Digital Discovery

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal

Cite this: DOI: 10.1039/d5dd00526d

Gradient-enhanced neural networks for model parameter estimation applied to flow chemistry automated platforms

Francisco Bolaños-García, * Jean-Marc Commenge and Laurent Falk

The acceleration of chemical process development through flow chemistry depends on obtaining reliable kinetic models. Model-based design of experiments (MBDoE) has been successfully applied with flow chemistry platforms for estimating reaction rate parameters for low-complexity models. However, its use with dynamic experiments involving computationally expensive models remains challenging, particularly when experimental conditions must be suggested in real time. Surrogate models can approximate complex models, but often lack the ability to reproduce the derivatives of the original model, which are essential for MBDoE as a gradient-based approach. This work investigates the use of gradient-enhanced neural networks as surrogate models for parameter estimation within an MBDoE framework. By incorporating gradient information, the surrogate model is able to reproduce the local sensitivity structure of the original first-principles model, ensuring predictive accuracy for both output values and gradients needed for parameter estimation. A case study on competitive–consecutive reactions demonstrates that artificial neural networks (ANNs) that were trained with gradient information can be used for parameter estimation, while substantially reducing computational cost by a factor of approximately 200 000. This enables sequential MBDoE suitable for real-time applications.

Received 25th November 2025
Accepted 19th May 2026

DOI: 10.1039/d5dd00526d

rsc.li/digitaldiscovery

1 Introduction

Nowadays, laboratories seek to automatize their workflows.[1] High-throughput experimentation (HTE) and flow chemistry platforms have emerged as essential tools for exploring broad chemical spaces and fine-tuning reaction conditions.[2] The goal is to intensify the data-generation process, which is useful for model development and efficient exploration while reducing the time and resources required. Flow chemistry platforms are well-suited for sequential experimental planning,[3] and a fully automated system can be achieved by integrating algorithms that perform model fitting using real-time data from process analytical tools (PATs),[4] enabling the development of a complete closed-loop workflow.

Identifying an appropriate model structure and estimating the values of model parameters are essential steps for the development of process engineering models for different system. In reactor design, for example, kinetic models that describe the chemical reaction should be obtained at an early stage, as they are fundamental for further scale-up, defining control strategies, and optimization. Systematic procedures, known as model-based design of experiments (MBDoE), have therefore been proposed and applied for model discrimination[5,6] and parameter estimation[7–9] with the objective of performing more informative experiments. Examples of these strategies in flow chemistry include the estimation of reaction rate parameters under steady-state conditions for single reactions, such as the Diels–Alder reaction,[10] and for multistep reactions, such as a nucleophilic aromatic substitution (S_NAr).[11] They have also been applied in batch-like flow platforms, where flow ramps are used to extract kinetic information, as demonstrated for the esterification of benzoic acid with ethanol.[12] However, there have been no reports of MBDoE applied to platforms using autosamplers and injection loops to intensify the experimental workflow,[13] even though this approach could reduce reagent consumption and lower both cost and time in chemical process development. In these systems, small volumes of reagents are sequentially injected into a reactor whose volume is significantly larger than the injected volume. As a result, the reactor operates under transient conditions.

To deploy sequential planning on automated platforms, some open-source tools have already been designed, for example EFCOSS,[14] that provides a modular environment coupling numerical simulation codes with optimization packages. Pyomo.DOE[15] formulates the MBDoE problem as a stochastic program and uses nonlinear sensitivity analysis. Another framework is GPdoemd,[16] which trains Gaussian process (GP) surrogate models to support model discrimination when candidate models correspond to computationally expensive black-box simulators and gradient information with respect to model parameters is not readily available. In that framework,

Université de Lorraine, CNRS, LRGP, Nancy F-54000, France. E-mail: francisco.bolanos-garcia@univ-lorraine.fr

GP surrogates are constructed from sampled model evaluations and used to evaluate design criteria based on their predictive distributions.

In contrast, the focus of the present work is on parameter estimation for first-principles models, where parameter sensitivities can be computed, but the integration of MBDoE with self-guided flow chemistry setups requires fast numerical optimization. During the procedure, a parameter estimation is performed using the available measurements, while a second optimization problem determines the experimental conditions that maximize the information content of subsequent experiments.[17] When model evaluations are computationally intensive, these optimization steps become a bottleneck, limiting MBDoE deployment in automated platforms by constraining the speed at which new experiments can be executed. A solution to this was presented by Friso and Galvanin,[18] where an optimization-free procedure is performed in which a relative information value is used to compare possible experimental conditions. However, for the chosen design to be near the unknown true optimal design depends on the pre-defined set of candidate experiments.

In process systems engineering, surrogate models have emerged as a tool to tackle complex models while saving computational time and resources.[19] Within experimental design frameworks, Artificial Neural Networks (ANNs) have been employed to identify suitable kinetic models from experimental data,[20,21] showing the potential of using surrogate models for model building. However, these applications have typically not extended to parameter estimation, which requires solving an inverse problem to find model parameters from the measurements. Other methodologies have used physics-informed neural networks (PINNs). With them, a simultaneous training of the surrogate model parameters and the kinetic parameters is performed,[22] which can be challenging if not enough data are available.

One strategy to improve the training of neural networks consists in including gradient information, as matching the Jacobian matrix provides additional information per training sample and can be interpreted as a form of data augmentation.[23,24] The idea is to apply surrogate models that are trained not only with the model output, but also using its corresponding derivatives. Named Sobolev-trained neural networks,[23] they benefit from current libraries, *e.g.* Pytorch,[25] that compute these gradients efficiently through automatic differentiation (AD). Previously, gradient-enhanced ANNs needed the explicit derivation of the gradient,[26,27] limiting their application. These types of models have already been applied to process engineering for gradient-based optimization,[28] where they have been proven to be beneficial in scenarios where small data sets are available.

Gradients also play a crucial role in MBDoE procedures, so their calculation should be accurate. The sensitivities of the model with respect to its parameters, captured by the gradients, are fundamental for defining the design criteria. In addition, gradients are used within the optimization routine that seeks to optimize these criteria.[17]

Motivated by this, the present work explores the use of ANNs enhanced by derivative training for parameter estimation through sequential planning, with the aim of allowing the deployment of such a strategy in automated platforms that must handle computationally expensive model evaluations. To the authors' knowledge, this work represents the first application of gradient-enhanced surrogate models to sequential MBDoE for parameter estimation with experiments under transient conditions. The following section introduces the background of the MBDoE procedure, followed by a description of the neural network enhancement with gradient information illustrated through an example, and finally a case study where parameters are estimated from *in silico* flow chemistry experiments under unsteady conditions.

2 Background: model based design of experiments

A model built from experimental data is defined as a function that predicts the observed output as a function of the experimental conditions and parameters. Mathematically, this is expressed as

$$\hat{\mathbf{y}} = f(\boldsymbol{\xi}, \boldsymbol{\theta}) \tag{1}$$

where $\boldsymbol{\xi}$ denotes the vector of operating conditions inside the design space Ξ, defined by equipment constraints or by a region of interest based on prior knowledge, and $\boldsymbol{\theta}$ denotes the vector of model parameters inside the model parameter space Θ. For instance, in a flow reactor, $\boldsymbol{\xi}$ may include temperature, residence time, and inlet concentrations, while $\boldsymbol{\theta}$ may represent unknown kinetic constants to be estimated. In this work, model parameters refer to physically meaningful parameters (*e.g.*, kinetic constants), as opposed to non-physical parameters like the weights and biases of data-driven models. This function can be a first-principles model, denoted by $f()$, or a data-driven model, such as a neural network, denoted by $f_{\mathrm{NN}}()$. Both models take the operating conditions and physically meaningful model parameters as inputs and return predicted measurable outputs such as concentrations or yields.

From a set of preliminary experiments, hypotheses can be made to propose a set of plausible models that describe the observed behaviour. After selection of model structures, it is important to be sure that they are identifiable and distinguishable.[29] If, from the data available, it is not possible to discriminate between the best two models, a sequential procedure to produce more observations to maximize discrepancy between predictions can be performed.[30]

Once we assume that the selected model structure is adequate, experiments that increase the information available for the estimation of parameters must be executed. This part of the MBDoE is the focus of this work. The definition of the parameter space Θ should be based on some prior knowledge of the system behaviour from the preliminary experiments. In order to obtain a parameter estimation $\hat{\boldsymbol{\theta}}$ from experimental data, an estimator of the fit quality of the model with respect to observations should be defined. The maximum likelihood estimator (MLE) is mostly used due to its consistency, efficiency and asymptotic normality under modest assumptions.[31] This

estimator maximizes the likelihood function $\ell(\mathbf{y}|\boldsymbol{\theta})$, defined as the joint probability of observing the experimental data $\mathbf{y}_e (e = 1, \ldots, N_e)$ given the N_p model parameters. To properly quantify it, the uncertainty in the measurements must be taken into account with the variance-covariance matrix of the experimental errors, Σ_y. Its diagonal elements represent the variance of the uncertainty associated with each of the N_y measurements within the eth experiment and the off-diagonal elements represent the covariance between pairs of them.[32,33]

$$\ln \ell(\mathbf{y}|\boldsymbol{\theta}) = -\frac{N_e N_y}{2}\ln 2\pi - \frac{N_e}{2}\ln\left(\det\Sigma_y\right) - \frac{1}{2}\sum_{e=1}^{N_e}(\mathbf{y}_e - \hat{\mathbf{y}}_e)^T \Sigma_y^{-1}(\mathbf{y}_e - \hat{\mathbf{y}}_e) \quad (2)$$

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \ln \ell(\mathbf{y}|\boldsymbol{\theta}) \quad (3)$$

s.t. $\hat{\boldsymbol{\theta}} \in \Theta$.

To improve the estimation with further experiments, it is important to quantitatively assess the influence of the parameter values on the outputs of the proposed model. This is represented by the sensitivity matrix $\mathbf{S}[N_y \times N_p]$, which is a local measurement depending on current parameter values and experimental conditions. For example, it captures how variations in a rate constant affect predicted outlet concentrations at a given temperature and residence time.

$$\mathbf{S}_e\left(\widehat{\boldsymbol{\theta}}, \boldsymbol{\xi}_e\right) = \begin{bmatrix} \frac{\partial \hat{y}_1}{\partial \theta_1} & \cdots & \frac{\partial \hat{y}_1}{\partial \theta_{N_p}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}_{N_y}}{\partial \theta_1} & \cdots & \frac{\partial \hat{y}_{N_y}}{\partial \theta_{N_p}} \end{bmatrix} \quad (4)$$

From this, a square matrix $N_p \times N_p$ known as the Fisher Information Matrix (FIM), $\mathbf{F}$, can be computed with eqn (5). This step is crucial as the variance-covariance matrix of the estimated parameters $\Sigma_{\hat{\boldsymbol{\theta}}}$ can be approximated using the inverse of the FIM evaluated at $\hat{\boldsymbol{\theta}}$,$\boldsymbol{\xi}$,[33] useful to determine the confidence region of the parameters and the correlation between them. In this context, experiments that produce stronger sensitivity of concentrations to kinetic parameters result in a more informative FIM. As stated above, the MLE is asymptotically Gaussian and unbiased. This means, with $\boldsymbol{\theta}^*$ being the vector of true parameters, the distribution of $\hat{\boldsymbol{\theta}}$ tends to $\mathcal{N}(\boldsymbol{\theta}^*, \mathbf{F}(\boldsymbol{\theta}^*)^{-1})$ as $N_e \to \infty$.[31]

In practice, however, only a limited number of experiments are available. As a result, this asymptotic approximation may not hold exactly, and its accuracy depends on several factors: the proximity of the estimated parameters to their true values, the degree of model nonlinearity with respect to the parameters, and the signal-to-noise ratio of the measurements. Its widespread use in MBDoE is motivated by its computational tractability and its effectiveness for comparing alternative experimental designs, rather than its ability to provide an exact quantification of uncertainty,[33] remaining as a practical and widely used tool for ranking candidate experiments before performing them.

$$\mathbf{F}_{N_e}\left(\widehat{\boldsymbol{\theta}}, \boldsymbol{\xi}\right) = \sum_{e=1}^{N_e} \mathbf{S}_e\left(\widehat{\boldsymbol{\theta}}, \boldsymbol{\xi}_e\right)^T \Sigma_y^{-1} \mathbf{S}_e\left(\widehat{\boldsymbol{\theta}}, \boldsymbol{\xi}_e\right) \quad (5)$$

$$\Sigma_{\hat{\boldsymbol{\theta}}} \approx \mathbf{F}_{N_e}(\hat{\boldsymbol{\theta}}, \boldsymbol{\xi})^{-1} \quad (6)$$

To facilitate sequential experimental design, a scalar function of the FIM is typically introduced to quantify the information content associated with a given set of operating conditions. This scalar metric serves as the design criterion and is maximized at each step to inform the selection of subsequent experiments. This enables ranking candidate operating conditions (*e.g.*, different temperatures or flow rates) according to their expected contribution to parameter estimation. The optimal design is, therefore, the one that maximizes the selected criterion $\psi(\cdot)$.

$$\boldsymbol{\xi}_{N_e+1}^{\mathrm{OPT}} = \underset{\boldsymbol{\xi}_{N_e+1}}{\operatorname{argmax}} \psi\left(\mathbf{F}_{N_e} + \mathbf{S}_{N_e+1}\left(\widehat{\boldsymbol{\theta}}, \boldsymbol{\xi}_{N_e+1}\right)^T \Sigma_y^{-1} \mathbf{S}_{N_e+1}\left(\widehat{\boldsymbol{\theta}}, \boldsymbol{\xi}_{N_e+1}\right)\right) \quad (7)$$

s.t. $\boldsymbol{\xi}_{N_e+1}^{\mathrm{OPT}} \in \boldsymbol{\Xi}$.

Several design criteria exist to decide the best next experiment.[17] In this work, the D-optimality is employed, as it is the most widely used, where the determinant of the FIM is the scalar property to be maximized. The aim is to find the experimental conditions that minimize the uncertainty of the parameter estimation, as increasing the determinant of the FIM reduces the volume of the confidence region of the estimated model parameters. In practice, this corresponds to selecting the next set of operating conditions that is expected to give the most informative measurements for refining the kinetic parameters.

$$\psi_D(\mathbf{F}(\hat{\boldsymbol{\theta}}, \boldsymbol{\xi})) := \det(\mathbf{F}(\hat{\boldsymbol{\theta}}, \boldsymbol{\xi})) \quad (8)$$

The whole workflow is presented in Fig. 1 illustrating the iterative interplay between model evaluation, parameter estimation, and experiment selection. Stopping criteria could be either a fixed number of experiments based on a pre-defined experimental budget, or a threshold value when assessing the adequacy and reliability of the estimated parameters by, for example, comparing χ^2 and t-values with reference values from the corresponding distribution with $N_e \times N_y - N_p$ degrees of freedom.[17] The focus of our work is on parameter estimation and the subsequent design steps, which require efficient computation of sensitivities, that is, first-order derivatives of the model outputs with respect to the parameters (eqn (4)). Since analytical expressions for these derivatives are rarely available, they are often obtained numerically, a process that can become prohibitively expensive in automated platforms when using finite-difference schemes.

Overall, the workflow takes a candidate model, experimental data (*e.g.*, concentration measurements), and admissible operating ranges (*e.g.*, temperature and flow limits) as inputs and returns updated parameter estimates together with the next

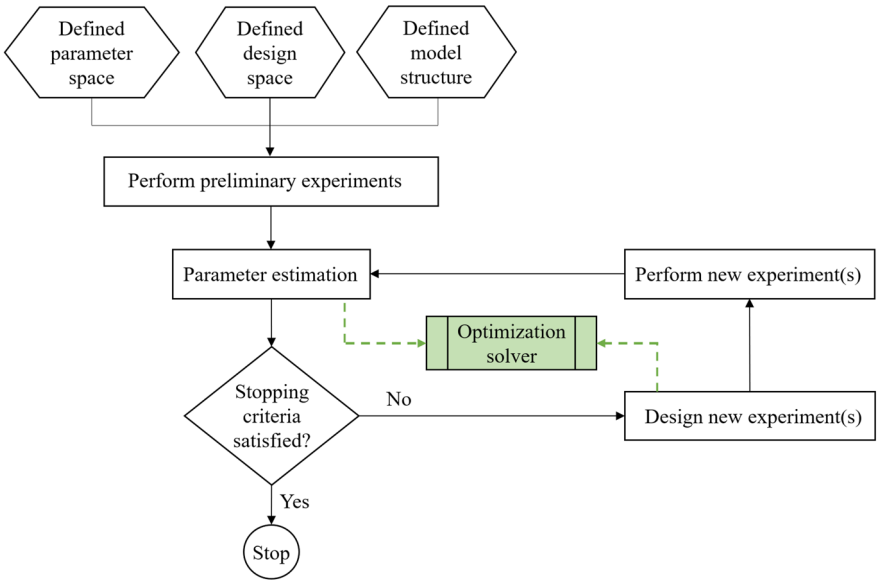

Fig. 1 MBDoE procedure focused on parameter estimation. An optimization solver is called at two key stages: to guide the design of new, informative experiments and to perform the parameter estimation.

optimal experimental conditions to be tested, guiding subsequent experiments.

3 Gradient-enhanced training of neural networks

Artificial neural networks (ANNs) are able to approximate any continuous function. Their architecture is based on multiple layers, where each one of them is formed by a number of neurons (also called perceptrons).[34] The output of a single layer is expressed as

$$\mathbf{z}_l = \phi_l(\mathbf{W}_l \mathrm{T}\mathbf{z}_{l-1} + \mathbf{b}_l),\ \forall l = 1,\ldots,L \tag{9}$$

where $z_{l-1}[N_{l-1} \times 1]$ and $\mathbf{z}_l[N_l \times 1]$ denote the input and output vectors, respectively, of the layer l, where N_l represents the number of neurons of this layer. $\mathbf{W}_l[N_{l-1} \times N_l]$ is the matrix of weights, $\mathbf{b}_l[N_l \times 1]$ is the vector of biases for the layer l, and $\phi_l()$ is the activation function of layer l. The most common activation functions for the hidden layers are the rectified linear unit (ReLU), sigmoid, and hyperbolic tangent, while a linear function is often used for the output layer in regression tasks. So, when using an ANN as a black box function $\hat{\mathbf{y}} = f_{\mathrm{NN}}(\boldsymbol{x})$, the output of the final layer $\mathbf{z}_L$ corresponds to the prediction value $\hat{\mathbf{y}}$ and the input vector $\boldsymbol{x}$ is the input $\mathbf{z_0}$ of the first hidden layer.

Different hyperparameters, such as the number of layers (L) and the number of neurons per layer ($N_l\ \forall\ l = 1, \ldots, L$), must be specified. The surrogate model is then trained on a designated training dataset of size N_s to determine the optimal weights and biases by solving an optimization problem that minimizes a chosen loss function. For regression tasks, commonly used loss functions include the mean absolute error (MAE) and the mean squared error (MSE). During training, backpropagation is performed where gradients of the loss with respect to all weights and biases are computed using the chain rule *via* automatic differentiation (AD). This AD framework can also provide the sensitivities of $f_{\mathrm{NN}}(\boldsymbol{x})$ not only during training but also when evaluating the ANN, as illustrated in Fig. 2.

$$\mathrm{MSE}(f(\boldsymbol{x}_n), f_{\mathrm{NN}}(\boldsymbol{x}_n)) := \frac{1}{N_y}\|f(\boldsymbol{x}_n) - f_{\mathrm{NN}}(\boldsymbol{x}_n)\|_2^{\,2} \tag{10}$$

If the loss function is restricted to the MSE (eqn (10)) between function outputs, the ANN can accurately predict the function values, but may perform poorly if required to yield the derivatives of the function with respect to the inputs, which are essential in gradient-based applications such as MBDoE for

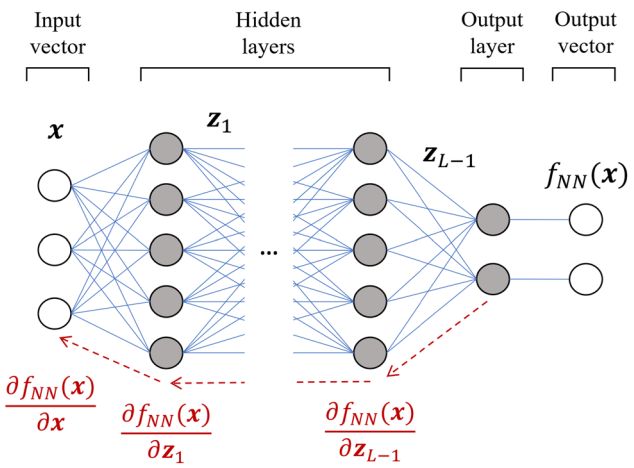

Fig. 2 Example of a fully connected artificial neural network with multiple hidden layers. Grey circles represent the neurons. The dashed red arrows illustrate the backpropagation step during training, where the gradients are computed.

parameter estimation. This limitation is significant since the surrogate model replaces the mechanistic model that provides a detailed but computationally expensive representation of the experimental setup. As shown by Czarnecki,[23] Sobolev-trained neural networks minimize the distance between the true function $f(\boldsymbol{x})$ and the surrogate model $f_{\mathrm{NN}}(\boldsymbol{x})$ in the Sobolev space by adding a loss term for each of the j elements in the set of available and relevant derivative orders $\mathcal{J}$.

$$\mathcal{L} = \frac{1}{N_s}\sum_{n=1}^{N_s}\mathrm{MSE}(f(\boldsymbol{x}_n), f_{\mathrm{NN}}(\boldsymbol{x}_n)) + \frac{1}{N_s}\sum_{j\in\mathcal{J}}\sum_{n=1}^{N_s}\lambda_j \mathrm{MSE}\left(D_{\boldsymbol{x}}{}^{j} f(\boldsymbol{x}_n), D_{\boldsymbol{x}}{}^{j} f_{\mathrm{NN}}(\boldsymbol{x}_n)\right) \quad (11)$$

To ensure that all terms in the loss function are of the same order of magnitude, Tsay[28] proposed a simple method to scale the derivative terms and select a proper value for each weight λ_j, avoiding the challenge of relying on heuristic tuning. Since it is common practice to normalize the input and output vectors before training the neural network, the derivatives used for training must be scaled consistently with eqn (12). We denote a scaled variable with an overbar symbol (*e.g.*, $\bar{\boldsymbol{x}}$). If resulting derivatives are also normalized using the min–max scaler, then $D_{\bar{\boldsymbol{x}}}{}^{j}\bar{f}(\bar{\boldsymbol{x}}_n) \in [0,1]$, and both terms in the loss function will have the same order of magnitude.

$$D_{\bar{\boldsymbol{x}}}{}^{j}\bar{f}(\bar{\boldsymbol{x}}_n) = \left.\frac{\partial^j \bar{f}}{\partial \bar{\boldsymbol{x}}^j}\right|_{\bar{\boldsymbol{x}}_n} = \frac{\mathrm{range}(\boldsymbol{x}_n)}{\mathrm{range}(f(\boldsymbol{x}_n))}\left.\frac{\partial^j f}{\partial \boldsymbol{x}^j}\right|_{\boldsymbol{x}_n} \quad (12)$$

$$\mathcal{L} = \frac{1}{N_s}\sum_{n=1}^{N_s}\mathrm{MSE}\left(\bar{f}(\boldsymbol{x}_n), f_{\mathrm{NN}}(\bar{\boldsymbol{x}}_n)\right) + \frac{1}{N_s}\sum_{j\in\mathcal{J}}\sum_{n=1}^{N_s}\mathrm{MSE}\left(\bar{D}_{\boldsymbol{x}}{}^{j}\bar{f}(\bar{\boldsymbol{x}}_n), \bar{D}_{\boldsymbol{x}}{}^{j} f_{\mathrm{NN}}(\bar{\boldsymbol{x}}_n)\right) \quad (13)$$

where the range is defined as the maximum minus the minimum value. Within the proposed framework of integrating a neural network into the standard MBDoE for parameter estimation, the input vector $\boldsymbol{x}_n$ consists of the experimental conditions and the parameter vectors $[\boldsymbol{\xi}_n, \boldsymbol{\theta}_n]$. The whole set of gradients $D_{\bar{\boldsymbol{x}}}\bar{f}$ is not needed and just the first derivative with respect to $\boldsymbol{\theta}_n$ will be computed. This avoids the need to construct complete gradient information over all inputs, providing a distinct structural advantage over other surrogate models, *i.e.*, gradient-enhanced Gaussian process (GEK) approaches, which generally require complete gradient information across all inputs to formulate stable joint covariance matrices.[35]

$$D_{\bar{\boldsymbol{\theta}}}\bar{f}\left(\boldsymbol{\xi}_n, \bar{\boldsymbol{\theta}}_n\right) = \left.\frac{\partial \bar{f}}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{\xi}_n, \bar{\boldsymbol{\theta}}_n} = \frac{\mathrm{range}(\boldsymbol{\theta}_n)}{\mathrm{range}(f(\boldsymbol{\xi}_n, \boldsymbol{\theta}_n))}\left.\frac{\partial f}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\xi}_n, \boldsymbol{\theta}_n} \quad (14)$$

$$\mathcal{L} = \frac{1}{N_s}\sum_{n=1}^{N_s}\mathrm{MSE}\left(\bar{f}(\boldsymbol{x}_n), f_{\mathrm{NN}}(\bar{\boldsymbol{x}}_n)\right) + \frac{1}{N_s}\sum_{n=1}^{N_s}\mathrm{MSE}\left(\bar{D}_{\bar{\boldsymbol{\theta}}}\bar{f}\left(\boldsymbol{\xi}_n, \bar{\boldsymbol{\theta}}_n\right), \bar{D}_{\bar{\boldsymbol{\theta}}} f_{\mathrm{NN}}\left(\bar{\boldsymbol{\xi}}_n, \bar{\boldsymbol{\theta}}_n\right)\right) \quad (15)$$

As discussed previously, the gradients of the neural network output with respect to its inputs can be obtained by AD, already available in libraries such as Pytorch (version 2.8.0). For the first-principles model, several strategies can be employed to obtain these derivatives. While analytical expressions offer exact results, they are often impractical for complex systems. Numerical finite differences provide a straightforward alternative but are computationally costly and prone to numerical error. A more systematic alternative is AD, which provides exact derivatives up to machine precision. Depending on the problem size, AD can be applied in forward or backward mode. In the context of models governed by ordinary differential equations (ODEs), the corresponding sensitivity equations (forward mode) or adjoint equations (backward mode) can be solved alongside the system dynamics to efficiently compute parameter sensitivities.[36,37]

4 Example using a gradient-enhanced neural network

To demonstrate the capabilities of a gradient-enhanced neural network, the following example considers a first-order reaction A → B with kinetic parameter k. The reaction occurs under steady-state conditions in a tubular microreactor with axial dispersion. Calculations were done assuming a tube of 5 mL with an internal diameter of 0.75 mm, a system that can be perfectly used in flow chemistry applications. In non-ideal tubular reactors, the Peclet (Pe) number must be determined to estimate the axial dispersion. For this, assuming laminar flow, the dispersion coefficient **D** was calculated using the Aris–Taylor equation (eqn (16)). Then, for first-order reactions, eqn (18) is used to determine the conversion X_A at the outlet of the tubular reactor.[38]

$$\mathbf{D} = \mathcal{D} + \frac{u^2 {d_t}^2}{192\mathcal{D}} \quad (16)$$

$$\mathrm{Pe} = \frac{uL}{\mathbf{D}} \quad (17)$$

$$X_A = 1 - \frac{C_A}{C_{A,0}} = 1 - \frac{4a\exp(\mathrm{Pe}/2)}{(1+a)^2\exp(a\mathrm{Pe}/2) - (1-a)^2\exp(-a\mathrm{Pe}/2)} \quad (18)$$

$$a = \sqrt{1 + \frac{4k\tau}{\mathrm{Pe}}} \quad (19)$$

where u denotes the linear velocity, d_t is the internal diameter of the reactor, $\mathcal{D}$ is the molecular diffusion, L is the length of the reactor, τ is the space time, and k is the kinetic constant. It was assumed that $\mathcal{D} = 1 \times 10^{-9}$ m^2 s^{-1}, a typical value for liquids. This first-principles model will serve as the reference to be replaced by a surrogate model.

A dataset $\{(\tau_n, k_n, X_{An}, \mathrm{d}X_{An}/\mathrm{d}k_n)\}_{n=1}^{100}$ was created by a standard Latin Hypercube Sampling (LHS) over the domain of the experimental conditions, *i.e.*, the space time ($\tau \in$ [10 s, 90 s]), and the domain of model parameters, *i.e.*, the kinetic constant

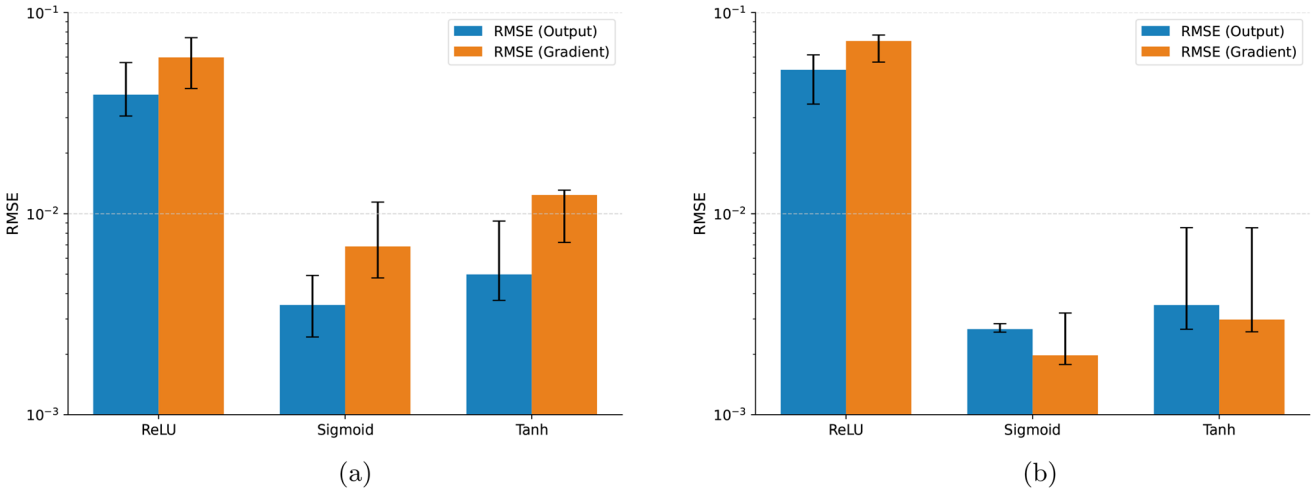

Fig. 3 Median RMSE obtained from 10 independent neural network trainings for each activation function. Error bars indicate the computed interquartile range. (a) Standard training and (b) gradient-enhanced training.

($k \in [0.001\ \mathrm{s}^{-1}, 0.1\ \mathrm{s}^{-1}]$). This dataset was then randomly partitioned into three subsets: a training set (70%), a validation set (15%), and an independent test set (15%). The architecture of the surrogate model is a fully connected neural network with two inputs, τ and k, one hidden layer of 10 neurons, and a single output neuron for X_A. The number of neurons in the hidden layer (10) was determined through preliminary testing, as increasing this number did not result in any further improvement in the surrogate model's predictive accuracy. For scaling the input vector, the boundaries of the design space were used as min–max values, and not the min–max values from the sampled dataset. Two surrogate models were trained by minimizing the loss function using the Adam optimizer: one using only the function values X_A and another using both the function values and the derivatives $\mathrm{d}X_A/\mathrm{d}k$. Training was performed for a maximum of 20 000 epochs with early stopping based on the validation loss, using a patience of 500 epochs. The trained network weights and biases corresponding to the lowest validation loss were retained, and final performance metrics were computed on the independent test set.

A comparative analysis was performed to determine the most effective activation function for the ANN architecture. The evaluation involved training 10 networks, each with a random weight initialization. The Root Mean Squared Error (RMSE) of the model output and of the gradient obtained through AD of the ANN was recorded. The median RMSE across the 10 trained networks was used as a performance metric. This comparison was carried out for both types of surrogate models, the standard trained and the enhanced with gradient information.

The median RMSE reported in Fig. 3 exhibits the performance for three different activation functions: ReLU, sigmoid and tanh. Under standard training, the sigmoid and tanh functions yielded similar performance, although the tanh activation showed greater variability. A common characteristic across all three activation functions was a greater error in predicting the gradient than in predicting the function value itself. When incorporating gradient information into the surrogate model training, the performance for the sigmoid and tanh functions improved. In contrast, the ReLU function consistently produced the highest RMSE and showed no significant improvement with gradient-informed training, an expected outcome given its non-differentiability at zero. Regarding computational cost, the inclusion of the gradient loss term did not lead to an increase in training time. In contrast, the gradient-enhanced training exhibited a slightly lower average training time (10 s) compared to standard training (12 s).

Because the sigmoid activation achieved the lowest RMSE for both the model output X_A and its gradient with respect to k, it was selected as the preferred choice for further analysis. An ablation study on the gradient loss weight was conducted to assess its influence on model performance. In addition to the baseline formulation (eqn (15)), weights of 0.5 and 2.0 for the gradient loss were evaluated. Both alternatives resulted in higher average total MSE values, 1.78×10^{-4} and 1.02×10^{-4}, respectively, compared to 3.91×10^{-5} obtained with the baseline weighting. These results indicate that the chosen weighting provides a near-optimal balance between fitting the function value and its gradient.

Fig. 4 compares the true function and gradient values with the predictions from the standard trained ANNs with the sigmoid activation function. The comparison is performed at $\tau = \{10\ \mathrm{s}, 50\ \mathrm{s}, 90\ \mathrm{s}\}$, which correspond to the lower boundary, centre, and upper boundary of the operating range used to generate the training dataset. The prediction of the conversion X_A is accurate for all three space time values and over the whole range of k values. This is not the case for the prediction of $\mathrm{d}X_A/\mathrm{d}k$ in the lower region of k values of the training set, especially at both boundary values of τ, where the predictions are less accurate. This is a common behaviour of ANNs and why they tend to perform badly when extrapolating. Fig. 5 shows the improvement on both the conversion and its first-order derivative prediction when adding the gradient information during training, particularly increasing accuracy near the lower values of k. This means that using gradient-enhanced training yielded

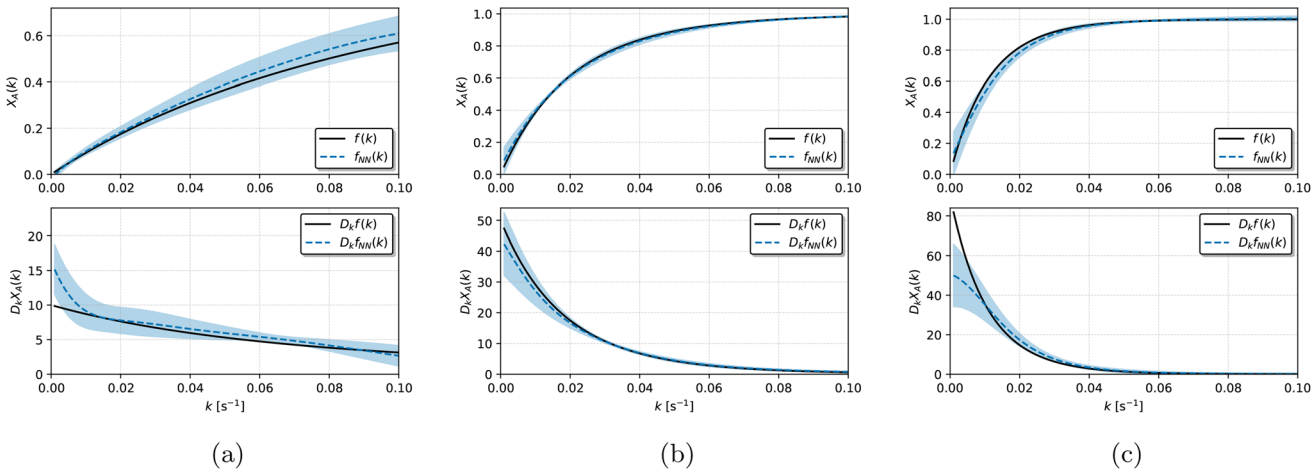

Fig. 4 Predictive accuracy for the conversion $X_A(k)$ and its sensitivity with respect to the kinetic constant $D_k X_A(k)$ obtained using ANNs trained without gradient information. The solid black line represents the true model output, the blue dashed line indicates the mean prediction of 10 ANNs, and the shaded area shows the mean ± one standard deviation. (a) $\tau = 10$ s, (b) $\tau = 50$ s and (c) $\tau = 90$ s.

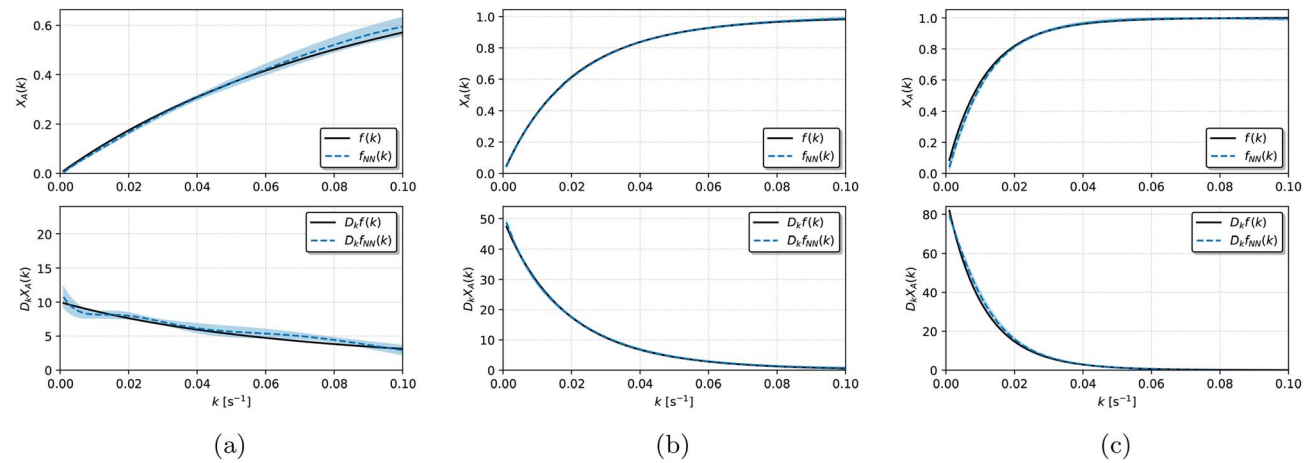

Fig. 5 Predictive accuracy for the conversion $X_A(k)$ and its sensitivity with respect to the kinetic constant $D_k X_A(k)$ obtained using ANNs trained with gradient information. The solid black line represents the true model output, the blue dashed line indicates the mean prediction of 10 ANNs, and the shaded area shows the mean ± one standard deviation. (a) $\tau = 10$ s, (b) $\tau = 50$ s and (c) $\tau = 90$ s.

a surrogate model that will behave more like the first-principles model, a relevant advantage in gradient-based applications.

Then, if the surrogate model is used by a MBDoE framework to determine the most informative experimental conditions for the estimation of k, the sensitivities are computed. As just one parameter is being estimated, the FIM becomes a scalar value $\mathcal{I}(\boldsymbol{\theta})$. This value was determined assuming a measurement error in the observed conversion, $\varepsilon \sim \mathcal{N}(0, {\sigma_e}^2)$, with $\sigma_e = 0.05$. Fig. 6 presents the predicted and true Fisher information for all possible values of the space time with $k = 0.022$ s^{-1}, a value for which $\mathrm{d}X_A/\mathrm{d}k > 0$ over the whole design space so the conversion is sensitive to variations in the kinetic constant. When comparing between the standard and the gradient-enhanced training, the latter is more accurate in the prediction of the Fisher information. This is relevant when performing an experimental planning to find the optimal value of τ that reduces the uncertainty on our estimation of k. Based on the first-principles model (full black line), an optimal space time for an experiment is close to 47 s, and the same decision would be taken if the gradient-enhanced ANN is used. In contrast, an ANN with a standard training would hesitate between the values of 47 s and 90 s, which could misguide a sequential experimental strategy toward less informative experiments.

5 Case study: competitive consecutive reactions in a discrete injection flow platform

In chemical development, reducing reagent consumption is desirable to lower both cost and time required. Slug flow platforms address this by isolating small reactive volumes (slugs) between gas bubbles, limiting axial dispersion and enhancing radial mixing by recirculation within the slug.[39] However, their

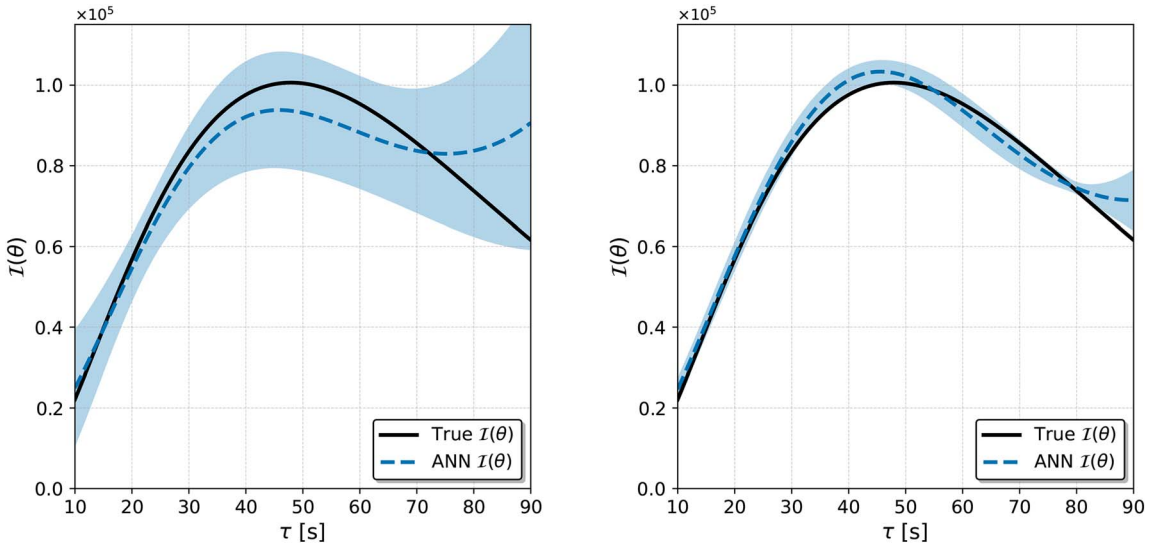

Fig. 6 Comparison between the true Fisher information (solid black line) and the prediction made by 10 randomly initialized neural networks (blue dashed line). The shaded area represents the mean ± one standard deviation. (a) Standard training and (b) gradient-enhanced training.

automation across a broad design space remains challenging. An alternative approach is to inject a discrete reactive volume directly into a continuous solvent stream (a rectangular pulse of reactant concentration), without any gas separator.[13] The resulting axial dispersion produces a concentration gradient along the microreactor.

Gaining knowledge on model parameters from this kind of system is not trivial, as neglecting the effect of dispersion will lead to an erroneous estimation of the kinetic constants. A detailed model is then needed to account for the real physical behaviour and precisely estimate the intrinsic values. The proposed case study consists of two competitive–consecutive reactions, the iodination of L-tyrosine. In this scheme, L-tyrosine (A) reacts with iodine in water (B) to form 3-iodotyrosine (R) and then R can further react with B to give 3,5-diiodotyrosine (S). Both reactions are second order,[8] with reaction rates denoted by r and kinetic constants denoted by k.

$$\mathrm{A} + \mathrm{B} \xrightarrow{k_1} \mathrm{R}$$

$$\mathrm{R} + \mathrm{B} \xrightarrow{k_2} \mathrm{S}$$

$$r_1 = k_1 C_\mathrm{A} C_\mathrm{B} \tag{20}$$

$$r_2 = k_2 C_\mathrm{R} C_\mathrm{B} \tag{21}$$

5.1 First-principles model

A first-principles model describing this system was built, and simulations were performed to generate training data for the surrogate neural network. The model inputs consist of the experimental conditions $\boldsymbol{\xi}$, corresponding to the space time τ and the inlet concentration ratio of tyrosine to total concentration, and the model parameters $\boldsymbol{\theta}$, *i.e.*, both kinetic constants k_1 and k_2. The mass balances over the four species along the tubular reactor with axial dispersion yields a set of partial differential equations:

$$\frac{\partial C_i(z,t)}{\partial t} = \mathbf{D}\frac{\partial^2 C_i(z,t)}{\partial z^2} - u\frac{\partial C_i(z,t)}{\partial z} + \nu_{i,1} r_1 + \nu_{i,2} r_2 \qquad \forall i \in \{\mathrm{A},\ \mathrm{B},\ \mathrm{R},\ \mathrm{S}\} \tag{22}$$

With the following boundary conditions:

$$C_i(z=0,t) = \begin{cases} C_{i,0} & 0 \le t \le t_\mathrm{inj} \\ 0 & t > t_\mathrm{inj} \end{cases} \tag{23}$$

$$\left.\frac{\partial C_i(z,t)}{\partial z}\right|_{z=L} = 0 \tag{24}$$

where C_i is the concentration and ν_i is the stoichiometric coefficient of the ith chemical species, t is the time after injection of the reactive volume started, t_inj is the injection duration, z is the spatial coordinate, and u is the linear velocity.

The model output $f(\boldsymbol{\xi},\boldsymbol{\theta})$ corresponds to a unique experimental measurement, the area under the curve of iodine concentration at the reactor outlet as a function of time (eqn (25)). Iodine concentration was selected as the measurable variable because it can be directly monitored in practice using an inline UV spectrometer.

$$f(\boldsymbol{\xi},\boldsymbol{\theta}) = \int_0^{3\tau} C_\mathrm{B}(z=L,t)\mathrm{d}t \tag{25}$$

To solve the set of PDEs numerically, we used the method of lines. The axial coordinate z was discretized into 1000 uniformly spaced nodes, and both first- and second-order spatial derivatives were approximated using centred finite differences. The

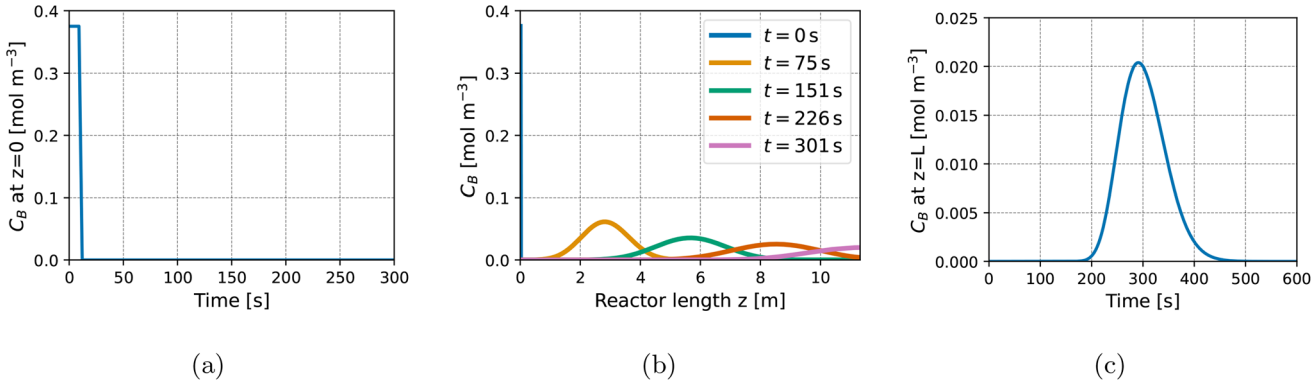

Fig. 7 Transient behaviour of species B in a tubular reactor with axial dispersion. (a) Rectangular-wave inlet concentration of B. (b) Concentration of B along the reactor length at selected times. (c) Concentration of B at the reactor outlet.

discretization converted the PDE into a system of ODEs, solved with a Runge–Kutta method from the SciPy library (version 1.16.2) in Python (version 3.11.7).[40]

For the simulations, a 5 mL tubular reactor with an internal diameter of 0.75 mm and a length of 11.32 m was considered. A molecular diffusion of 1×10^{-9} m^2 s^{-1} was assumed, and the dispersion coefficient **D** was determined as described in the previous section. The total concentration, $(C_{A,0} + C_{B,0})$, was held constant at 0.75 mol m^{-3}, which defined the height of the rectangular injection profile. The ratio t_{inj}/τ was fixed at 0.04, corresponding to a 200 μL injection volume. As an example, Fig. 7 shows the simulated behaviour of species B for a space time of 5 min, an injection duration of 12 seconds, an injection concentration of $C_B = 0.375$ mol m^{-3}, $k_1 = 0.05$ m^3 s^{-1} mol^{-1}, and $k_2 = 0.02$ m^3 s^{-1} mol^{-1}. The simulation starts with the discrete injection of the reagents (Fig. 7a). The initial concentration of B decreases along the reactor length due to the combined effects of reaction and dispersion (Fig. 7b). The outlet concentration of B is tracked as a function of time (Fig. 7c), mimicking the response that could be measured with an inline UV detector. The area under this curve is then used as the single output of our first-principles model.

5.2 ANN training and testing

A Latin hypercube sampling was performed to generate 100 full model simulations to train the ANNs. The dataset was generated by varying $\tau \in$ [1 min, 10 min], the initial concentration $C_{A,0} \in$ [0.0375 mol m^{-3}, 0.7125 mol m^{-3}], and both reaction rate constants, k_1 and k_2, within the range [0.01 m^3 s^{-1} mol^{-1}, 0.1 m^3 s^{-1} mol^{-1}]. Simulations were run to estimate the value of the peak area for B at the outlet of the reactor (eqn (25)), and the values of $\partial f(\boldsymbol{\xi},\boldsymbol{\theta})/\partial\theta_p$ were determined by the central difference method. The resulting peak areas ranged from 0.01 to 15.02 mol s m^{-3}. The dataset was randomly divided into three subsets: 70% for training, 15% for validation, and 15% reserved as an independent test set. The data used to train the ANN should not be confused with the experimental data set used for parameter estimation.

The architecture of the neural network consisted of 4 inputs (τ, $C_{A,0}$/0.75, k_1, and k_2), 1 hidden layer of 16 neurons, and one output layer corresponding to the area under the curve of B. The hidden layer size of 16 neurons was selected based on preliminary tests, as increasing the number of neurons beyond this value did not lead to any improvement in the surrogate model's accuracy. A training for 10 000 epochs with early stopping (a patience parameter of 500) was performed. Both sigmoid and tanh functions were tested by training 10 neural networks with a randomly initialized set of weights.

Fig. 8 shows the RMSE obtained with both activation functions. Under standard training, the tanh activation achieved lower RMSE values than the sigmoid function for both output and gradient predictions. The higher accuracy of tanh in this case may be attributed to its resemblance to the error function (erf), which is commonly used to describe cumulative distributions such as the area under the curve after a pulse injection. Incorporating gradient information into the training reduced RMSEs in all cases relative to standard training, except for the output predictions with tanh, leading to similar performance between the two activation functions. The advantage provided by the resemblance between tanh and erf during standard training appears to diminish once gradient terms are included in the loss function. In addition, consistent with observation in Section 4, the inclusion of gradient information did not introduce additional computational cost. The gradient-enhanced models converged with an average training time of 4.2 s compared to 5 s for the standard approach.

Since the tanh outperformed the sigmoid activation in approximating the true function values under standard training, it was selected for further comparison with the gradient-enhanced neural network. Alternative gradient loss weights of 0.5 and 2.0 were tested, yielding higher average total MSE values (2.51×10^{-3} and 3.01×10^{-3}, respectively) than the baseline value of 2.06×10^{-3} obtained with a weight of 1.0.

From each of the two sets of 10 ANNs, one with standard training and one with gradient-enhanced training, the model with the lowest total RMSE (sum of output and gradient errors) was selected. These two surrogate models were then compared to evaluate their predictive ability and performance when applied to a MBDoE framework for parameter estimation. Both the best standard and gradient-enhanced networks accurately

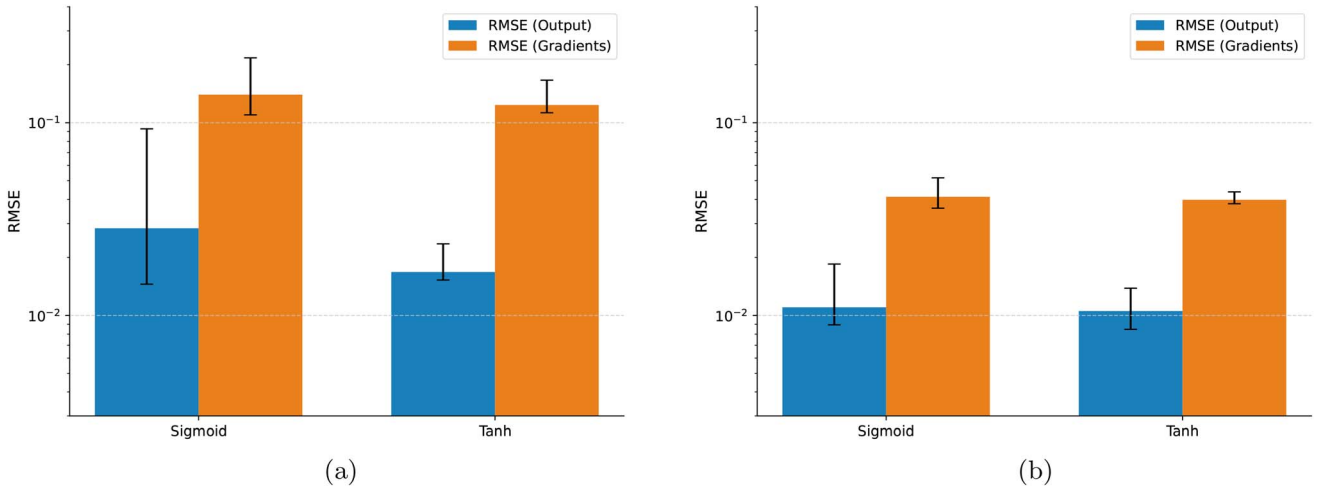

Fig. 8 Median RMSE obtained from 10 independent neural network trainings for each activation function. Error bars indicate the computed interquartile range. (a) Standard training and (b) gradient-enhanced training.

predicted the peak area of B, as shown in the parity plots in Fig. 9, indicating that the inclusion of gradient information does not compromise the accuracy of output predictions.

In contrast, differences arise when comparing gradient predictions. The parity plots from Fig. 10 and 11 demonstrate that the gradient-enhanced network provides an improved accuracy for the derivatives w.r.t. both kinetic parameters k_1 and k_2. In addition, the sign of the gradients is distinguished with colours, as all derivatives should be negative for physical consistency: indeed, increasing either kinetic constant while holding the other fixed increases the consumption of B and reduces the peak area at the reactor outlet, implying a negative derivative. In the standard training case (Fig. 10), several predicted gradients were positive, which is physically unrealistic. Incorporating gradient information (Fig. 11) reduced the occurrence of such predictions, and the few remaining positive values were close to zero, despite no hard constraints being imposed during training. Therefore, gradient-enhanced training not only improves the accuracy of gradient predictions but also mitigates the occurrence of gradients that lack physical meaning. As an additional baseline, a standard GP surrogate trained on the same dataset yielded a total MSE (1.89×10^{-2}) and parity plots comparable to those of the standard ANN (Fig. S.1), indicating similar limitations in reproducing parameter sensitivities.

5.3 ANNs integrated into an MBDoE sequential framework

As a final evaluation, the surrogate models were integrated within a MBDoE framework. To initialize the *in silico* experimental planning, the first-principles model was simulated using the kinetic parameters $k_1 = 0.0562$ m^3 mol^{-1} s^{-1} and $k_2 = 0.0193$ m^3 mol^{-1} s^{-1}, previously reported.[8] A random experimental error, $\varepsilon \sim \mathcal{N}(0, {\sigma_e}^2)$, was added to the computed area

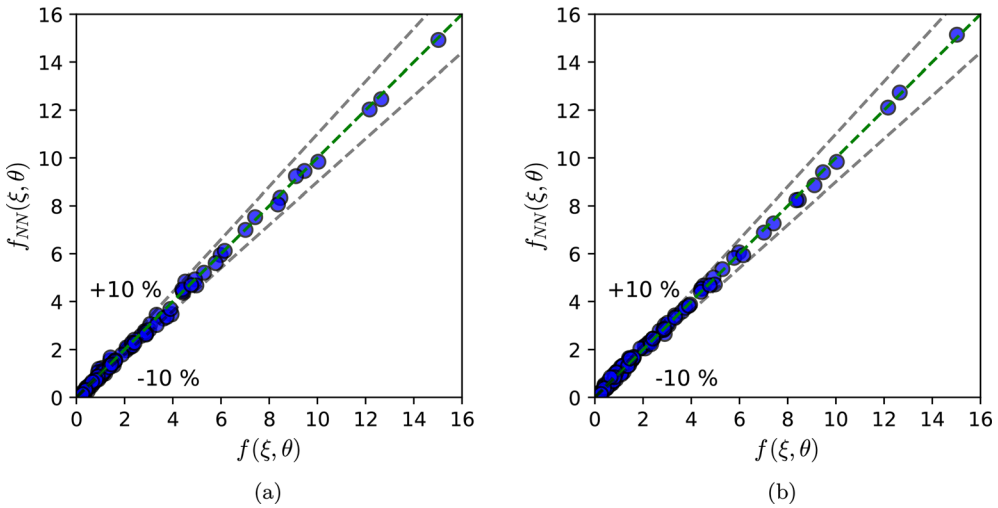

Fig. 9 Parity plots comparing the ANN predicted output $f_{NN}(\boldsymbol{\xi},\boldsymbol{\theta})$ against the true values obtained from the full mechanistic model $f(\boldsymbol{\xi},\boldsymbol{\theta})$. (a) Standard training and (b) gradient-enhanced training.

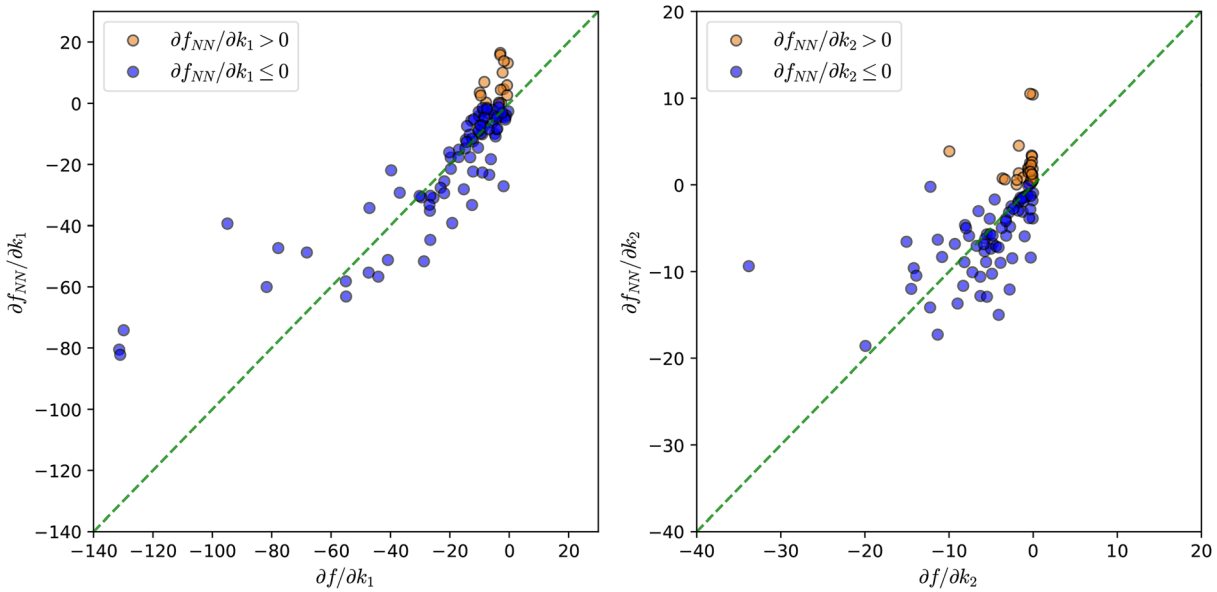

Fig. 10 Parity plots comparing the ANN predicted gradients with respect to k_1 and k_2 under standard training against the true gradients obtained from the mechanistic model.

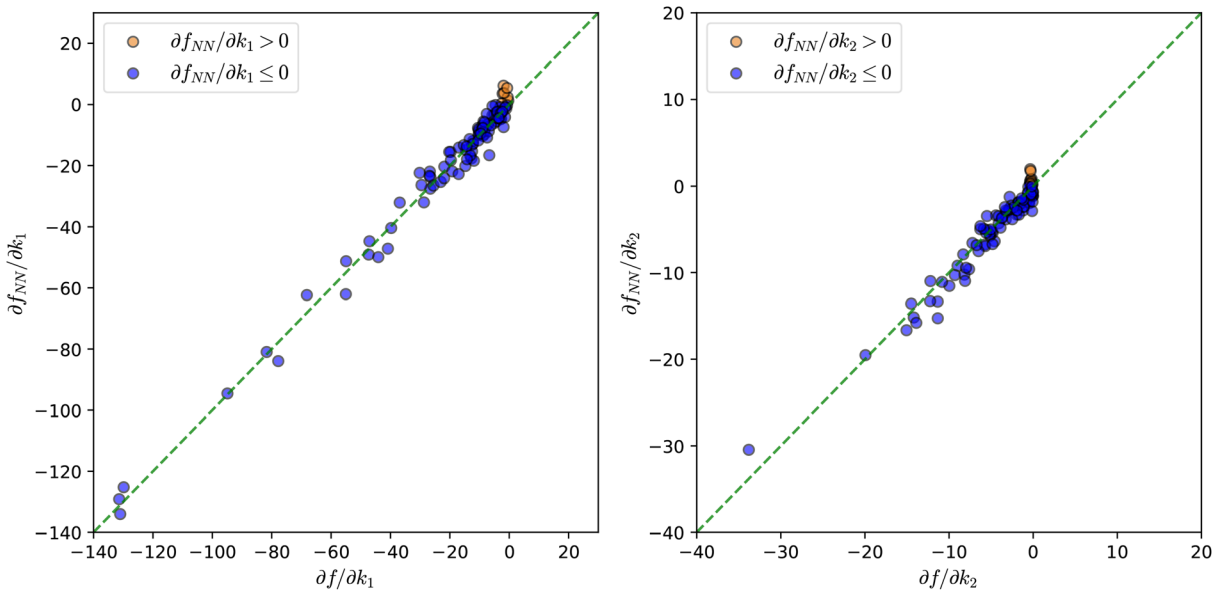

Fig. 11 Parity plots for the ANN predicted gradients with respect to k_1 and k_2 under gradient-enhanced training against the true gradients obtained from the mechanistic model.

under the curve for species B, assuming a constant σ_e across all measurements. The impact of the noise magnitude is tested below.

The campaign began with five preliminary runs selected *via* LHS. Because the initial design may influence the outcome, different LHS designs comprising five runs were evaluated. Subsequently, the sixth and all following experimental conditions were chosen iteratively according to the D-optimality criterion. Optimization of ψ_D was performed using a differential evolution algorithm, as implemented in SciPy.[40]

As discussed in previous sections, a primary motivation for employing surrogate models within an MBDoE framework is the substantial reduction in the computational burden associated with the optimization routines used to design each experiment. Computing the sensitivity matrix $\mathbf{S}$ using the full first-principles model with a central difference scheme, for instance, requires two model evaluations for each parameter $\boldsymbol{\theta}_p$ under every experimental condition $\boldsymbol{\xi}_e$. To assess computational cost, we measured the time required to compute the $[5 \times 2]$ sensitivity matrix for the five preliminary experiments on

a workstation with 16 cores (Intel Xeon w5-3433, 2.00 GHz) and 256 GB RAM. Using finite differences with the first-principles model, which required 20 full model evaluations, the computation time was 970 ± 143 s. In contrast, the gradient-enhanced ANN required only 5 surrogate model evaluations and completed the same task in 0.005 ± 0.003 s, corresponding to a reduction in computational cost by a factor of approximately 200 000. For this comparison, finite differences were preferred over an adjoint approach. Given that only two parameters are estimated, the benefits of employing an adjoint formulation are limited. Additionally, due to the stiffness of the governing equations, the evaluation of sensitivities *via* the adjoint method proved to be more computationally demanding (4571 ± 1278 s).

To quantify the time needed for a full MBDoE cycle, the number of function evaluations was assessed. On average, 640 model evaluations are required for parameter estimation, along with 339 sensitivity computations to determine the D-optimal experimental conditions. Consequently, identifying the optimal experimental design using the first-principles model requires approximately 3–6 days, compared to only 1–5 s when using the surrogate model. This acceleration is critical for an online MBDoE strategy, which relies on a fast iterative calculation of the sensitivity matrix to inform the next optimal experiment in real-time applications.

Effectiveness was assessed using two complementary metrics: (i) the distance between estimated and true parameter values and (ii) the probability that the true parameters lie within the estimated confidence region at the end of the workflow. Results are shown for a total budget of ten experiments, including five preliminary runs, in Fig. 12 for two noise levels (σ_e = 0.2 and 0.4 mol s m^{-3}, corresponding to approximately 7.5% and 15% relative error relative to the mean output).

To account for variability due to the initial design, the procedure was repeated using ten different LHS initializations, and the median performance after each experiment was reported. In addition, the coverage probability was estimated from 100 independent runs performed under the same experimental budget. The resulting coverage probabilities, along with the median volume ratio and correlation error, are summarized in Tables 1 and 2.

For comparison, three surrogate models were evaluated: the standard ANN with the lowest $RMSE_{total}$ ($RMSE_{total} = RMSE_{output} + RMSE_{grad}$), the standard ANN with the lowest $RMSE_{output}$, and the gradient-enhanced ANN with the lowest $RMSE_{total}$. The results in Fig. 12 reveal that the standard-trained ANNs failed to converge to the true parameter values, even in the lower noise scenario. By contrast, the gradient-enhanced ANN converged to the correct parameters. This behaviour arises because the D-optimal designs often involve experimental conditions located at the boundaries of the design space (Fig. S.2–S.4). As demonstrated in the previous section, standard ANNs exhibit reduced accuracy near boundary regions, particularly under limited training data. Consequently, the MBDoE procedure systematically selects experiments in regions where the surrogate is least reliable, creating a feedback loop that reinforces model bias and prevents convergence to the true parameters.

As expected in sequential MBDoE, all surrogates guide the algorithm toward experiments that reduce parameter uncertainty, as evidenced by the steady decrease in the median determinant of the FIM (Fig. S.5). However, reducing uncertainty alone is insufficient; a reliable surrogate must also ensure that the resulting confidence region is centered on the true parameter values. This aspect is captured by the coverage probability.

Under lower noise conditions, the standard ANNs exhibit low coverage probabilities of 4% and 7%. As shown in Fig. 12a, this failure is primarily due to their inability to approach the true parameter values. Interestingly, these models still show

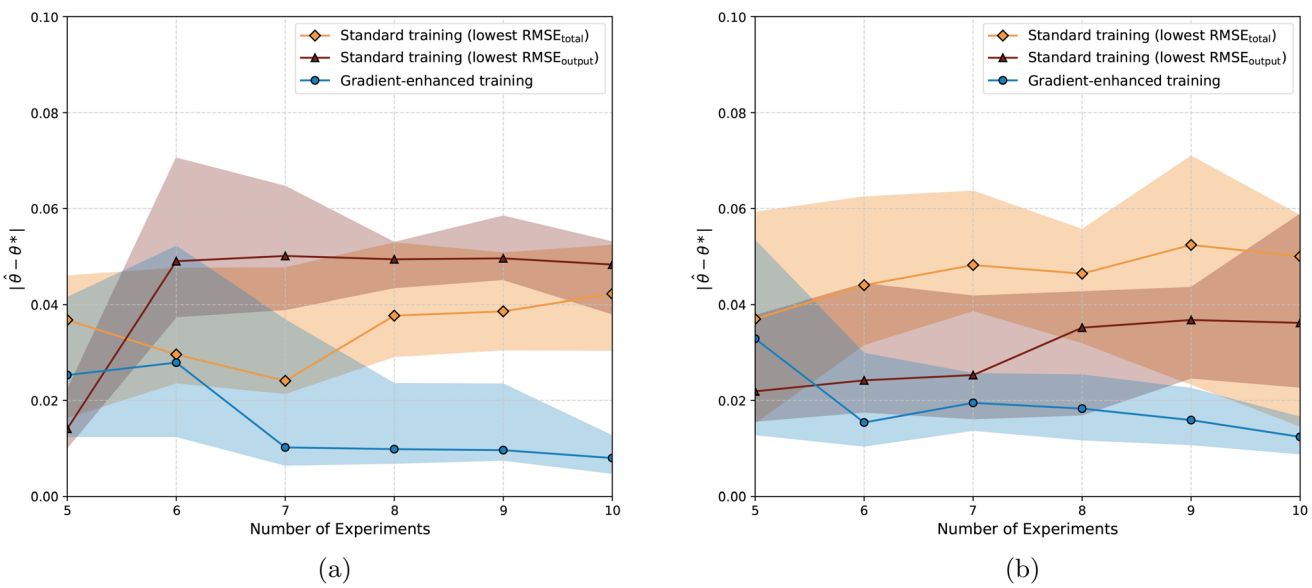

Fig. 12 Median distance between the estimated parameters $\hat{\boldsymbol{\theta}}$ and the true parameter values $\boldsymbol{\theta}^*$ after each experiment. The shaded area shows the interquartile range of the distribution of values obtained for 10 different initial LHS designs. (a) σ_e = 0.2 mol s m^{-3} and (b) σ_e = 0.4 mol s m^{-3}.

Table 1 Confidence region accuracy and coverage probabilities for sequential experimental design under lower noise conditions ($\sigma_e = 0.2$ mol s m^{-3}). The volume ratio and correlation error are reported as the median [interquartile range] across 100 independent tests

ANN	Volume ratio	Correlation error	Coverage probability
Standard (output-optimized)	0.30 [0.28, 0.34]	0.12 [0.10, 0.15]	4%
Standard (total-optimized)	0.26 [0.22, 0.46]	0.09 [0.04, 0.13]	7%
Gradient-enhanced	0.44 [0.39, 0.49]	0.15 [0.12, 0.21]	60%

Table 2 Confidence region accuracy and coverage probabilities for sequential experimental design under higher noise conditions (σ_e = 0.4 mol s m^{-3}). The volume ratio and correlation error are reported as the median [interquartile range] across 100 independent tests

ANN	Volume ratio	Correlation error	Coverage probability
Standard (output-optimized)	0.42 [0.33, 0.55]	0.20 [0.18, 0.23]	46%
Standard (total-optimized)	0.25 [0.21, 0.37]	0.07 [0.05, 0.16]	36%
Gradient-enhanced	0.62 [0.52, 0.76]	0.22 [0.13, 0.29]	68%

reasonable alignment in terms of correlation error (Table 1), indicating that their FIM-based covariance matrices adequately describe the spread of their own parameter estimates. This can be visually assessed in the SI (Fig. S.6–S.8), which illustrates that while the shape and size of the covariance ellipses generally match the parameter distributions, they are centered far from the true values due to the flawed local sensitivity structure they provide to the experimental design algorithm. In contrast, the gradient-enhanced ANN minimizes the distance to the true parameters, yielding an improved coverage probability of 60%. This value remains below the nominal 95% and may be attributed to limitations of FIM-based uncertainty quantification at this experimental budget ($N_e = 10$), as the FIM provides a local approximation of the parameter covariance matrix that may not fully capture nonlinear effects. As discussed in Section 2, these factors can lead to an underestimation of uncertainty, even when the surrogate accurately represents local model sensitivities.

Comparing the higher noise scenarios reveals an important characteristic of the FIM-based uncertainty metrics. Under higher noise, the coverage probabilities for the standard models improve, rising to 46% and 36% (Table 2). This increase, however, does not reflect an improvement in parameter estimation accuracy; the distance trajectories (Fig. 12b) show that the estimates remain far from the true values. Rather, the higher experimental variance inflates the volume of the resulting confidence ellipses. The coverage probability increases primarily because these wider confidence bounds contain the true parameters more frequently, despite the central estimates remaining off-target. The gradient-enhanced ANN scales its coverage to 68% under these wider bounds.

As an example of sequential experimental planning for parameter estimation, Fig. 13 shows the evolution of the estimated parameters and the associated confidence ellipse. The results are displayed after the five preliminary experiments

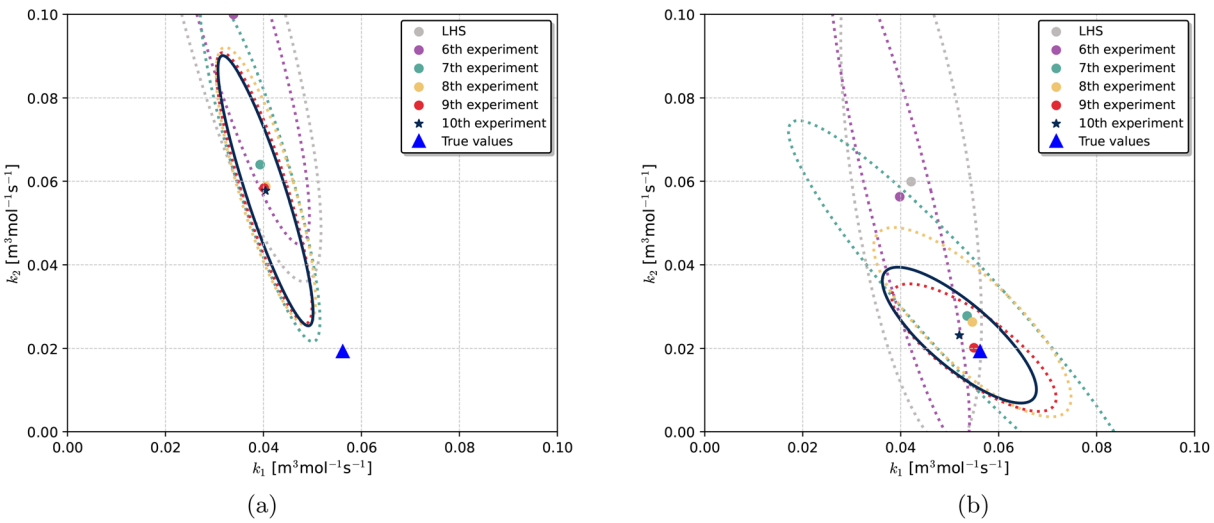

Fig. 13 Confidence regions of the estimated kinetic parameters k_1 and k_2 during an MBDoE campaign. (a) Standard trained ANN with the lowest RMSE$_{total}$. (b) Gradient-enhanced trained ANN.

(LHS), after the sixth optimal experiment, and at subsequent steps, for both the ANN with the lowest $RMSE_{total}$ and the gradient-enhanced ANN. This trajectory is representative of the behaviour discussed above. In both cases, the sequential addition of experiments leads to a reduction of the confidence regions, indicating improved parameter precision. However, the gradient-enhanced ANN gives estimates closer to the true parameter values. This improved accuracy and reliability make the gradient-enhanced ANN a promising candidate to replace first-principles models, enabling its use in real-time, automated experimental platforms for flow chemistry.

6 Conclusions

One of the main barriers to applying MBDoE in automated flow chemistry platforms is the computational burden associated with complex models, particularly when dynamic experiments must be described. This work addressed this challenge by implementing a gradient-enhanced neural network within the MBDoE as a surrogate model to approximate the behaviour and sensitivities of first-principles models. Because automatic differentiation is available in most neural network implementation software, it is possible to compute derivatives with respect to model parameters during training. By incorporating gradient information, the local sensitivity structure is preserved, so the ANN is able to reproduce both model outputs and parameter sensitivities relevant for parameter estimation.

The case study highlighted the effectiveness of the gradient-enhanced neural network through *in silico* synthetic validation. The use of gradient information during training improved predictive accuracy of model derivatives, mitigated the occurrence of physically implausible gradients, and delivered reliable predictions near boundary values that are typically targeted by MBDoE as highly informative regions. The gradient-enhanced approach demonstrated higher robustness in parameter estimation, converging toward true parameter values in a higher percentage of cases than the standard ANN, supported by an increase in coverage probability. Standard ANN surrogates, however, exhibited poor coverage probability in scenarios with low noise and systematic bias in parameter estimates, indicating that accurate reproduction of sensitivities is key to ensuring reliable parameter estimation within MBDoE. In addition, assessments of optimization timing confirmed a substantial reduction in the computational cost required for iterative model evaluations. As a proof-of-concept, these results establish gradient-enhanced neural networks as promising candidates for substituting first-principles models, providing a framework for their future application in self-driven flow chemistry platforms where experimentation, parameter estimation, and optimization can be achieved in a fully automated closed-loop.

Beyond flow chemistry, this framework could be applicable to other domains requiring accelerated parameter estimation in real-time applications. However, the present study relies on synthetic validation and specific noise model assumptions, which may not fully capture experimental variability or potential mismatch between the model and the experimental platform encountered in practice. Validation with real experimental workflows on automated platforms is therefore needed to assess the robustness of the proposed approach. Additionally, limitations arise in systems with a large number of parameters, where surrogate training becomes increasingly challenging and ill-conditioning of the FIM may occur. Future work with larger models could focus on strategies before surrogate training, such as identifying a subset of the most estimable parameters. This reduction in dimensionality would improve numerical stability during FIM inversion and ease the training process by concentrating on the most influential parameters under the given experimental conditions.

Author contributions

Francisco Bolaños-García: conceptualization, methodology, investigation, software, formal analysis, visualization, writing – original draft. Jean-Marc Commenge: conceptualization, formal analysis, methodology, validation, supervision, writing – review & editing. Laurent Falk: validation, supervision, funding acquisition, writing – review & editing.

Conflicts of interest

There are no conflicts to declare.

Data availability

The datasets and source code used for the first-principles model and for the training of standard and gradient-enhanced surrogate models are available in the GitHub repository at https://github.com/FranBol/ANN-MBDoE. The repository also includes the weights and biases for all neural networks. An archived release of the repository has been deposited on Zenodo and is accessible *via* DOI: https://doi.org/10.5281/zenodo.19818341. Supplementary information (SI) regarding the experimental conditions proposed for improving parameter estimation and visual representation of the estimated confidence regions is available. Supplementary information is available. See DOI: https://doi.org/10.1039/d5dd00526d.

Acknowledgements

The authors acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-22-CE51-0025 (project OPTIFLEX).

References

1 F. Delgado-Licona, D. Addington, A. Alsaiari and M. Abolhasani, *Nat. Chem. Eng.*, 2025, **2**, 277–280.
2 G. Lyall-Brookes, A. C. Padgham and A. G. Slater, *Digital Discovery*, 2025, **4**, 2364–2400.
3 A. Slattery, Z. Wen, P. Tenblad, J. Sanjosé-Orduna, D. Pintossi, T. den Hartog and T. Noël, *Science*, 2024, **383**, eadj1817.

4 M. Rodriguez-Zubiri and F.-X. Felpin, *Org. Process Res. Dev.*, 2022, **26**, 1766–1793.
5 G. Buzzi Ferraris, P. Forzatti, G. Emig and H. Hofmann, *Chem. Eng. Sci.*, 1984, **39**, 81–85.
6 G. Buzzi-Ferraris and F. Manenti, *Chem. Eng. Sci.*, 2009, **64**, 1061–1074.
7 M. Geremia, S. Macchietto and F. Bezzo, *Chem. Eng. Sci.*, 2026, **319**, 122347.
8 F. Mathieu, J.-M. Commenge, L. Falk and S. Lomel, *Chem. Eng. Sci.*, 2013, **104**, 829–838.
9 Z. Jiang, J.-F. Portha, J.-M. Commenge and L. Falk, *Chem. Eng. Res. Des.*, 2019, **146**, 290–310.
10 J. P. McMullen and K. F. Jensen, *Org. Process Res. Dev.*, 2011, **15**, 398–407.
11 B. J. Reizman and K. F. Jensen, *Org. Process Res. Dev.*, 2012, **16**, 1770–1782.
12 C. Waldron, A. Pankajakshan, M. Quaglio, E. Cao, F. Galvanin and A. Gavriilidis, *React. Chem. Eng.*, 2020, **5**, 112–123.
13 A. Senthil Vel, K. E. Konan, D. Cortés-Borda and F.-X. Felpin, *Org. Process Res. Dev.*, 2023, **28**(5), 1597–1606.
14 A. Rasch and H. M. Bücker, *ACM Trans. Math. Software*, 2010, **37**(2), 1–37.
15 J. Wang and A. W. Dowling, *AIChE J.*, 2022, **68**, e17813.
16 S. Olofsson, L. Hebing, S. Niedenführ, M. P. Deisenroth and R. Misener, *Comput. Chem. Eng.*, 2019, **125**, 54–70.
17 G. Franceschini and S. Macchietto, *Chem. Eng. Sci.*, 2008, **63**, 4846–4872.
18 A. Friso and F. Galvanin, *Comput. Chem. Eng.*, 2024, **187**, 108724.
19 A. Bhosekar and M. Ierapetritou, *Comput. Chem. Eng.*, 2018, **108**, 250–267.
20 M. Quaglio, L. Roberts, M. S. Bin Jaapar, E. S. Fraga, V. Dua and F. Galvanin, *Comput. Chem. Eng.*, 2020, **135**, 106759.
21 E. Sangoi, M. Quaglio, F. Bezzo and F. Galvanin, *Comput. Chem. Eng.*, 2024, **187**, 108752.
22 G. S. Gusmão, A. P. Retnanto, S. C. da Cunha and A. J. Medford, *Catal. Today*, 2023, **417**, 113701.
23 W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Świrszcz and R. Pascanu, Sobolev Training for Neural Networks, *arXiv*, 2017, preprint, arXiv:1706.04859, DOI: 10.48550/arXiv.1706.04859.
24 S. Srinivas and F. Fleuret, Knowledge Transfer with Jacobian Matching, *arXiv*, 2018, preprint, arXiv:1803.00443, DOI: 10.48550/arXiv.1803.00443.
25 A. Paszke *et al.*, in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Curran Associates Inc., 2019.
26 R. Sellar and S. Batill, in *6th Symposium on Multidisciplinary Analysis and Optimization*, 1996, p. 4019.
27 W. Liu and S. Batill, in *8th Symposium on Multidisciplinary Analysis and Optimization*, 2000, p. 4923.
28 C. Tsay, *Comput. Chem. Eng.*, 2021, **153**, 107419.
29 S. Asprey and S. Macchietto, *Comput. Chem. Eng.*, 2000, **24**, 1261–1267.
30 W. G. Hunter and A. M. Reiner, *Technometrics*, 1965, **7**, 307–323.
31 L. Pronzato and A. Pazman, *Design of experiments in nonlinear models, 2013th ed.; Lecture notes in statistics*, Springer, New York, NY, 2013.
32 Y. Bard, *Nonlinear parameter estimation*, Academic press New York, 1974, vol. 1209.
33 É. Walter and L. Pronzato, *Identification de modèles paramétriques à partir de données expérimentales*, Masson, 1994.
34 D. M. Himmelblau, *Korean J. Chem. Eng.*, 2000, **17**, 373–392.
35 S. Ulaganathan, I. Couckuyt, T. Dhaene, J. Degroote and E. Laermans, *Eng. Comput.*, 2015, **32**, 15–34.
36 F. Sapienza, J. Bolibar, F. Schäfer, B. Groenke, A. Pal, V. Boussange, P. Heimbach, G. Hooker, F. Pérez, P.-O. Persson and C. Rackauckas, Differentiable Programming for Differential Equations: A Review, *arXiv*, 2024, preprint, arXiv:2406.09699, DOI: 10.48550/arXiv.2406.09699.
37 R. T. Q. Chen, Y. Rubanova, J. Bettencourt and D. Duvenaud, Neural Ordinary Differential Equations, *arXiv*, 2019, preprint, arXiv:1806.07366, DOI: 10.48550/arXiv.1806.07366.
38 O. Levenspiel, *Chemical Reaction Engineering*, Wiley, 1999.
39 F. Wagner, P. Sagmeister, C. E. Jusner, T. G. Tampone, V. Manee, F. G. Buono, J. D. Williams and C. O. Kappe, *Advanced Science*, 2024, **11**, 2308034.
40 P. Virtanen, *et al.*, *Nat. Methods*, 2020, **17**, 261–272.