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

Data-driven Modeling of a Continuous Direct Compression Tableting Process using SINDy

Pau Lapiedra Carrasquer[a,b], Satyajeet S. Bhonsale[a], Carlos André Muñoz López[b], Kristof Dockx[b], Jan F.M. Van Impe[a]*

[a] BioTeC+, KU Leuven, Gebroeders De Smetstraat 1, Gent 9000, Belgium
[b] Janssen Pharmaceutica NV, a Johnson & Johnson company, Turnhoutseweg 30, 2340 Beerse, Belgium
* Corresponding Author: jan.vanimpe@kuleuven.be.

ABSTRACT

Understanding the complex dynamics of continuous processes in pharmaceutical manufacturing is essential to ensure product quality across the production line. This paper presents a data-driven modeling approach using Sparse Identification of Nonlinear Dynamics with Control (SINDYc) to capture the dynamics of a continuous direct compression (CDC) tableting line. By incorporating delayed control inputs into the candidate function library, the model effectively captures deviations from steady state in response to dynamic changes. The proposed model was developed by finding a balance between accuracy and sparsity, with focus on the ability to generalize to a wide range of operating conditions.

Keywords: Big Data, Dynamic Modelling, Industry 4.0, Machine Learning, Modelling, SINDy

INTRODUCTION

Continuous manufacturing has emerged as a crucial innovation in pharmaceutical tableting production, bringing substantial improvements in process efficiency, scalability, and product quality assurance. However, the complex dynamics of continuous processes pose significant challenges for modeling, especially in ensuring consistent quality across the production line. In continuous manufacturing lines, residence time distribution (RTD) is a key concept describing how material flows through the system. In practice, RTD is usually modeled using a sequence of ideal reactor units (e.g., combinations of continuous stirred-tank reactors and plug-flow reactors). These models often rely on idealized assumptions that may not capture the complexities of real-world processes. This can lead to discrepancies between model predictions and actual system behavior [1]. Additionally, these models may lack the flexibility to generalize across different setups or scenarios, making it challenging to predict performance under untested conditions [2].

One promising approach to modeling nonlinear and time-varying systems is through data-driven methods. The Sparse Identification of Nonlinear Dynamics (SINDy) framework is a powerful tool for identifying system dynamics from experimental data without assuming a predefined structure. The SINDy algorithm has the ability to derive parsimonious models, with fewer terms.

In this study, SINDy was applied to model a Continuous Direct Compression (CDC) tableting line, using simulated data generated with the gPROMS flowsheet modeling environment. This approach provides the flexibility to simulate a wide variety of experimental conditions, producing the data needed to train the SINDy model. The control input for this model is the mass flow rate of the active pharmaceutical ingredient (API) from the feeder, while blend uniformity and content uniformity are defined as state variables. A key challenge in this system is the presence of inherent time delays between changes in the mass flow rate and their observed effects on blend uniformity. To effectively capture these delayed dynamics, the SINDy with control (SINDYc) algorithm was used, allowing for the explicit inclusion of delayed control inputs into the system's dynamic equations.

THEORY AND METHODS

SINDy method

The SINDy algorithm described in Brunton et al. [3] uses sparse regression to identify a parsimonious set of terms governing the system's dynamics. Consider a nonlinear dynamical system in the form

$$\frac{dx}{dt} = f(x) \tag{1}$$

where $x \in \mathbb{R}^n$ are the state variables over time and the function f represents the dynamics. SINDy uses sparse regression to identify the few active terms out of a library of candidate nonlinear functions $\Theta(X)$. From the time derivative of the state variables $\dot{x}$(t), SINDY finds a sparse matrix Ξ such that:

$$\dot{X} = \Theta(X)\Xi \tag{2}$$

where Ξ contains the coefficients for each candidate function. In this case, the derivatives $\dot{x}$(t), are numerically calculated from the state data.

To incorporate the effects of control inputs in the CDC tableting line, an extension of SINDy known as SINDy with control (SINDYc) is used [4]. SINDYc allows the inclusion of external inputs, in this case the mass flow rate of API in the feeder, directly into the dynamic equations, capturing the influence of these inputs on the system's response variables. The expression of Eq. (1) and (2) can be rewritten as:

$$\frac{dx}{dt} = f(x,u) \tag{3}$$

$$\dot{X} = \Theta(X,U)\Xi \tag{4}$$

where U(t) represents the control inputs and Θ(X,U) is a library matrix containing candidate functions of both the state variables X and control inputs U. Commonly used candidate functions include constant, polynomial, trigonometric and exponential terms or a combination of them:

$$\Theta\,(X,U) = \\ [1 \;\; X \;\; U \;\; X^2 \;\; U^2 \;\; XU \;\; sin(X) \;\; sin(U) \;\; e^X \;\; e^U \;\cdots] \tag{5}$$

In many systems, the dynamics can be accurately represented using only a few active terms from a larger set of candidate functions. To identify these terms, sparse regression is used to find the coefficients in the matrix Ξ, minimizing the following objective function:

$$\Xi = \underset{\Xi}{argmin}\frac{1}{2}\left\|\dot{X} - \Theta(X,U)\Xi\right\|_2^2 + \lambda\|\Xi\|_1 \tag{6}$$

where λ is a sparsity-promoting hyperparameter that determines the degree of sparsity in the model. This term penalizes non-zero elements in Ξ, encouraging the model to retain only the most significant terms. During the optimization process, any coefficients in Ξ that fall below a threshold set by λ are set to zero, effectively removing less important terms from the model. This optimization problem can be solved by sequential threshold least squares (STLSQ), where λ serves as the threshold [5].

Flowsheet model of a Continuous Direct Compression line

In this work, we consider a Continuous Direct Compression (CDC) process, where the API is mixed with the excipients and lubricant to finally move into the tablet press. A flow diagram of the CDC process is shown in Figure 1. CDC was chosen because it is the simplest tableting configuration, resulting in more resources being available in the literature for building the model.

The data used in this work was generated with a flowsheet model of a Continuous Direct Compression (CDC) line implemented in gPROMS FormulatedProducts 2023.1.1. Its modeling environment allows us to connect the models of the different unit operations needed for the flowsheet model [6].

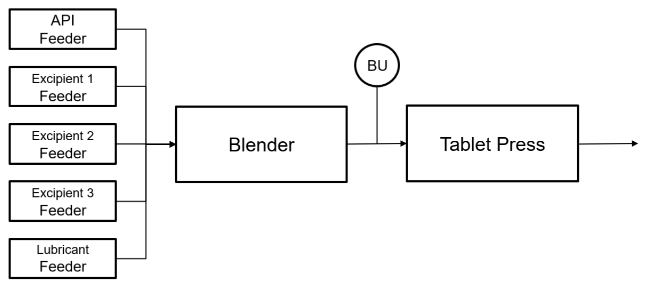

Figure 1. Process flow diagram of the Continuous Direct Compression line.

The data generated for this study consists of a series of step changes on the API mass flow rate, which in this work represents the control input u,and its response at the end of the blender in terms of blend uniformity (BU), that serves as our state variable x, as it is commonly done when characterizing the RTD. To guarantee unit consistency between the control input and the state variable, the mass flow rate and the blend uniformity are expressed as % from the target value. Additionally, unusual control input signals were generated to test the generalization ability of the model identified with the SINDy approach.

SINDy with control for a CDC line

One of the key steps in implementing SINDy is selecting a candidate function library that best captures the dynamics of the system. A common approach is to start with a set of polynomial functions and then increase the library's complexity by incorporating additional terms, as shown in Eq. (4). Typically, SINDYc includes the control input u as a separate variable in the library [7]. However, in this study, we opted to represent the control input as $u - x$. This formulation is particularly suited to continuous direct compression (CDC) systems, which generally operate in a steady state for extended periods before experiencing a dynamic change, such as a step change or pulse input in the API mass flow. By defining the control input as $u - x$, it effectively captures the relative difference or deviation between the control signal and the state. This deviation term can better capture how the control input drives the system response away from the current state.

An effective approach to selecting the functions of the candidate library is using the knowledge of the process. In a continuous line, there is usually a time delay in the response after a change occurs upstream. To account for this time delay, we propose to include a possible discrete delay τ on the control input terms as part of the candidate function library as follows:

$$\Theta\,(x,u) = [1 \quad x \quad u \quad u(t-\tau_1) \quad u(t-\tau_2) \quad u(t-\tau_3) \;\ldots\; u(t-\tau_n)]. \qquad (7)$$

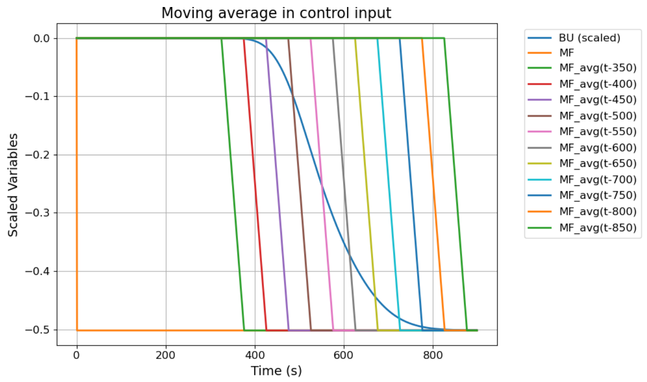

Figure 2. Example of using the moving average on the delayed control input functions used in the candidate library

The magnitude and number of delays are selected based on the approximate duration of the step change response of the training data. Additionally, to increase the amount of information captured by including the delayed versions of u, we opted to use a moving average $\hat{u}(t-\tau_i)$ over a time window around each delayed signal $u(t-\tau_i)$. For example, if the delay of the response is around 400 seconds and that response stabilizes at around 800 seconds, a list of delayed versions of u from 350 until 850 seconds every 50 seconds is chosen. For each delayed term, we calculate a moving average over a ±25 s window rather than using a single point in time. This approach helps capture the short-term variations that would otherwise be lost when discretizing the input to $u(t-\tau_i)$. In other words, by including the surrounding data points, we account for small fluctuations around the delayed term, instead of relying only on a single value. This can improve both the smoothness of the input signal and the model's ability to learn the underlying dynamics. This example is represented in Figure 2, where the control input u is defined as mass flow rate (MF) and the response or state variable is the blend uniformity (BU).

RESULTS

The SINDYc with delay approach was trained on simulated data representing step changes of various sizes around the target value. Pre-processing included scaling the data, with a target value of 0 for both the API mass flow (control input) and the API concentration (response), while 1 and -1 were set as the extreme values for the training data. The SINDy model was implemented using the PySINDY package in Python [8, 9]. The derivatives of the state variable were approximated using second-order finite differences. After constructing the model, the resulting ordinary differential equation (ODE) was solved numerically with the fourth order Runge-Kutta method to generate predictions of the state variable. Model performance was evaluated using root mean-square error (RMSE), calculated as follows:

$$RMSE = \sqrt{\tfrac{1}{N}\textstyle\sum_{i=1}^{N}(x_i - \hat{x}_i)^2} \qquad (8)$$

where x_i represents the actual value of the response

Table 1: Root mean square error (RMSE) of SINDy models under various configurations for training and two test datasets. "AW" = Averaging Window (±s), "DI" = Delay Interval (s). The values in bold indicate the selected model configuration.

Regularization	Configuration	RMSE		
λ	(±AW, DI)	Training	Test 1	Test 2
$\lambda = 0.001$	(±25 s, 25 s)	0.7278	0.8045	0.7372
	(±25 s, 50 s)	0.2585	0.2888	0.2600
	(±50 s, 25 s)	0.4931	0.5460	0.4801
	(±50 s, 50 s)	**0.1709**	**0.1632**	**0.1765**
$\lambda = 0.0001$	(±25 s, 25 s)	0.1162	0.0835	0.3958
	(±25 s, 50 s)	0.2158	0.2109	0.3263
	(±50 s, 25 s)	0.1094	0.0872	0.1673
	(±50 s, 50 s)	0.1256	0.0845	0.1362

(blend uniformity) and $\hat{x}_i$ stands for the predicted value from the SINDYc model.

To determine the level of sparsity from Eq.(6) various values of λ were tested. Lower λ generally improves RMSE because the regression is less penalized for including more terms; however, it can result in a model with many active terms, undermining the sparsity objective of SINDy. On the other hand, higher λ forces the solution to discard more terms, reducing complexity but sometimes increasing prediction error. At λ=0.001, the model achieved the highest level of sparsity before all coefficients were reduced to zero. While decreasing λ to 0.0001 improved model performance, it resulted in most coefficients being selected, which could reduce robustness in the presence of noisy data.

In this SINDy approach, two additional hyperparameters related to the delayed control input need to be tuned: the delay interval and the averaging window. Based on the system's dynamics, where the response to a step change begins around 400 s after the control input and stabilizes around 800 s, a delay range from 350 to 850 s was chosen. Within the estimated response range, we tested two delay interval increments, every 25 s and every 50 s, to control the model complexity. Using 25 s intervals resulted in more terms in the model, while 50 s intervals produced fewer. In either case, each delayed control input was further smoothed using a moving average of either ±25 s or ±50 s to capture the input's gradual effect. These window lengths were chosen so that ±25 s partially covers the 25 s delay interval around the nominal delay, while ±50 s fully spans the 50 s delay increment. In this way, the moving average captures the gradual or integrated effect of the control input in the immediate surroundings of each delay, ensuring that small timescale fluctuations are included without unnecessarily increasing the complexity of the model.

Table 1 summarizes the root mean square error (RMSE) obtained for each configuration of the SINDYc model on three data sets: the original training data, a first test set with a conventional control input profile (test 1), and a second test set with a more challenging input profile (test 2). In training and test 1 data sets, the control input profiles mimic typical conditions during process development in a continuous direct compression (CDC) line, with step changes of moderate magnitude and duration. By contrast, test 2 data set explores significantly different operating conditions, including a different initial condition, sinusoidal variation, and sudden spikes, to test the model's robustness well beyond the scenarios used for training.

Each configuration in Table 1 varies two key factors: the regularization parameter λ, which controls the sparsity of the model, and the delay/averaging combination used to create candidate functions (e.g., 25 s vs. 50 s increments, ±25 s vs. ±50 s moving average windows.

As shown in Table 1, the highlighted configuration achieves a good balance of accuracy and parsimony across all three data sets. Although the configuration with λ=0.0001 shows slightly lower RMSE values in some cases, it selects nearly all candidate terms, resulting in a much more complex model that could lead to potential overfitting. In contrast, the chosen configuration with λ=0.001 retains comparable predictive accuracy on both test 1 and test 2, while maintaining a significantly smaller number of terms. This aligns with the SINDy principle of favouring parsimonious models that capture essential dynamics without overfitting.

To demonstrate how the chosen regularization parameter λ affects the model sparsity, we present in Eq.(8) the final form of the obtained model with λ=0.001, and in Eq.(9) the corresponding model when λ=0.0001. Both models use the same delay/averaging configuration (i.e., ±50 s averaging windows and 50 s delay increments), but differ in the strength of their sparsity.

$$\dot{x} = 0.008(\hat{u}(t-450)-x) + 0.004(\hat{u}(t-500)-x) + 0.005(\hat{u}(t-550)-x) + 0.001(\hat{u}(t-650)-x) \quad (9)$$

$$\dot{x} = 0.0008(\hat{u}(t-400)-x) + 0.003(\hat{u}(t-450)-x) + 0.002(\hat{u}(t-500)-x) - 0.002(\hat{u}(t-550)-x) - 0.0006(\hat{u}(t-600)-x) - 0.001(\hat{u}(t-650)-x) - 0.0007(\hat{u}(t-700)-x) - 0.0008(\hat{u}(t-750)-x) - 0.0001(\hat{u}(t-850)-x) \quad (10)$$

In Eq. (8), only four terms are present, which manage to capture the main time delays while preserving interpretability. But, when λ is reduced to 0.0001 in Eq. (9), the regularization penalty on additional terms is much weaker, so, more delayed terms remain active, offering a marginally lower RMSE (as shown in Table 1) but at the expense of simplicity.

In addition to selecting λ=0.001, we also explored two key parameters for constructing our candidate functions: the averaging window (±25 s vs. ±50 s) and the delay interval (25 s vs. 50 s). From Table 1 it can be seen that at λ=0.001, widening the averaging window from ±25 s to ±50 s results in a slight improvement in RMSE, but the more influential factor is the choice of delay interval. When the model is forced to choose fewer delayed terms (i.e., using a 50 s delay interval rather than 25 s), it performs slightly better. This indicates that at higher regularization, including too many candidate terms (as occurs with the finer 25 s interval) might dilute the model's ability to identify a compact set of truly relevant dynamics. In other words, a coarser delay interval can sometimes help the algorithm focus on the most dominant time lags in the process, improving predictive performance and interpretability.

When λ=0.0001, the effect of averaging window and delay interval becomes less straightforward. Because the

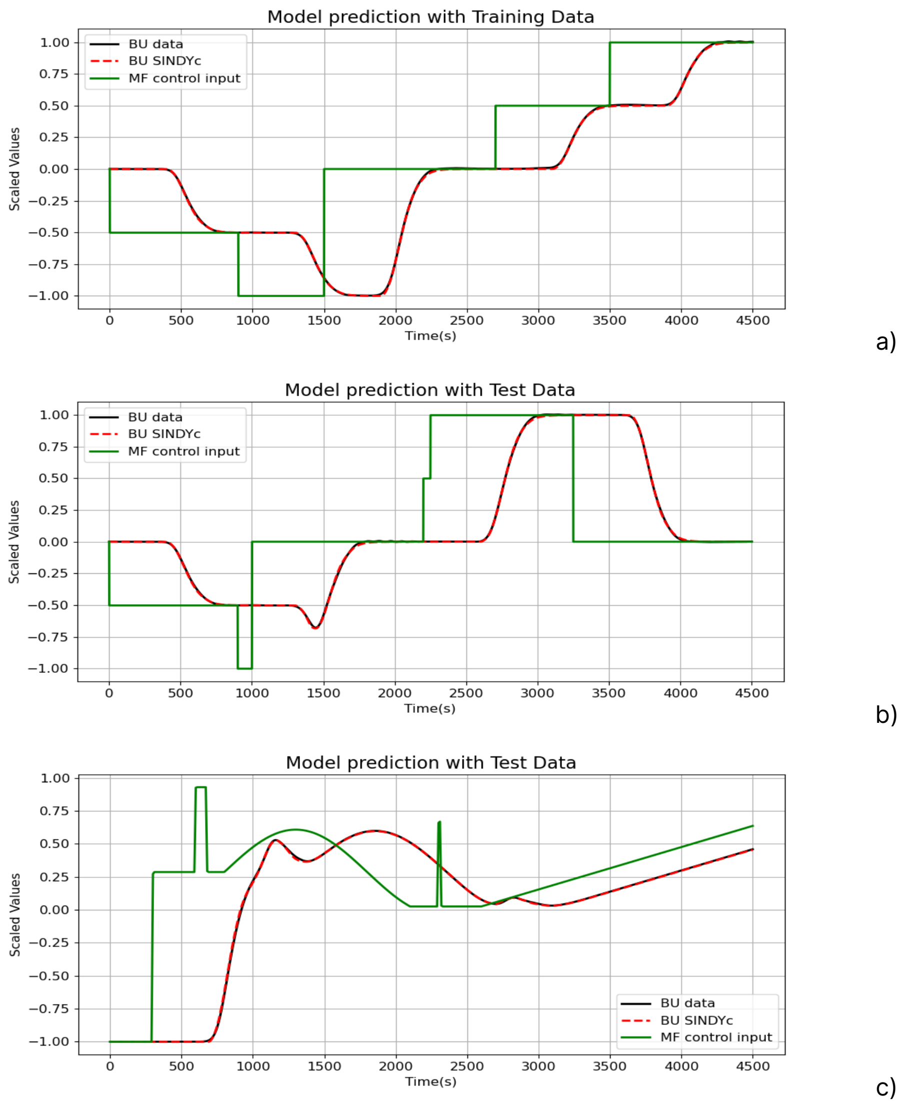

Figure 3. Training (a) and test 1 (b) , and test 2 (c) data sets. The in silico BU process data (x) is compared with the SINDYc model prediction given a control input signal u (mass flow rate or MF)

penalization of extra terms is much lower, the model can retain many more active terms in the final solution—potentially benefiting from finer resolution of time delays (25 s intervals). The trade-off is that these extra terms may capture minor effects rather than main dynamics. As a result, some configurations with 25 s intervals and smaller averaging windows at λ=0.0001 can show lower RMSE on the training set, but at the cost of reduced sparsity and potentially greater risk of overfitting. Future work may employ cross-validation to further refine these hyperparameter selections.

Figure 3a-3c illustrate representative time-series predictions for each data set at the chosen model configuration (i.e., λ=0.001, ±50 s averaging windows and 50 s delay increments). For the training data (Figure 3a), the model prediction (dotted red line) closely tracks the observed blend uniformity or BU (black line) across successive step changes in the mass flow rate (green line). In test 1 (Figure 3b), the same model continues to perform well, with only minor deviations from the true signal in the transitions. Test 2 (Figure 3c) tries far more drastic input variations. While the prediction error increases slightly, the model still manages to capture major trends, indicating that it generalizes reasonably well to unseen input profiles.

Overall, these results indicate that a careful choice of λ, along with a suitable delay-averaging configuration, can result in a compact, interpretable SINDy model that remains robust to diverse control input scenarios—ranging from typical operating conditions (test 1) to more extreme or unconventional inputs (test 2).

CONCLUSIONS

In this study, a SINDy-based model with delayed control input terms was developed and tested to capture the dynamics of a continuous direct compression (CDC) line. By incorporating delayed versions of the control input in the candidate function library and systematically optimizing the delay intervals and averaging windows, the resulting model effectively balances accuracy and sparsity. The selection of an appropriate regularization parameter λ was crucial to achieving this balance, as higher values promoted interpretability by enforcing sparsity, while lower values, although with lower RMSE, led to overfitting with an excessive number of active terms.

The analysis of different configurations showed that using a coarser delay interval (50 s) resulted in improved model performance compared to a finer interval (25 s) when a higher regularization parameter was applied. This suggests that fewer, well-chosen delay terms allow the model to focus on capturing the dominant dynamics of the system, potentially reducing the risk of overfitting to noise or minor effects. The choice of the averaging window also played a role in enhancing the model's robustness, with a ±50 s window providing a more smoothed representation of the input that improved predictive accuracy across various operating conditions. These findings highlight the importance of tuning hyperparameters to achieve an optimal trade-off between model complexity and predictive capability.

The chosen model configuration showed strong generalization performance when applied to test data, including both typical operating profiles encountered in CDC lines and more challenging scenarios such as sinusoidal variations, constant increasing slopes, and sudden spikes. Overall, this approach demonstrates the feasibility of using SINDy for data-driven modelling in CDC processes and provides a foundation for future work to address more complex dynamic responses and noisy data.

ACKNOWLEDGEMENTS

The authors want to acknowledge the financial support from the Flanders Innovation & Entrepreneurship (VLAIO) and Janssen Pharmaceutica NV in Belgium (Grant No. HBC.2022.0177).

REFERENCES

1. G Tian, A Koolivand, Z Gu, M Orella, R Shaw, and T F O'Connor, Development of an RTD-based flowsheet modeling framework for the assessment of in-process control strategies. *AAPS PharmSciTech* 22: 10.1208/s12249-020-01913-8
2. P Bhalode, H Tian, S Gupta, S M Razavi, A Roman-Ospino, S Talebian, R Singh, J V Scicolone, F J Muzzio, and M Ierapetritou, Using residence time distribution in pharmaceutical solid dose manufacturing – a critical review. *International Journal of Pharmaceutics* 610:121248 10.1016/J.IJPHARM.2021.121248
3. S L Brunton, J L Proctor, and J N Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences of the United States of America* 113:3932–3937 10.1073/pnas.1517384113
4. E Kaiser, J N Kutz, and S L Brunton, Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474: 10.1098/rspa.2018.0335
5. U Fasel, E Kaiser, J N Kutz, B W Brunton, and S L Brunton, SINDy with control: a tutorial.
6. N Metta, M Ghijs, E Schäfer, A Kumar, P Cappuyns, I Van Assche, R Singh, R Ramachandran, T De Beer, M Ierapetritou, and I Nopens, Dynamic flowsheet model development and sensitivity analysis of a continuous pharmaceutical tablet manufacturing process using the wet granulation route. *Processes* 7:1–35 10.3390/pr7040234
7. F Abdullah and P D Christofides, Data-based modeling and control of nonlinear process systems using sparse identification: an overview of recent results. *Computers and Chemical Engineering* 174:108247 10.1016/j.compchemeng.2023.108247
8. A A Kaptanoglu, B M de Silva, U Fasel, K Kaheman, A J Goldschmidt, J Callaham, C B Delahunt, Z G Nicolaou, K Champion, J-C Loiseau, J N Kutz, and S L Brunton, PySINDy: a comprehensive python package for robust sparse system identification. *Journal of Open Source Software* 7:3994 10.21105/joss.03994
9. B M de Silva, K Champion, M Quade, J-C Loiseau, J N Kutz, and S L Brunton, PySINDy: a python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software* 5:2104 10.21105/joss.02104

© 2025 by the authors. Licensed to PSEcommunity.org and PSE Press. This is an open access article under the creative commons CC-BY-SA licensing terms. Credit must be given to creator and adaptations must be shared under the same terms. See https://creativecommons.org/licenses/by-sa/4.0/