StrikePy: NONLINEAR OBSERVABILITY ANALYSIS OF INPUTS, STATES, AND PARAMETERS IN PYTHON

David Rey Rostro[1], Alejandro F. Villaverde[1,2]
[1] Universidade de Vigo, Department of Systems Engineering & Control, 36310 Vigo, Galicia, Spain
[2] CITMAga, 15782 Santiago de Compostela, Galicia, Spain
e-mails: davidreyrostro@gmail.com, afvillaverde@uvigo.gal

Abstract

Dynamic models typically have unknown parameters that must be estimated from data, and they may also have unknown inputs (disturbances). The concept of observability, which describes the possibility of inferring the internal state of a system by measuring its output, can be extended to account also for the possibility of inferring its unknown parameters and inputs. Such an extension leads to a property that may be called FISPO (Full Input, State, and Parameter Observability). Its analysis is particularly relevant in systems biology, since models from this area often have a large number of unknown parameters, as well as state variables that cannot be measured due to experimental limitations. It is usually challenging to assess the FISPO of nonlinear models, which has motivated the development of specialized software such as the MATLAB toolbox STRIKE-GOLDD. However, despite the increasing popularity of Python among the biological modelling community, there was a lack of computational tools for FISPO analysis in this language. To fill this gap, we have developed an open source software toolbox, StrikePy, which implements the core functionalities in STRIKE-GOLDD.

Keywords: nonlinear systems, observability, identifiability, dynamic modelling, biosystems.

1 INTRODUCTION

The concept of observability describes the theoretical possibility of inferring the internal state of a system from observations of its outputs [5]. Observability is a classical control-theoretic property that is commonly assessed in control engineering applications. A concept closely related to observability is structural identifiability (SI), which refers to the possibility of inferring the unknown parameters in a model. Initially, the study of SI was motivated by biological modelling applications [1], and since then it has been studied with particular interest in the research community working in dynamic modelling and systems biology [3, 15].

Observability and SI can be studied jointly [12] by considering parameters as constant state variables [11]. Note that this makes a linear model nonlinear; a number of classical mathematical results are available for the analysis of nonlinear observability [4], which have been recently extended to consider problems with unknown inputs [6]. While a number of methodological approaches exist for this purpose, they are mathematically involved and their application is far from trivial. The challenging nature of these analyses has motivated the development of a number of specialized software tools [2, 7, 9, 12, 15].

One such tool is the Matlab toolbox STRIKE-GOLDD, which was originally presented in [13] and extended to analyse "Full Input-State-Parameter Observability" (FISPO) in [14]. STRIKE-GOLDD casts the structural local identifiability problem as a nonlinear observability problem, and it adopts a differential geometry approach – also known as geometric control – for its analysis. Essentially, the observability of all model variables (states, parameters, and unknown inputs) is determined by calculating the rank of a generalized observability-identifiability matrix, which is built using Lie derivatives. When the matrix does not have full rank, the model contains unobservable variables. If these variables are parameters, they are called (structurally) unidentifiable.

Here we present StrikePy v1.0, a Python implementation of the FISPO algorithm from the MATLAB toolbox STRIKE-GOLDD. It analyses the structural local identifiability and observability of nonlinear dynamic models, which may have multiple time-varying and possibly unknown inputs. StrikePy implements the core functionalities of the FISPO algorithm included in STRIKE-GOLDD. It determines the subset of identifiable parameters, observable states, and observable (also called reconstructible) inputs. This approach is directly applicable to many models of small and medium size; larger systems can be analysed using additional features of the method. One of them is to build observability-identifiability matrices with a reduced number of Lie derivatives. In some

 https://doi.org/10.17979/spudc.9788497498418.0430

cases, these additional procedures allow to determine the identifiability of every parameter in the model (complete case analysis); when such result cannot be achieved, at least partial results – i.e. identifiability of a subset of parameters – can be obtained.

The remainder of this paper is structured as follows. In Section 2 we summarize the main theoretical results about nonlinear observability analysis. In Section 3 we describe how StrikePy implements said analyses. In Section 4 we assess its performance by applying it to a set of problems and providing the computational results. Finally, in Section 5 we discuss the most relevant aspects of this work.

2 OBSERVABILITY OF NONLINEAR MODELS

2.1 MODELS AND CONCEPTS

In this work we analyse deterministic models of ordinary differential equations,

$$\mathcal{M} = \begin{cases} \dot{x}(t) = f\left(x(t), \theta, u(t), w(t)\right) & (1) \\ y(t) = g\left(x(t), \theta, u(t), w(t)\right) & (2) \\ x^0 = x(t_0, \theta) & (3) \end{cases}$$

where f and g are nonlinear functions whose variables are states $x(t) \in \mathbb{R}^{n_x}$, known inputs $u(t) \in \mathbb{R}^{n_u}$, unknown constants (i.e. parameters) $\theta \in \mathbb{R}^{n_\theta}$, and unknown inputs (i.e. disturbances) $w(t) \in \mathbb{R}^{n_w}$. The output vector $y(t) \in \mathbb{R}^{n_y}$ contains the measurable functions of model variables. Both types of inputs, $u(t)$ and $w(t)$, are infinitely differentiable functions. In the following, we may omit the dependency of a variable on time (t) whenever it is convenient to simplify the notation, as long as it is obvious from the context.

Definition 1 (Structural local identifiability)**.** *A parameter θ_i of model $\mathcal{M}$ is structurally locally identifiable (SLI) if for almost any vector of values $\theta^* \in \mathbb{R}^{n_\theta}$ there is a neighbourhood $\mathcal{N}(\theta^*)$ where the following condition is fulfilled [3]:*

$$\hat{\theta} \in \mathcal{N}(\theta^*) \text{ and } y(t, \hat{\theta}) = y(t, \theta^*) \Rightarrow \hat{\theta}_i = \theta_i^* \quad (4)$$

If (4) does not hold, θ_i is *structurally unidentifiable* (SU). A model is said to be SLI if all its parameters are SLI. If it has one or more SU parameters, it is called SU.

Parameter identifiability can be considered a particular case of observability, since parameters can be seen as constant states. Accordingly, in [14] the acronym FISPO – Full Input-State-Parameter Observability (or Observable, if used as an adjective) – was introduced to describe this property:

Definition 2 (Full Input-State-Parameter Observability, FISPO)**.** *For a model $\mathcal{M}$ described by equations (1), (2), (3), let us denote the vector of its states, parameters, and unknown inputs as $z(t) = \big(x(t), \ \theta, \ w(t)\big)$. We say that $\mathcal{M}$ has the FISPO property (or, equivalently, that it is FISPO) if the value of every variable z_i included in z can be determined from knowledge of $y(t)$ and $u(t)$, with $t \in [t_0, t_f]$. This means that $\mathcal{M}$ is FISPO if, for every $z(t_0)$ and for almost any vector $z^*(t_0)$, there is a neighbourhood $\mathcal{N}\left(z^*\left(t_0\right)\right)$ in which the following condition holds for all $\hat{z}(t_0) \in \mathcal{N}\left(z^*\left(t_0\right)\right)$:*

$$y\left(t, \hat{z}(t_0)\right) = y\left(t, z^*\left(t_0\right)\right) \Rightarrow \hat{z}_i\left(t_0\right) = z_i^*\left(t_0\right),$$

where $1 \leq i \leq n_x + n_\theta + n_w$.

The FISPO property encapsulates parameter structural identifiability, state observability, and input observability. Conceptually, a state variable at time τ, $x_i(\tau)$, is (structurally locally) *observable* if it can be distinguished from all other neighbouring states by observing the output $y(t)$ and input $u(t)$ in a finite time interval $t_0 \leq \tau \leq t \leq t_f$. Otherwise, $x_i(\tau)$ is *unobservable*. A model is said to be observable if all its states are observable. Similarly, model $\mathcal{M}$ is called *input observable* if it is possible to reconstruct its unknown input $w(t)$ – which is also called observable in that case – from observations of its output.

We remark that the properties described above are local, i.e. they refer to the possibility of distinguishing the true value of a variable – be it a parameter, input, or state – from other values in its neighbourhood, but they do not guarantee their uniqueness outside this neighbourhood. Thus, even for a locally identifiable parameter there may exist several indistinguishable solutions in the parameter space. However, for many practical purposes it is sufficient to have a locally identifiable and observable model, because locally identifiable parameters are often globally identifiable, and even when they are not, there is sometimes only one solution within the admissible range of values.

2.2 ANALYSING NONLINEAR OBSERVABILITY

To analyse the FISPO property we consider the unknown parameters of the model, θ, as additional state variables; since they are constant, their time derivatives are equal to zero, $\dot{\theta}(t) = 0$. In this way we obtain an augmented state vector $\widetilde{x} = (x^T, \theta^T)^T$ [11].

We proceed in a similar way with the unknown inputs, $w(t)$, with the difference that their derivatives may be nonzero. Thus we need to augment

https://doi.org/10.17979/spudc.9788497498418.0430

the state vector not only with w as additional states, but also with their derivatives up to some predefined order, l:

$$\widetilde{x}(t) = \begin{pmatrix} x(t)^T & \theta^T & w(t)^T & \dots & w(t)^{(l)^T} \end{pmatrix}^T, \tag{5}$$

Thus, the $l-$augmented dynamics are given by:

$$\dot{\widetilde{x}}(t) = \tilde{f}\left(\widetilde{x}(t), u(t)\right) = \begin{pmatrix} f\left(x(t), \theta, u(t), w(t)\right) \\ 0_{n_\theta \times 1} \\ \dot{w}(t) \\ \vdots \\ w^{(l+1)}(t) \end{pmatrix},$$

and the $l-$augmented model is:

$$\mathcal{M}^l = \begin{cases} \dot{\widetilde{x}}(t) = \tilde{f}(\widetilde{x}(t), u(t)) & (6) \\ y(t) = g(\widetilde{x}(t), u(t)) & (7) \end{cases}$$

We adopt a differential geometry approach to assess the property FISPO. To this end, we build an augmented observability-identifiability matrix using Lie derivatives, which are defined as [14]:

$$L_{\tilde{f}} g\left(\tilde{x}, u\right) = \frac{\partial g}{\partial \tilde{x}}\left(\tilde{x}, u\right) \tilde{f}\left(\tilde{x}, u\right) + \frac{\partial g}{\partial u}\left(\tilde{x}, u\right) \dot{u}.$$

The expression above is the first order Lie derivative. The zero-order Lie derivative is the model output, $L^0_{\tilde{f}} g = g$. Higher ($i-$order) Lie derivatives are calculated recursively as:

$$\begin{aligned} L^i_{\tilde{f}} g\left(\tilde{x}, u\right) =& \frac{\partial L^{i-1}_{\tilde{f}} g}{\partial \tilde{x}}\left(\tilde{x}, u\right) \tilde{f}\left(\tilde{x}, u\right) \\ &+ \sum_{j=0}^{i-1} \frac{\partial L^{i-1}_{\tilde{f}} g}{\partial u^{(j)}}\left(\tilde{x}, u\right) u^{(j+1)}. \end{aligned}$$

Thus, the observability-identifiability matrix of $\mathcal{M}$ (6) is defined by:

$$\mathcal{O}_I\left(\tilde{x}, u\right) = \frac{\partial}{\partial \tilde{x}} \begin{pmatrix} L^0_{\tilde{f}} g\left(\tilde{x}, u\right) \\ L_{\tilde{f}} g\left(\tilde{x}, u\right) \\ L^2_{\tilde{f}} g\left(\tilde{x}, u\right) \\ \vdots \\ L^{n_{\tilde{x}}-1}_{\tilde{f}} g\left(\tilde{x}, u\right) \end{pmatrix}, \tag{8}$$

Matrix (8) can be used to assess the FISPO property using the following condition:

Theorem 1 (Observability-Identifiability Condition (OIC) [14])**.** *A model $\mathcal{M}$ given by equations* (6)–(7) *is FISPO around a (possibly generic) point in the augmented state space $\tilde{x}_0$ if the observability-identifiability matrix defined by* (8) *has full rank, that is: rank$(\mathcal{O}_I\left(\tilde{x}_0, u\right)) = n_{\tilde{x}} = n_x + n_\theta + n_w$.*

If the matrix is not full rank, there are unidentifiable parameters and/or unobservable states or inputs. However, some of the model variables (parameters, states, and unknown inputs) may still be observable. To determine the observability of the j^{th} variable, we remove the j^{th} column of the observability-identifiability matrix (8) and recalculate its rank. If the rank remains constant after removing the column, the j^{th} variable is unobservable, and if the rank decreases, the variable is observable.

3 IMPLEMENTATION OF StrikePy

3.1 DESCRIPTION AND AVAILABILITY

The Python implementation of StrikePy follows closely the original MATLAB implementation of STRIKE-GOLDD. Thus, the main algorithm is implemented in a script called `strike_goldd`, which loads the model, builds the observability-identifiability matrix, calculates its rank, and stores the results. If the model is not fully FISPO, an additional function `elim_and_recalc` analyses the observability of each variable following the procedure described in the end of Section 2.2. Finally, the user can specify options by editing a file named `options`.

StrikePy is structured in several directories. Models are stored in a folder called `models`, the documentation in `doc`, and auxiliary functions in `functions`.

StrikePy can be downloaded from two software repositories, GitHub and from PyPI:

https://github.com/afvillaverde/StrikePy

https://pypi.org/project/StrikePy/

Detailed instructions of how to use StrikePy are provided in the documentation that accompanies the toolbox. Figure 1 shows an screenshot of the execution of StrikePy.

3.2 COMPARISON WITH BMSS2

We note that a partial implementation of STRIKE-GOLDD in Python was recently made available as part of BMSS2, a sowftware tool for model selection and identifiability analysis [8]. However, the BMSS2 implementation of STRIKE-GOLDD has a number of limitations.

https://doi.org/10.17979/spudc.9788497498418.0430

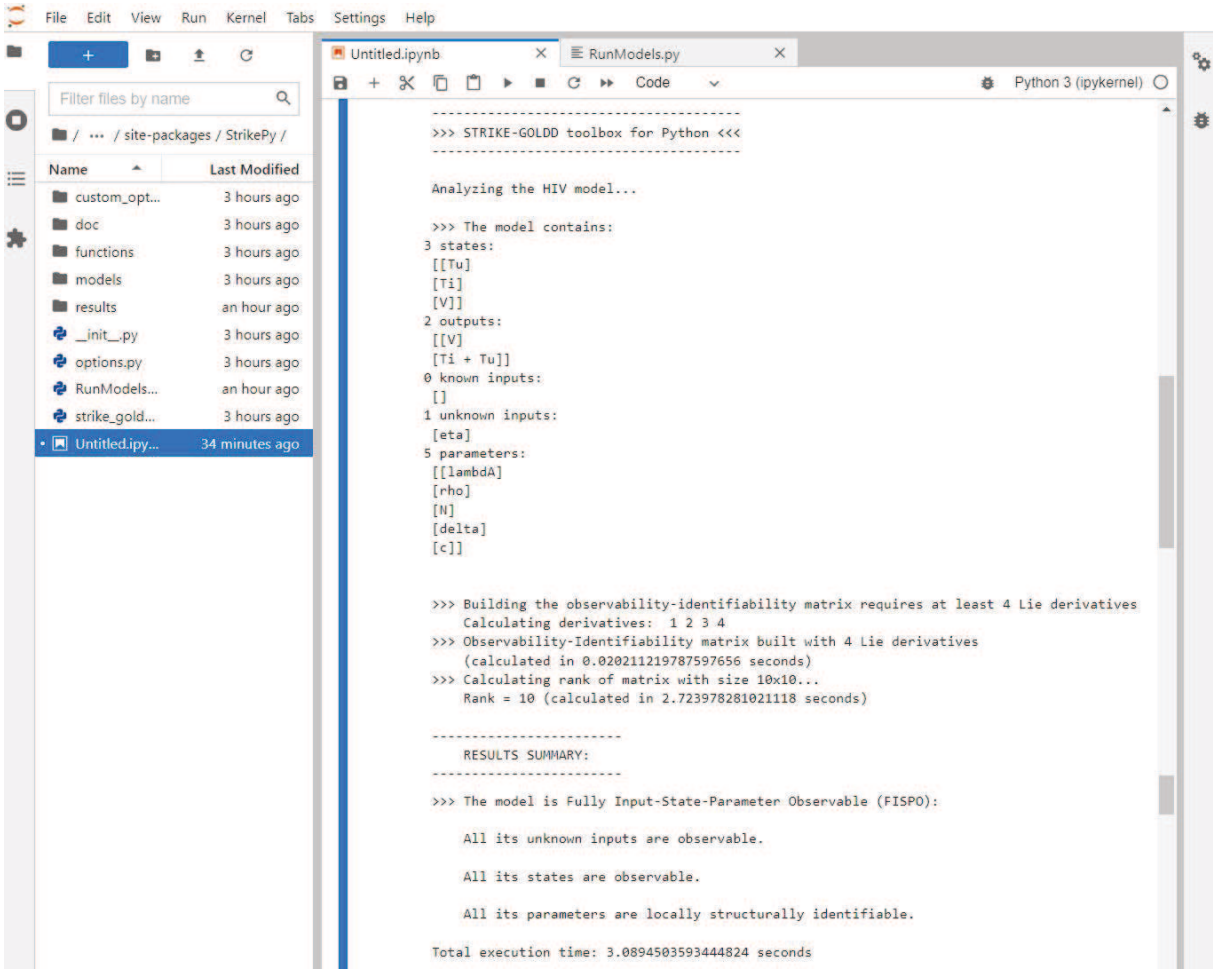

Figure 1: Screenshot of an execution of StrikePy.

The first one is conceptual, as BMSS2 does not allow for the analysis of models with unknown inputs, which is an important feature of STRIKE-GOLDD in comparison to other observability and identifiability toolboxes.

Second, the symbolic calculations performed by BMSS2 are computationally inefficient due to limitations of the methods from the sympy package that it uses to calculate the matrix rank (https://pypi.org/project/sympy/); as a result, BMSS2 is unable to analyse some models that can be analysed by StrikePy. Furthermore, while BMSS2 provides the option to perform the analysis numerically instead of symbolically (which is computationally more efficient), in this case its results vary depending on the choice of numerical values, which makes it unreliable.

Finally, we found that the use of the STRIKE-GOLDD implementation provided with BMSS2 is only briefly described in its documentation, which makes it difficult to apply the tool.

The reasons summarized above prompted us to create a new Python implementation.

4 COMPUTATIONAL RESULTS

We have assessed the performance of StrikePy by applying it to a total of 13 dynamic models of biological systems and processes. The 13 case studies were selected from the ones provided in the STRIKE-GOLDD models folder, and they are provided in the 'models' folder of StrikePy. Their names are listed in Table 1; their equations can be inspected by reading the corresponding files.

As a benchmark, we have also analysed them with the Matlab implementation of STRIKE-GOLDD. In all cases, StrikePy produced correct results, i.e. the conclusions of the analyses were identical to those obtained with STRIKE-GOLDD. These results were obtained with default settings, i.e. without the need for tuning the user options.

Another important aspect of the performance evaluation is the computational cost. The computation times are given in Table 1, where it can be noticed that for all but the smallest examples the

Table 1: Comparison of computation times, in seconds, between StrikePy (Python) and STRIKE-GOLDD (MATLAB). Each entry shows the mean and standard deviation of five runs. Results obtained with a computer with Intel Core i7 processor, 2.80 GHz, 8 GB RAM, OS Windows 10. StrikePy was executed in Python 3.9 using the IDE PyCharm Community Edition version 2020.2.2. STRIKE-GOLDD was executed in Matlab R2020b.

Model name	**StrikePy**	**STRIKE-GOLDD**
1A_integral	0.404 ± 0.361	0.595 ± 0.067
1B_prop_integral	0.324 ± 0.398	0.511 ± 0.034
1C_nonlinear	26.565 ± 1.246	2.843 ± 0.195
Bolie	56.956 ± 2.193	1.327 ± 0.065
C2M	1.499 ± 0.573	0.591 ± 0.018
C2M_unknown_input_known_b	44.21 ± 1.532	1.556 ± 0.099
CR	1.814 ± 0.558	0.663 ± 0.065
HIV	2.267 ± 0.647	0.584 ± 0.108
HIV_1	2641.3 ± 7.414	36.609 ± 40283
phos	55.438 ± 3.547	0.990 ± 0.159
PK	149.662 ± 3.468	2.707 ± 0.541
tumor	791.545 ± 34.861	28.456 ± 1.192
sirs	3818.684 ± 19.294	98.872 ± 2.748

StrikePy implementation was, as expected, slower than STRIKE-GOLDD.

Further comparisons can be found in [10], where the performance of StrikePy was benchmarked against other 10 tools for structural identifiability analysis, using a set of 25 models.

5 DISCUSSION

We have presented StrikePy, a free, open-source Python tool for analysing the structural identifiability and observability of ODE models. This toolbox addresses the need for a toolbox capable of performing said analyses in Python, which is a popular programming language among the systems biology community. While another Python implementation – included in a toolbox named BMSS2 – was already available, it contained a number of limitations that are not present in StrikePy.

Our tests indicate that StrikePy is computationally less efficient than STRIKE-GOLDD, the Matlab toolbox that it implements. This was to be expected, since Matlab's symbolic math toolbox outperforms the currently available Python packages. In this aspect there is room for improvement, which may be achieved in the future as a result of new developments in Python routines.

Acknowledgement

This work has received funding from MCIN/AEI/10.13039/501100011033 through grant PID2020-113992RA-I00 (PREDYCTBIO), from the Xunta de Galicia, Consellería de Cultura, Educación e Universidade through grant ED431F 2021/003, and from grant RYC-2019-027537-I funded by MCIN/AEI/10.13039/501100011033 and by "ESF Investing in your future".

References

[1] R. Bellman and K. J. Åström. On structural identifiability. *Math. Biosci.*, 7(3):329–339, 1970.

[2] O.-T. Chiş, J. R. Banga, and E. Balsa-Canto. Structural identifiability of systems biology models: a critical comparison of methods. *PLoS One*, 6(11):e27755, 2011.

[3] J. DiStefano III. *Dynamic systems biology modeling and simulation*. Academic Press, 2015.

[4] R. Hermann and A. J. Krener. Nonlinear controllability and observability. *IEEE Trans. Autom. Control*, 22(5):728–740, 1977.

[5] R. E. Kalman. Contributions to the theory of optimal control. *Bol. Soc. Mat. Mexicana*, 5(2):102–119, 1960.

[6] A. Martinelli. Nonlinear unknown input observability: Extension of the observability rank condition. *IEEE Trans. Autom. Control*, 64(1):222–237, 2019.

[7] H. Miao, X. Xia, A. S. Perelson, and H. Wu. On identifiability of nonlinear ODE models and applications in viral dynamics. *SIAM Rev.*, 53(1):3–39, 2011.

 https://doi.org/10.17979/spudc.9788497498418.0430

[8] R. K. J. Ngo, J. W. Yeoh, G. H. W. Fan, W. K. S. Loh, and C. L. Poh. BMSS2: a unified database-driven modelling tool for systematic model selection and identifiability analysis. *bioRxiv*, 2021.

[9] A. Raue, J. Karlsson, M. P. Saccomani, M. Jirstrand, and J. Timmer. Comparison of approaches for parameter identifiability analysis of biological systems. *Bioinformatics*, 30(10):1440–1448, 2014.

[10] X. Rey Barreiro and A. F. Villaverde. Benchmarking tools for *a priori* identifiability analysis. *arXiv preprint*, 2022.

[11] E. T. Tunali and T.-J. Tarn. New results for identifiability of nonlinear systems. *IEEE Trans. Autom. Control*, 32(2):146–154, 1987.

[12] A. F. Villaverde. Observability and structural identifiability of nonlinear biological systems. *Complexity*, 2019:8497093, 2019.

[13] A. F. Villaverde, A. Barreiro, and A. Papachristodoulou. Structural identifiability of dynamic systems biology models. *PLoS Comput. Biol.*, 12(10):e1005153, 2016.

[14] A. F. Villaverde, N. Tsiantis, and J. R. Banga. Full observability and estimation of unknown inputs, states, and parameters of nonlinear biological models. *J. R. Soc. Interface*, 16:20190043, 2019.

[15] F.-G. Wieland, A. L. Hauber, M. Rosenblatt, C. Tönsing, and J. Timmer. On structural and practical identifiability. *Curr. Opin. Syst. Biol.*, 2021.