doi.org/10.1504/IJAMECHS.2011.042615.
Yao, K.Z., Shaw, B.M., Kou, B., McAuley, K.B., Bacon, D.W., 2003. Modeling ethylene/butene copolymerization with multi-site catalysts: parameter estimability and experimental design. Polym. React. Eng. 11, 563–588. https://doi.org/10.1081/PRE-120024426.