Systems & Control Transactions

Research Article - Peer Reviewed Conference Proceeding
ESCAPE 35 - European Symposium on Computer Aided Process Engineering
Ghent, Belgium. 6-9 July 2025
Jan F.M. Van Impe, Grégoire Léonard, Satyajeet S. Bhonsale,
Monika E. Polańska, Filip Logist (Eds.)

PSE PRESS

A Python/Numpy-based package to support model discrimination and identification

Seyed Zuhair Bolourchian Tabrizi[a,b], Elena Barbera[a], Wilson Ricardo Leal da Silva[b], and Fabrizio Bezzo[a]*

[a] Department of Industrial Engineering, University of Padova, via Marzolo 9, 35131 Padova PD, Italy
[b] FLSmidth Cement, Green Innovation, Denmark
* Corresponding Author: fabrizio.bezzo@unipd.it.

ABSTRACT

Addressing challenges in process design and optimisation, especially with complex models and data uncertainties, requires effective tools for model development, selection, and identification. Techniques such as Model-based Design of Experiments (MBDoE) help support this task by screening and discriminating between models and, eventually, calibrating them. Open-source and user-friendly Python packages have implemented some model identification techniques. However, the need for a tool that can couple with various model simulators and account for the steps of model identification as well as physical constraints of systems in design of experiments remains unmet. In that light, we present the python package MIDDOE (Model-(based) Identification, Discrimination, and Design of Experiments) to address this gap. It integrates rival models screening, parameter estimation, uncertainty analysis, and MBDoE techniques, while adapting to various process constraints. These functionalities are demonstrated via an in-silico study for a semi-batch fermentation reactor model identification.

Keywords: model identification, model-based design of experiments, model discrimination, model calibration, open-source software

INTRODUCTION

Process design and optimisation often requires the precise determination of underlying phenomena and identification of accurate models to describe them. This process becomes complex when multiple rival models and high uncertainty exist, and is further complicated by the costly data required for model discrimination and calibration. To address these challenges, numerical techniques have been introduced to streamline the pre-discrimination stage by screening models and narrowing the pool of candidates without additional experimental effort. Building on these techniques, MBDoE methods have been developed to design new experiments that maximize information, enabling easier discrimination between rival models (MBDoE-MD) [1]. Additionally, they reduce the confidence ellipsoid volume of estimated parameters by enriching the information matrix (MBDoE-PP) via optimal experiment design [1].

Open-source Python packages, most notably PYOMO [2], are essential tools for digital modeling frameworks. PYOMO facilitates parameter estimation using Parmest [3] and supports MBDoE-PP through DOE [4]. Despite many advantages in integrating with the PYOMO ecosystem, these tools have some limitations. They are effective when screening, reparametrizing, and discriminating between models are not required. Additionally, they rely on PYOMO's array and model structure, which makes them incompatible with other types of simulators. This dependency limits their flexibility in handling diverse model structures, solvers, and constraints related to design decisions. Another significant limitation is the lack of coherence in problem definitions across the entire workflow. This arises from the need to use different packages for tasks such as screening, identifying, designing experiments, validating, and illustrating results. Additionally, some of these tools are not readily available, further complicating the workflow.

These gaps are here addressed by proposing the Python package MIDDOE (https://pypi.org/project/middoe), which wraps dynamic lumped models using standard Python/NumPy arrays and integrates seamlessly with simulators. MIDDOE provides tools not only for calibrating models, but also for screening, discriminating, and validating models. It provides flexibility to perform these tasks using local methods (single start or multi-core, multi-start) as well as global or joint methods (multi-core) ensuring high computational efficiency.

The package offers MBDoE techniques tailored for both discrimination and calibration, specifically adapted to the physical constraints of experimental campaigns and apparatus. Key features include options for enforcing constraints on design decisions and cost functions, with the flexibility to structure MBDoE optimization problems to accommodate these limitations. This functionality significantly aids experimenters in designing feasible and optimal experiments.

Finally, MIDDOE emphasizes user-friendliness by clearly distinguishing model components and enabling one-time definitions for all techniques, eliminating additional programming. Designed as a numerical wrapper, it can integrate with external simulators or internal Python/NumPy functions, enabling non-expert users to avoid specific programming syntaxes. Data import is simplified by allowing experimental data to be added as Excel files, organized into batches using separate sheets. It supports various data input types and automates the generation and restoration of results, data, and figures. By representing its structure, we demonstrate a part of its novel capabilities through an in-silico experimental campaign for identifying a generic model.

PACKAGE ARCHITECTURE

MIDDOE is a Python library designed to provide a modular structure (see Figure 1) for performing essential model identification steps. These steps include: 1) setting up models and configuring control variables and constraints; 2), 3) screening models with sensitivity and estimability analyses; 4), 5) parameter estimation and uncertainty analysis; 6), 7) designing experiments for model discrimination and parameter precision; and 8) model validation. The package organizes these steps into a user-friendly syntax, beginning with the first step, which involves importing the model(s) for the identification problem and defining structured dictionaries to symbolically decode and classify model components. Models are imported as Differential-Algebraic Equations (DAEs) and solved using a method that efficiently handles both stiff and non-stiff problems simultaneously. Variables are categorized as time-variant or time-invariant inputs and outputs, while parameters are addressed separately. The design space, along with its associated physical constraints, is also defined, followed by specifying the solver properties. Furthermore, each module requires and received its own specific properties to ensure flexibility and adaptability.

For screening methods, the software includes two advanced techniques: Global Sensitivity Analysis (GSA) using the Sobol method [5] and Estimability Analysis (EA) using the orthogonalization method [6] corresponding to steps 2 and 3, respectively. EA, a key feature of the package, helps restructure parametrically complex models by ranking parameters from higher to lower significance. This analysis introduces a methodology for calculating corrected critical ratios and selecting an optimal subset of parameters for estimating, prioritizing those with the lowest corrected critical ratios. This selection ensures that the model structure avoids over-parameterization and biases.

Once the set of rival models is defined, parameter estimation and uncertainty analysis are employed to assess the precision of estimations and the predictability of the identified model, corresponding to steps 4 and 5. These steps are carried out with the flexibility to choose

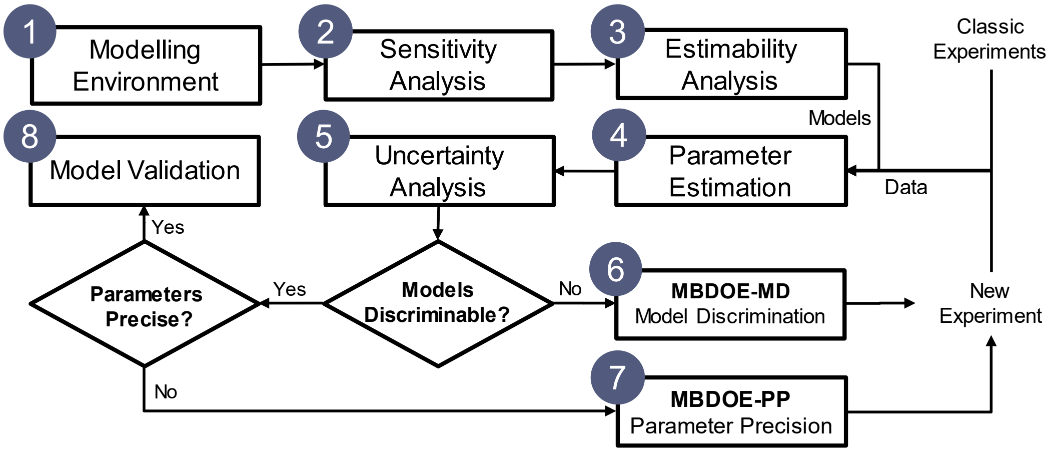

Figure 1: Model identification logic and modules provided in the package.

Table 1: True parametric values for in-silico experiments and their definitions in Monod model.

Parameter	Units	Value	Definition
θ_1	$[\mathrm{h}^{-1}]$	0.31	Maximum specific growth rate
θ_2	$[\mathrm{g.L}^{-1}]$	0.11	Michaelis constant
θ_3	$[-]$	0.65	Yield coefficient
θ_4	$[\mathrm{h}^{-1}]$	0.25	Biomass loss rate due to non-modeled factors
θ_5	$[\mathrm{g.L}^{-1}]$	5.00	Substrate inhibition constant

Table 2: Design space for time-invariant and -variant process controls in Monod model.

Variable	Units	Range	Approximation Profile	Definition
$y_{1,0}$	$[\mathrm{g.L}^{-1}]$	$1-10$	–	Biomass initial concentrations
$y_{2,0}$	$[\mathrm{g}\cdot\mathrm{L}^{-1}]$	$1-10$	–	Substrate initial concentrations
$\mathbf{u_1}$	$[\mathrm{h}^{-1}]$	$0.05-0.2$	Constant Piecewise-relaxed	Time-variant dilution rate
$\mathbf{u_2}$	$[\mathrm{g}\cdot\mathrm{L}^{-1}]$	$5-35$	Constant Piecewise-decreasing	Time-variant feed substrate concentration

between global, gradient-free methods or local, gradient-based methods, supported by an internal Finite Difference Method (FDM) auto-differentiator. A key practical feature of the package addresses scenarios where the experimental campaign fails to effectively discriminate between rival models. In such cases, MBDoE-MD [7] is proposed as step 6. Similarly, MBDoE-PP is utilized in step 7 to design experiments that improve parameter precision in the selected model. Both modules offer flexibility in cost function selection, addressing discrimination and precision goals while incorporating physical constraints into the MBDoE framework.

Finally, an identified model can undergo iterative validation tests against experimental batches in step 8 to assess its robustness. Our proposed package also includes additional visualization tools to support the conclusions of the modeling campaign, such as:

- model fitting with calibration and validation data,
- visualization of inter-parametric confidence ellipsoid areas and hyper-ellipsoid volume shrinkage,
- parameter precision evaluation via t-values and confidence interval changes across batches, and
- p-value comparisons across models.

CASE STUDY AND RESULTS

Model structure and design space

As a matter of example, the software applies estimability analysis (step 3) and MBDoE-PP (step 7). However, executing steps 1, 4, and 5 is necessary to support these tasks. Simplified in silico simulated experiments with a normally distributed noise of 5% on both of the responses are considered using a generic Monod model [8]. This model predicts the concentrations of biomass ($\mathbf{y_1}\,[\mathrm{g.L}^{-1}]$) and substrate ($\mathbf{y_2}\,[\mathrm{g.L}^{-1}]$) in a semi-batch fermentation reactor with continuous feed. Table 1 presents the process model parameters for Eqs. (1) to (3), while Table 2 presents the design space for input variables. The experimental campaign begins with 2 experiments at the center, and maximum of design space boundaries, followed by up to five MBDoE-designed experiments. Each experiment has a budget of 6 sampling points and spans a duration of 10 hours. The use of E-optimality criterion ensures parameter precision exceeds the t-value threshold for all parameter estimations.

$$\frac{\partial y_1(t)}{\partial t} = (r - u_1(t) - \theta_4)\cdot y_1(t) \qquad (1)$$

$$\frac{\partial y_2(t)}{\partial t} = -\frac{r\cdot y_1(t)}{\theta_3}\cdot u_1(t)(u_2 - y_2) \qquad (2)$$

$$r = \left(\frac{\theta_1\cdot x_2}{\theta_2 + x_2}\right)\cdot exp\left(-\frac{x_2}{\theta_5}\right) \qquad (3)$$

Estimability analysis

After providing the necessary information and specifications as domain knowledge (step 1) to the software and simulating the initial experiments, the initial observation matrix is formed. Then, this matrix is used for estimation of parameters (steps 4, and 5), providing enough information to return to Step 3: Estimability Analysis. This step evaluates the significance ranking of parameters and derives the optimal subset for estimation. The ranking of parameters based on their significance, from highest to lowest, is θ_4, θ_3, θ_1, θ_5, and θ_2 for the first experiment observations, and θ_1, θ_4, θ_3, θ_5, and θ_2 for the combined observations from the first and second experiments.

Based on this ranking, the estimability analysis calculates the corrected critical ratios for the estimated subset of parameters as illustrated in Figure 2. With a limited observation matrix, it suggests retaining θ_4 and θ_3 and with an extended one it suggests retaining θ_1, θ_4, and θ_3, as these correspond to the subset with the lowest critical ratio. For both cases, θ_2 and θ_5 are evaluated as insignificant and suggested to be dropped out in the identification procedure by fixing them to their rough estimation. This approach aligns with the t-values (a precision metric based on the 95% confidence intervals of parameters after estimation) obtained at the end of the identification campaign, viz. Figure 3 that indicates greater uncertainty in the estimation of θ_2 with a t-value of 0.006.

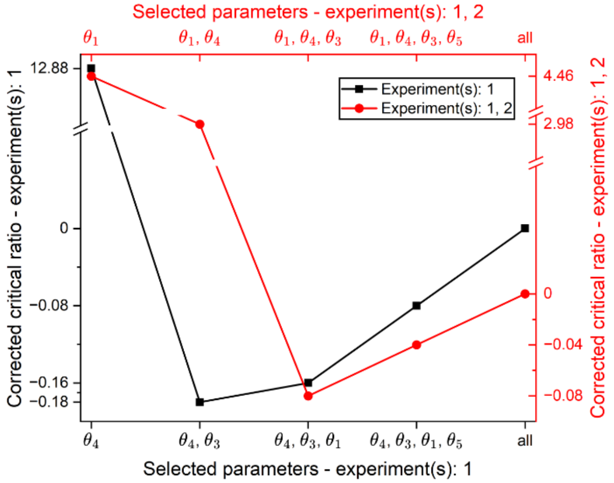

Figure 2: Corrected critical ratios across parameter subsets.

Assuming no additional experimental budget is available and avoiding further experimentation, θ_2 and θ_5 are fixed, resulting in a reduced model with only three parameters to estimate. This reduced model is subsequently employed for parameter estimation (Steps 4 and 5). It achieves a slight improvement in accuracy for all parameters by shifting the Probability Density Function (PDF) of estimates closer to the true values. Additionally, it significantly enhances precision, as indicated by the narrower distribution and more reliable estimates, although the t-values of θ_1, and θ_3 remain below the reference threshold of 2.08.

This module is particularly effective in scenarios with limited experimental budgets, where accurate predictions and precise estimation of key parameters are critical. However, to avoid structural simplifications and achieve higher precision in estimating the original model, we revert to its full form and design new experiments. In Step 7 (Figure 1), MBDoE-PP is employed to design new experiments, enhance the information content of the observation matrix, and estimate all parameters with acceptable precision.

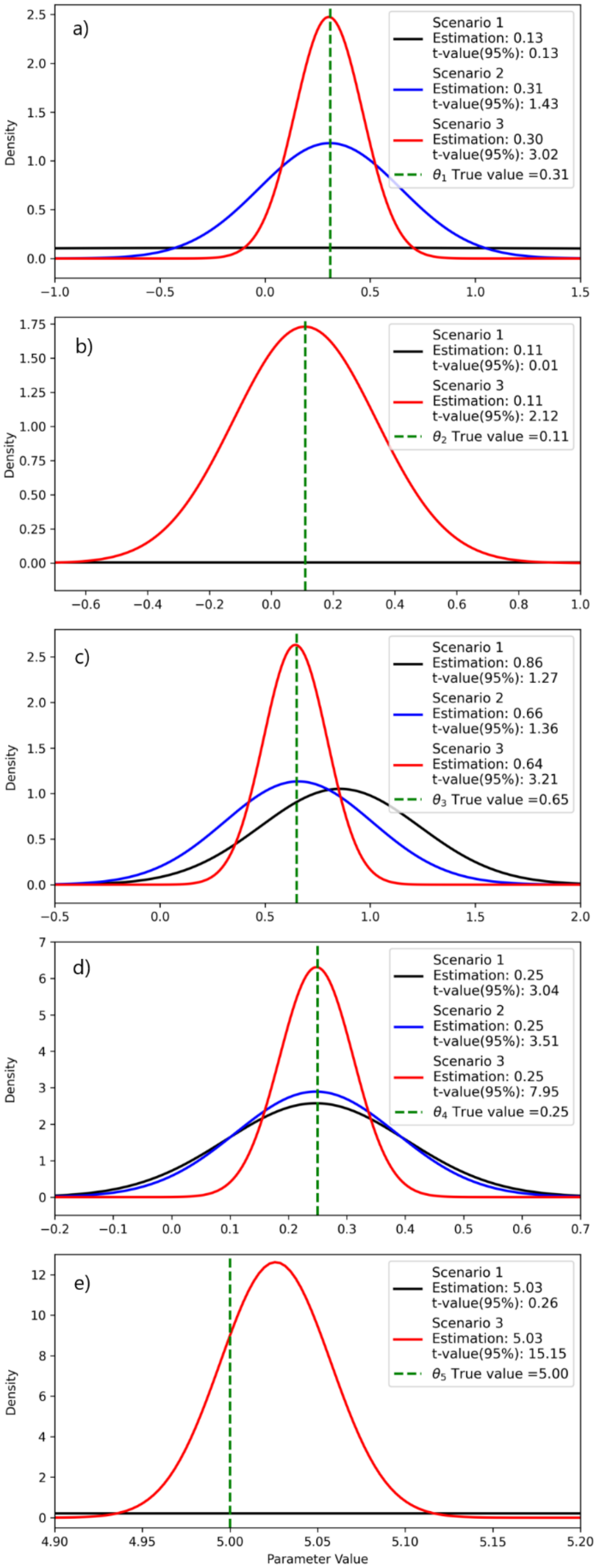

Figure 3: Estimation PDFs, mean and 95% t-values for a) θ_1, b) θ_2, c) θ_3, d) θ_4, and e) θ_5, for the initial experiments (Scenario 1), reduced model (Scenario 2), and MBDoE-assisted campaign (Scenario 3).

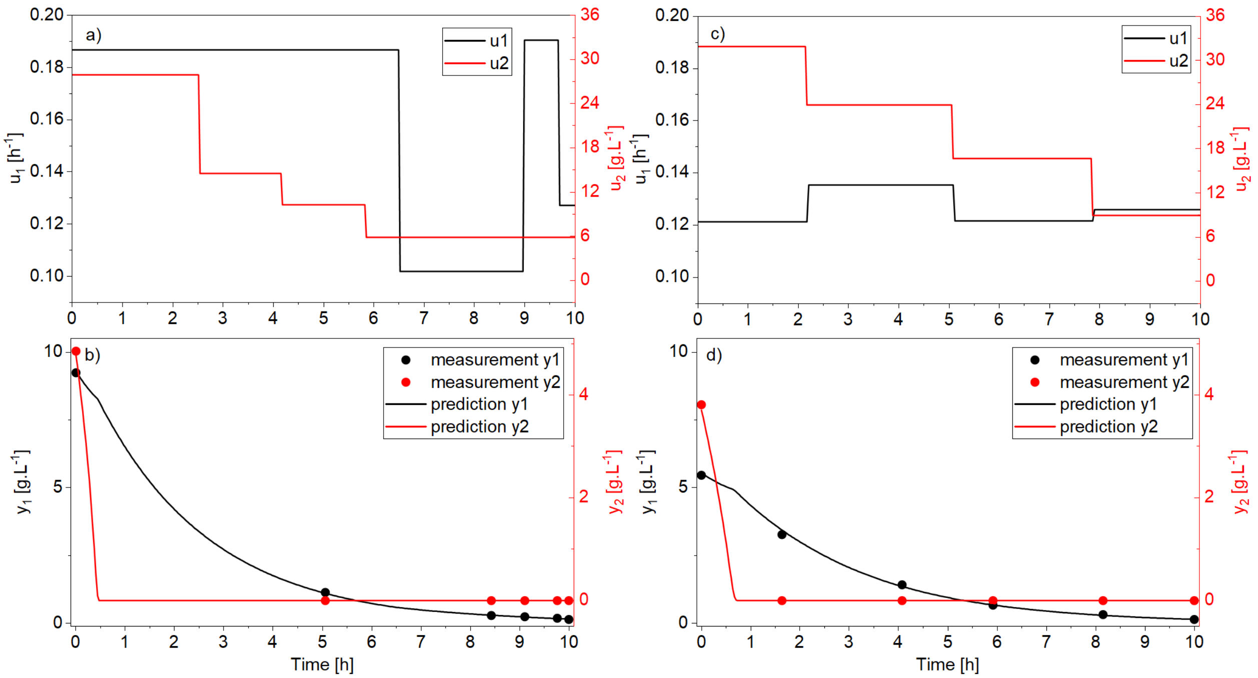

Figure 4: Control vectors $\mathbf{u_1}$ and $\mathbf{u_2}$ for a) 1st, and c) 2nd MBDoE designed experiments and experimental data and corresponding model responses $\mathbf{y_1}$ and $\mathbf{y_2}$ for b) 1st, and d) 2nd MBDoE designed experiments

Model-based design of experiments

The MBDoE-related designs incorporate constraints detailed in Table 2 and additional physical constraints to demonstrate the capabilities of the code. These constraints include three switching times for time-variant controls, with a minimum signal variation of $\mathbf{u_1}$ and $\mathbf{u_2}$ set at 0.015 $[\mathrm{h}^{-1}]$ and 3.0 $[\mathrm{g}\cdot\mathrm{L}^{-1}]$, respectively. Switching and sampling times are required to have a minimum interval of 15 minutes, and additional sampling is prohibited during the first and last 15 minutes of the experimental campaign. Not only the signal levels of controls but also the switching times and sampling times are part of the design decisions in the MBDoE-PP optimization problem.

The design decisions for the MBDoE problem are obtained using a global-local joint optimisation algorithm that employs Differential Evolution for the initial exploration of the design space, followed by Sequential Quadratic Programming (SQP) with a trust-region method for refinement. These results are illustrated in Figures 4a and 4c, showing that all obtained results comply with the previously enforced physical constraints of the system.

By simulating the experiments and complementing the observations matrix, steps 4 and 5 are repeated. The model is successfully identified using only two MBDoE-designed experiments when the t-value for all parameters exceeds the reference threshold of 2.02, while predictivity remains high with R^2=0.98. Figures 4b and 4d illustrate the model's behaviour in comparison with the experimental data, and Figure 3 summarizes the accuracy and precision achieved with this approach after the identification procedure.

The MBDoE-assisted experimental campaign improves the likelihood of identifying parameters previously considered insignificant or difficult to estimate with acceptable precision. These estimations are not only precise but also accurate, closely aligning with the true parameter values. However, in cases of limited experimental budgets or when restructuring the system or reproducing samples is infeasible, estimability analysis can help identify a predictive model that remains acceptably precise.

CONCLUSION

We present a new Python library (MIDDoE) designed to perform essential model identification steps, including rival model screening, parameter estimation, uncertainty analysis, and model-based design of experiments (MBDoE). MIDDoE offers a modular and flexible workflow that accounts for process constraints and solver flexibilities, making it suitable for various physical systems and computational platforms. Part of these capabilities, aimed at improving the precision and accuracy of estimations, along with the application of estimability analysis and model-based design of new experiments, are demonstrated using an in-silico case study.

ACKNOWLEDGEMENTS

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.

REFERENCES

1. Franceschini G., Macchietto S. Model-based design of experiments for parameter precision: state of the art. Chem Eng Sci 63:4846–4872 (2008). https://doi.org/10.1016/j.ces.2007.11.034
2. Hart W. E., Watson J. P., Woodruff D. L. Pyomo: modeling and solving mathematical programs in Python. Math Program Comput 3:219–260 (2011). https://doi.org/10.1007/s12532-011-0026-8
3. Klise K. A., Nicholson B. L., Staid A., Woodruff D. L. Parmest: parameter estimation via Pyomo. Muñoz SG, Laird CD, Realff MJ, Eds. Proceedings of the 9th International Conference on Foundations of Computer-Aided Process Design 47:41–46 (2019). https://doi.org/10.1016/B978-0-12-818597-1.50007-2
4. Wang J., Dowling A. W. Pyomo.DOE: an open-source package for model-based design of experiments in Python. AIChE J 68:e17813 (2022). https://doi.org/10.1002/aic.17813
5. Saltelli A., Ratto M., Tarantola S., Campolongo F. Sensitivity analysis for chemical models. Chem Rev 105:2811–2827 (2005). https://doi.org/10.1021/cr040659d
6. Wu S., McLean K. A. P., Harris T. J., McAuley K. B. Selection of optimal parameter set using estimability analysis and MSE-based model-selection criterion. Int J Adv Mechatron Syst 3:188–197 (2011). https://doi.org/10.1504/IJAMECHS.2011.042615
7. Chen B. H., Asprey S. P. On the design of optimally informative dynamic experiments for model discrimination in multiresponse nonlinear situations. Ind Eng Chem Res 42:1379–1390 (2003). https://doi.org/10.1021/ie0203025
8. Espie D., Macchietto S. The optimal design of dynamic experiments. AIChE J 35:223–229 (1989). https://doi.org/10.1002/aic.690350206