STRIKE-GOLDD v3.2
User manual

Alejandro F. Villaverde

afvillaverde@uvigo.gal —— http://afvillaverde.webs.uvigo.gal/

May 24, 2022

Contents

1 Introduction

STRIKE-GOLDD is a MATLAB toolbox that analyses the local structural identifiability and observability of nonlinear dynamic models, which can have multiple time-varying and possibly unknown inputs. It can also be used to find the symmetries in the model equations that lead to lack of identifiability or observability, and to automatically reparameterize the model in order to remove those symmetries.

1.1 Theoretical foundations

STRIKE-GOLDD adopts a differential geometry approach, recasting the identifiability problem as an observability problem. Essentially, the observability of the model variables (states, parameters, and 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, there are some unobservable variables. If these variables are parameters, they are called (structurally) unidentifiable.

STRIKE-GOLDD determines the subset of identifiable parameters, observable states, and observable (also called reconstructible) inputs, thus performing a "Full Input-State-Parameter Observability" analysis (FISPO). STRIKE-GOLDD provides three different algorithms that perform this analysis:

- the <u>FISPO</u> algorithm presented in [1], which is implemented in the `STRIKE-GOLDD.m` file and can be applied to nonlinear models in general. Said procedure 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 decomposition into more tractable submodels, which can be either defined by the user or found via combinatorial optimization. Another possibility is to build observability-identifiability matrices with a reduced number of Lie derivatives. In some 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 <u>ORC-DF</u> algorithm [2] originally presented in [3], which can be used if the model is affine in the inputs.
- the <u>Prob_Obs_Test</u> algorithm originally presented in [4], which can be used if the model is rational.

Furthermore, STRIKE-GOLDD can also perform certain analyses related to the existence of Lie symmetries in the model equations, which are a possible source of non-identifiability and non-observability:

- the <u>Lie_symmetry</u> function searches for the Lie symmetries that prevent a model from being FISPO.
- the <u>AutoRepar</u> function removes the symmetries found by Lie_symmetry procedure, thus reparameterizing the model so as to render it FISPO.

1.2 Version and publication history: main releases

- The first version of STRIKE-GOLDD was presented in [5].
- STRIKE-GOLDD v2.0 introduced the use of extended Lie derivatives [6]. This extension enabled the analysis of structural identifiability for *time-varying inputs* and, additionally, the characterization of the input profile required to make the parameters identifiable.
- STRIKE-GOLDD v2.1.1 incorporated the possibility of analysing models with *unknown time-varying inputs*, thus performing a FISPO analysis for the first time. The corresponding methodological details are given in [1].
- STRIKE-GOLDD v2.1.6 incorporated a procedure for finding *Lie symmetries* in a model, as well as for calculating the transformations that break them. The procedure is described in [7].
- STRIKE-GOLDD v2.2 incorporated the *ORC-DF* algorithm [3] for models that are affine in the inputs. A comparison between FISPO and ORC-DF is reported in [2]. STRIKE-GOLDD v2.2 also incorporated a multi-experiment analysis routine.

- STRIKE-GOLDD v3.0 incorporated an automatic reparameterization procedure, AutoRepar [8], which removes Lie symmetries to render the model FISPO.
- STRIKE-GOLDD v3.2 incorporated the Prob_Obs_Test algorithm, which is computationally more efficient than the FISPO algorithm and can be applied to rational models.

2 License

STRIKE-GOLDD is licensed under the GNU General Public License version 3 (GPLv3), a free, copyleft license for software.

3 Availability

STRIKE-GOLDD can be downloaded from `https://github.com/afvillaverde/strike-goldd`.

4 Software contents

The STRIKE-GOLDD v3.2 toolbox consists of several MATLAB files stored in several folders. The user typically interacts with the following three files, which are included in the root folder:

- `install.m`: installation file, which must be executed at the beginning of each session.
- `options.m`: file where the user enters the problem to solve and the options for solving it.
- `STRIKE-GOLDD.m`: main file; running it executes STRIKE-GOLDD. It implements the FISPO algorithm (which is the default one), and has calls to other functions.

Additionally, a number of scripts are provided in the functions folder:

- `AutoRepar.m`: implements the AutoRepar method [8], which reparameterizes the model specified in the `options.m` file, removing any Lie symmetries found in order to make it fully identifiable and observable. This entails analysing the structural identifiability and observability of the model; if this has already been done, the results file can be specified in `options.m`.
- `build_OI_ext.m`: builds the generalized observability-identifiability matrix for a given number of Lie derivatives, which is passed as the argument. The resulting array is stored in a MAT file.
- `decomp.m`: decomposes the model into submodels (either defined by the user, or found via optimization) and analyses their identifiability and observability.
- `elim_and_recalc.m`: determines the identifiability of individual parameters one by one, by successive elimination of its column in the identifiability matrix and recalculation of its rank.
- `graph_model.m`: creates a graph showing the relations among model states, outputs, and parameters (i.e. which variables appear in which equations).
- `Lie_Symmetry.m`: searches for Lie symmetries using the method in [7]; if it finds symmetries admitted by the model, it calculates transformations of the variables in order to break them.
- `ME_analysis.m`: performs Multi-Experiment analysis; it modifies the model equations so that the analysis of the resulting model yields observability results from multiple experiments (by default, STRIKE-GOLDD considers a single experiment).
- `ORC_DF.m`: implements the ORC-DF algorithm [2] originally presented in [3], which can be used if the model is affine in the inputs.
- `Prob_Obs_Test.m`: implements the Prob_Obs_Test algorithm originally presented in [4], which can be used if the model is rational.

The folder tree structure of the toolbox files is as follows:

Root folder (/STRIKE-GOLDD/)

- install.m
- options.m
- STRIKE-GOLDD.m
- doc
  - AutoRepar_supplementary_information.pdf
  - STRIKE-GOLDD_manual.pdf
- functions
  - AutoRepar.m
  - build_OI_ext.m
  - decomp.m
  - elim_and_recalc.m
  - graph_model.m
  - Lie_symmetry.m
  - ME_analysis.m
  - ORC_DF.m
  - Prob_Obs_Test.m
  - aux_AutoRepar
    - (...)
  - aux_Decomposition
    - (...)
  - aux_Lie_symmetry
    - (...)
  - aux_Prob_Obs_Test
    - (...)
- models
  - (*models to be analysed by the toolbox, includes several provided models*)
- results
  - (*output files generated by the toolbox*)

5 Requirements and installation

5.1 Requirements

STRIKE-GOLDD can run on any operating system compatible with MATLAB. Dependencies:

- The MATLAB Symbolic Math Toolbox.
- (OPTIONAL:) The MATLAB MEIGO toolbox [9], which can be freely downloaded from `http://gingproc.iim.csic.es/meigom.html`. The MEIGO toolbox is only needed if the optimization-based model decomposition is used.

STRIKE-GOLDD v3.2 has been tested with MATLAB versions R2017B (Symbolic Math Toolbox Version 8.0), R2019B (Symbolic Math Toolbox Version 8.4), R2020B (Symbolic Math Toolbox Version 8.6), and R2021B (Symbolic Math Toolbox Version 9.0).

5.2 Download and install

1. Download STRIKE-GOLDD from `https://github.com/afvillaverde/strike-goldd`.
2. Unzip it.
3. (OPTIONAL STEP–only needed to perform optimization-based decomposition:)
   (a) Download MEIGO from: `http://gingproc.iim.csic.es/meigom.html`.
   (b) Unzip the MEIGO folder.
   (c) Specify the location of MEIGO by modifying the corresponding line in the `options.m` file as follows (replace the example below with the actual location in your computer):
   `paths.meigo = 'C:\Users\My_name\Documents\MEIGO_M-v03-07-2014\MEIGO_M';`

6 Quick start: using STRIKE-GOLDD in one minute

To start using STRIKE-GOLDD, simply follow these steps:

1. Follow the installation instructions in Section 5.2.
2. Open a MATLAB session and go to the STRIKE-GOLDD root directory ("STRIKE-GOLDD").
3. Run `install.m`.
4. Define the problem and options by editing the script `options.m` (see Section 7.2 for details).
   - QUICK DEMO EXAMPLE: If you skip this step and leave `options.m` unedited, STRIKE-GOLDD will analyse a two-compartment model with default options.
5. Run `STRIKE-GOLDD.m` (to do this you can either type "STRIKE-GOLDD" in the command window, or right-click `STRIKE-GOLDD.m` in the "Current Directory" tab and select "run").

Done! Results will be reported in the MATLAB screen. A screenshot of an execution is shown in Fig. 1.

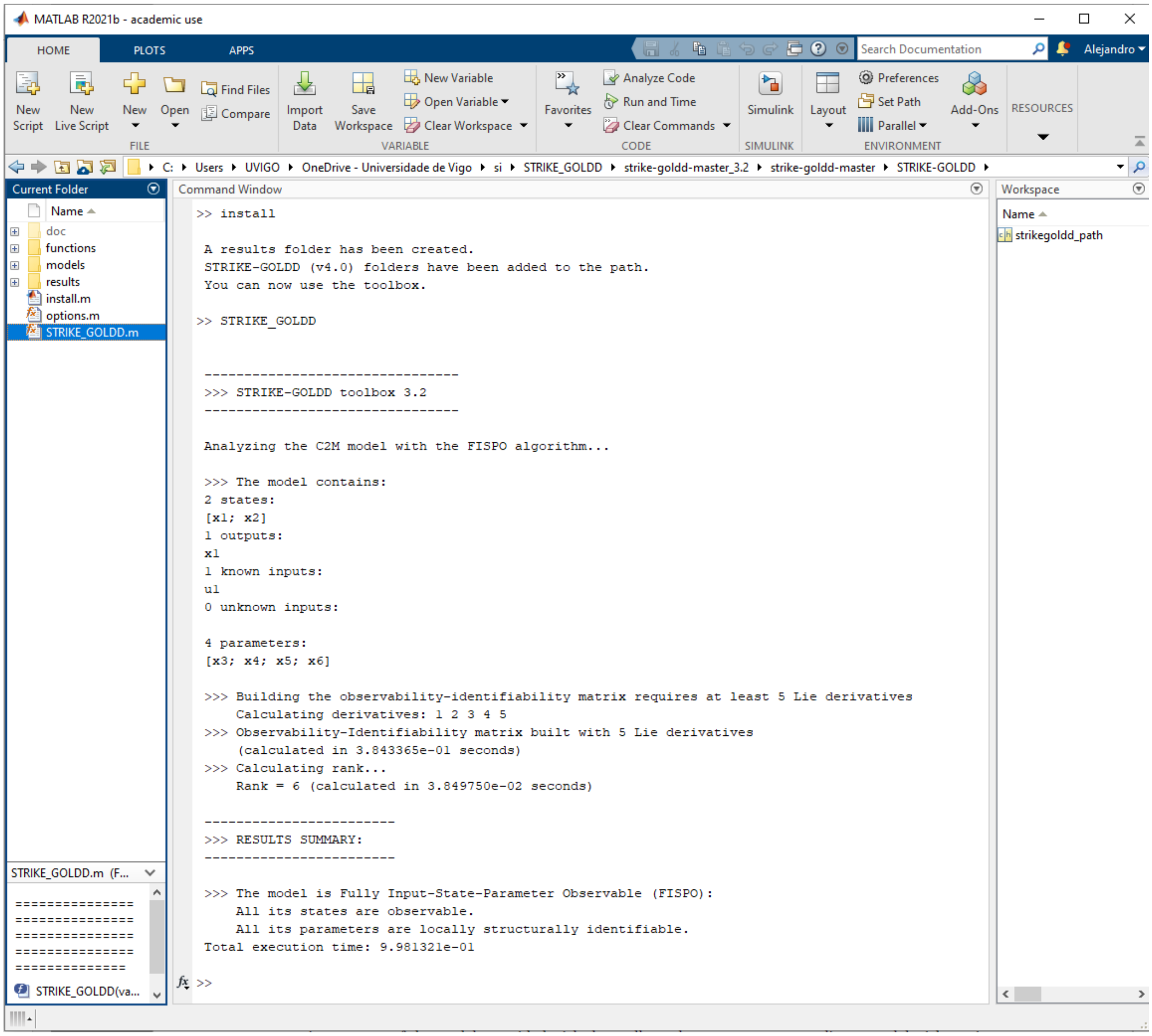

Figure 1: **Screen shot of a STRIKE-GOLDD run.**

7 Usage

7.1 Options

The model to analyse, as well as the options for performing the analysis, are entered in the `options.m` file. All options have default values that can be modified by the user. Their meaning is explained in the comments of the `options.m` file. In the `options.m` file the options are classified in the following blocks:

(1) **CHOOSE MODEL TO ANALYSE:** The first block consists of solely one option, the name of the model to analyse. By default it is set to one of the models provided with the toolbox, the two-compartment linear model with one input:

```
modelname = 'C2M';
```

The user may select other models provided with the toolbox – included in folder `/models` – or define a new model as explained in Section 7.2.

(2) **CHOOSE TYPE OF ANALYSIS:** In this block, the user can select among 5 analyses to perform:

```
opts.algorithm = 1;
```

Choose (1) for the FISPO algorithm, (2) for Prob_Obs_Test, (3) for ORC-DF, (4) for Lie_symmetry, and (5) for AutoRepar.

(3) **MAIN STRUCTURAL IDENTIFIABILITY & OBSERVABILITY (SIO) OPTIONS:** The third block consists of the following options:

`opts.maxLietime`, `opts.nnzDerU`, `opts.nnzDerW`, `opts.numeric`, `opts.replaceICs`, `prev_ident_pars`. Their meaning is explained in the comments of the `options.m` file.

Important: for models with several inputs, `opts.nnzDerU` and `opts.nnzDerW` must be row vectors with the same number of elements as inputs, e.g., for two inputs:

```
opts.nnzDerU = [0 1];
```

The optional vector `prev_ident_pars` is used for entering parameters that have already been classified as identifiable. This reduces the computational complexity of the calculations and may thus enable a deeper analysis, which can lead to more complete results. For example, if STRIKE-GOLDD has already determined that two parameters p_1 and p_2 are identifiable, we may enter:

```
syms p1 p2
prev_ident_pars = [p1 p2];
```

Otherwise, we must leave it blank:

```
prev_ident_pars = [];
```

This option can also be used to assume that some parameters are known, despite being entered as unknown in the model definition. This is useful to test what happens when fixing some parameters, without having to modify the model file.

(4) **OPTIONS FOR AFFINE SYSTEMS (apply only to the ORC-DF algorithm):** If the ORC-DF algorithm is selected *and* the model is indeed affine, it is possible to specify additional settings, including the use of parallelization.

(5) **DECOMPOSITION OPTIONS (apply only to the FISPO algorithm):** If the user wishes to decompose the model for its analysis, it can be specified in this block. It is possible to set a number of options, as well as to define the submodels to analyse if the model is decomposed. They *only* need to be specified in this way if the user wants to define them manually instead of relying on the optimisation algorithm. In the former case, every submodel must be specified as a vector containing the indices of the states included in it. For example, the following lines define two submodels, consisting of states [x(1), x(2)] and [x(2), x(3)], respectively:

```
submodels    = [];
submodels{1} = [1 2];
submodels{2} = [2 3];
```

(6) **MULTI-EXPERIMENT OPTIONS (apply to the 3 SIO algorithms):** If `opts.multiexp` is set to one, the observability of the model is analysed for multiple experiments. The number of experiments considered is set with `opts.multiexp_numexp`. For each experiment it is possible to specify different types of inputs and initial conditions.

(7) **LIE SYMMETRIES & REPARAMETERIZATION OPTIONS:** This block defines a number of options related to the search for Lie symmetries, and to the reparameterization of the model based on said symmetries: `opts.ansatz`, `opts.degree`, `opts.tmax`, and `opts.ode_n` tune the search for symmetries, while `opts.use_existing_results` and `opts.results_file` let the user specify if the structural identifiability and observability of the model has already been analysed.

(8) **PATHS:** In this block the user only needs to modify the path of the MEIGO toolbox (although even this can be skipped if the model is *not* decomposed using optimization):

```
paths.meigo  = 'C:\Users\My_name\Documents\MEIGO_M-v03-07-2014\MEIGO_M';
```

(9) **DEPRECATED OPTIONS (FOR COMPATIBILITY WITH OLDER VERSIONS):** The final block lists a number of options that are no longer supported, bt may be encountered in older versions.

7.2 Input: entering a model

STRIKE-GOLDD reads models stored as MATLAB MAT-files (.mat). The model states, outputs, inputs, parameters, and dynamic equations must be defined as vectors of symbolic variables; the names of these vectors must follow the specific convention shown in Table 1. **Important:** **x, p, u, w, f, h** are reserved names, which must be used for the variables and/or functions listed in Table 1 and cannot be used to name any other variables. However, it is possible to use variants of them, e.g. $x_1, x_2, p_{23}, xp, \ldots$

7.2.1 Example: defining the MAPK model

Here we illustrate how to define a model using the MAPK example included in the `models` folder. The file read by STRIKE-GOLDD is `MAPK.mat`. This file, which stores the model variables, can be created from the M-file `z_create_MAPK_model.m`. In the following lines we comment the different parts of the M-file, illustrating the process of defining a suitable model.

First, all the parameters, states, and any other entities (such as inputs or known constants) appearing in the model must be defined as symbolic variables:

Name	Reserved for:	Common mathematical notation:
x	state vector	$x(t)$
p	unknown parameter vector	θ
u	known input vector	$u(t)$
w	unknown input vector	$w(t)$
f	dynamic equations	$\dot{x}(t) = f(x(t), u(t), w(t), \theta)$
h	output function	$y = h(x(t), u(t), w(t), \theta)$

Table 1: **reserved variable and function names.** The names in the table are reserved for certain variables and functions. They must not be used for naming arbitrary model quantities. However, it is possible to use variants of them, e.g. $x_1, x_2, p_{23}, xp, \dots$

```
syms k1 k2 k3 k4 k5 k6 ...
     ps1 ps2 ps3 ...
     s1t s2t s3t ...
     KK1 KK2 n1 n2 alpha  ...
```

Then we define the state variables, by creating a column vector named x:

```
x    = [ps1; ps2; ps3];
```

Similarly, we define the vector of output variables, which must be named h. In this case they coincide with the state variables:

```
h    = x;
```

Similarly, we define the known input vector, u, and the unknown input vector, w. If there are no inputs, enter blank vectors:

```
u  = [];
w  = [];
```

The vector of unknown parameters must be called p:

```
p    = [k1; k2; k3; k4; k5; k6;s1t; s2t; s3t; KK1; KK2; n1; n2; alpha];
```

The dynamic equations dx/dt must also be entered as a column vector, called f. It must have the same length as the state vector x:

```
f    = [k1*(s1t-ps1)*(KK1^n1)/(KK1^n1+ps3^n1)-k2*ps1;
            k3*(s2t-ps2)*ps1*(1+(alpha*ps3^n2)/(KK2^n2+ps3^n2))-k4*ps2 ;
            k5*(s3t-ps3)*ps2-k6*ps3 ];
```

The vector of initial conditions must be called *ics*. If you want to analyse the model for generic initial conditions (which is the most common case), enter a blank vector:

```
ics  = [];
```

Additionally we define another vector, `known_ics`, to specify which generic initial conditions should be replaced with the specific values defined in `ics` during the analysis. The vector `known_ics` must have the same length as the state vector x, and its entries should be either 1 or 0, depending on whether the corresponding initial condition is replaced or not, respectively:

```
known_ics = [0,0,0];
```

Finally, save all the variables in a MAT-file:

```
save('MAPK','x','h','u','w','p','f','ics','known_ics');
```

7.3 Analysing a model: known vs unknown inputs

The use of STRIKE-GOLDD for analysing a model was already illustrated in section 6. This section provides a few more details, basically about the use of models with known and unknown inputs.

7.3.1 Example: two-compartment model with known input

Section 6 showed how to analyse the default example, which is a two-compartment model with a known input, using default settings. By default, the option `opts.nnzDerU` is set to `opts.nnzDerU = 1` in the options file. This means that the model is analysed with exactly one non-zero derivative of the known input.

if we set `opts.nnzDerU = 0`, all input derivatives are set to zero. Running the two-compartment model example with this setting yields that the model is unidentifiable. Hence, this model requires a ramp or a higher-order polynomial input to be structurally identifiable and observable.

Note that for models with several inputs it is necessary to specify a vector, e.g. `opts.nnzDerU = [0 1]` for two inputs (or any other numbers, e.g. `opts.nnzDerU = [2 2]`).

7.3.2 Example: two-compartment model with unknown input

Let us now consider the two-compartment model with unknown input, and with the parameter b considered as known. This is already implemented in the model file `C2M_unknown_input_known_b` provided with the toolbox. The analysis of this model yields that all its parameters are structurally unidentifiable and its unmeasured state and input are unobservable. This is obtained for any choice of `opts.nnzDerW` (0, 1, 2, ...).

We now consider a version of this model in which both b and k_{1e} are considered known. This is implemented in the file `C2M_unknown_input_known_b_k1e`. In this case, the analysis yields that the model is fully observable (FISPO). This is obtained for any choice of `opts.nnzDerW` (0, 1, 2, 3, ...).

7.4 Searching for Lie symmetries

The existence of Lie symmetries amounts to lack of observability (in the generalized sense, that is, full input, state, and parameter observability, or 'FISPO'). Thus, determining whether a model admits a Lie group of transformations is a way of determining if it is observable and structurally identifiable.

The analysis of Lie symmetries can be performed by running the function `Lie_Symmetry`. It can be called in two different ways:

1. Without arguments, in which case it searches for symmetries in the model specified in the `options` file.
2. Specifying the model file as the argument, e.g.: `Lie_Symmetry('PK')`.

Furthermore, a number of options can be tuned. They are explained in the first lines of the script `Lie_Symmetry.m`. To modify them, directly edit the corresponding lines at the beginning of the script.

7.5 Reparameterizing a model

If a model is not fully identifiable or observable, the ideal solution is to reparameterize it so that it becomes FISPO. This can be done automatically with the `AutoRepar.m` function. By executing it, STRIKE-GOLDD performs a number of steps:

1. First it analyses the structural identifiability and observability of the model. If this analysis has already been done, this step may be skipped by setting `opts.use_existing_results = 1` in the options file, and specifying the mat-file that stores the results in `opts.results_file`.

2. Then, if the model is not FISPO, it searches for the Lie symmetries that cause this lack of identifiability (this step automatically calls `Lie_Symmetry.m`).

3. Finally, it uses the information provided by the Lie symmetries to introduce symmetry-breaking transformations that make the model FISPO.

In step 3 there may be several possible transformations. In this case, the toolbox lets the user decide which parameters or states can be transformed and which ones should remain intact if possible.

The AutoRepar methodology is described in [8], which provides several examples.

7.6 Output

STRIKE-GOLDD reports the main results of the identifiability analysis on screen. Additionally, it creates several MAT-files in the `results` folder:

- A file named `id_results_MODELNAME_DATE.mat`, with the results of the identifiability analysis and most of the intermediate results. The main results are: `p_id` (list of identifiable parameters), `p_un` (unidentifiable parameters), `obs_states` (observable states), and `unobs_states` (unobservable states).

- One or several files named `obs_ident_matrix_MODEL_NUMBER_OF_Lie_deriv.mat`, with the generalized observability-identifiability matrices calculated with a given number of Lie derivatives. They are stored in separate files so that they can be reused in case a particular run is aborted due to excessive computation time.

- If decomposition is used, STRIKE-GOLDD creates a subfolder in the `results` folder named `decomp_MODEL_DATE_MAXSTATES_MAXLIETIME` (i.e., it specifies the model name, the date, the maximum number of states allowed in every submodel, and the maximum time allowed for performing a Lie derivative). Inside the folder it stores one MAT-file per submodel with partial results. Additionally, the same MAT-file as described in the previous point is created in the `results` folder.

8 Contributors

STRIKE-GOLDD has been developed mainly by Alejandro F. Villaverde (Univ. Vigo, `afvillaverde@uvigo.gal`). The code for finding Lie symmetries and reparameterizing the model based on them was written by Gemma Massonis Feixas (CSIC). The implementation of the ORC-DF algorithm and the automatic transformation for multi-experiment analysis was done by Nerea Martínez (Univ. Vigo & CSIC). The Prob_Obs_Test algorithm was implemented by Sandra Díaz Seoane (Univ. Vigo) A number of collaborators have contributed to theoretical and/or application aspects: Antonio Barreiro Blas (Univ. Vigo), Antonis Papachristodoulou (Oxford Univ.), Neil D. Evans (Warwick Univ.), Michael J. Chappell (Warwick Univ.), Julio R. Banga (CSIC), Nikolaos Tsiantis (CSIC), and Xabier Rey Barreiro (Universidade de Vigo). Alejandro F. Villaverde thanks Gleb Pogudin (École Polytechnique, Institut Polytechnique de Paris) for helpful suggestions.

9 Funding

STRIKE-GOLDD has received funding from the Xunta de Galicia, Consellería de Cultura, Educación e Universidade through the I2C postdoctoral program (fellowship ED481B2014/133-0) and through grant ED431F 2021/003; from the Spanish Ministry of Economy and Competitiveness (MINECO) through grants DPI2013-47100-C2-2-P and DPI2017-82896-C2-2-R; from the EPSRC projects EP/M002454/1 and EP/J012041/1; from the European Union's Horizon 2020 research and innovation programme under grant agreement No 686282 ("CANPATHPRO"); from the CSIC intramural project grant PIE 202070E062 ("MOEBIUS"); from grant RYC-2019-027537-I funded by MCIN/AEI/ 10.13039/501100011033 and by "ESF Investing in your future"; and from grant PID2020-113992RA-I00 funded by MCIN/AEI/ 10.13039/501100011033 (PREDYCT-BIO).

References

[1] Villaverde AF, Tsiantis N, Banga JR. Full observability and estimation of unknown inputs, states and parameters of nonlinear biological models. Journal of the Royal Society Interface. 2019;16(156):20190043.

[2] Martínez N, Villaverde AF. Nonlinear observability algorithms with known and unknown inputs: analysis and implementation. Mathematics. 2020;8(11):1876.

[3] Maes K, Chatzis M, Lombaert G. Observability of nonlinear systems with unmeasured inputs. Mechanical Systems and Signal Processing. 2019;130:378-94.

[4] Sedoglavic A. A probabilistic algorithm to test local algebraic observability in polynomial time. J Symb Comput. 2002;33:735-55.

[5] Villaverde AF, Barreiro A, Papachristodoulou A. Structural identifiability of dynamic systems biology models. PLoS Computational Biology. 2016;12(10):e1005153.

[6] Villaverde AF, Evans ND, Chappell MJ, Banga JR. Input-Dependent Structural Identifiability of Nonlinear Systems. IEEE Control Systems Letters. 2019;3(2):272-7.

[7] Massonis G, Villaverde AF. Finding and breaking Lie symmetries: implications for structural identifiability and observability in biological modelling. Symmetry. 2020;12(3):469.

[8] Massonis G, Banga JR, Villaverde AF. AutoRepar: A method to obtain identifiable and observable reparameterizations of dynamic models with mechanistic insights. Internat J Robust Nonlinear Control. 2021.

[9] Egea JA, Henriques D, Cokelaer T, Villaverde AF, MacNamara A, Danciu DP, et al. MEIGO: an open-source software suite based on metaheuristics for global optimization in systems biology and bioinformatics. BMC Bioinformatics. 2014;15:136.