symmetryDetetction- A Python program for detecting structural non-identifiabilities in ODE models

Benjamin Merkt, Physikalisches Institut, Universität Freiburg
For questions and problems feel free to contact merktbenjamin@gmail.com.

General Information

For a detailed discussion of the underlying theory of the calculations performed by this python script, see [Merkt et al., 2015].

The required software to run symmetryDetection.py is Python version 2.6.x or 2.7.x together with the symbolical manipulation package SymPy version 2.7.2 or newer. Furthermore, a recent version of SciPy/NumPy is required.

To see all the possible arguments of the program type

```
python symmetryDetection.py --help
```

In the simplest case, the program requires two arguments:

```
python symmetryDetection.py model observation
```

The first argument `model` specifies a .csv file containing the model definition. It is expected to have the form

		x_1	x_2	$\cdots$
description 1	v_1	S_{11}	S_{21}	
description 2	v_2	S_{12}	S_{22}	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	

were x_i are the names of the variables, v_j the expression of the flows and S_{ji} the elements of the stoichiometry matrix. The descriptions in the first column are optional and not used in the analysis (however, the column itself must exist). If a different delimiter than "," is used, this has to be specified by the additional argument `--delim DELIM`. The differential equations are then given by

$$\dot{\mathbf{x}}(t) = S \cdot \boldsymbol{v}(t).$$

The second file, `observation` is a plain text file containing one observation function per row:

$$\begin{aligned} Obs_1 &= g^1(x) \qquad (1) \\ Obs_2 &= g^2(x) \\ \vdots &= \vdots \end{aligned}$$

The variables Obs_j are the names of the observed variables which are assumed to be fixed.The algebraic expressions $g^j(x)$ define the observation in terms of the variables x_i given in the model file. Note that the file `observation` may be empty if the analysis is to be performed without observations.

Examples

The corresponding files for the reaction

$$A + A \xrightarrow{k} \varnothing$$

with ODEs

$$\dot{A}(t) = -2\,k\,A^2(t)$$

and observation

$$A^{Obs}(t) = s_1 \, \frac{A(t)}{1 + s_2 A(t)}.$$

are provided in the folder "/examples" and named "simple.csv" and "simple_obs.txt". The analysis for this model is started by running

```
python symmetryDetection.py examples/simple.csv examples/simple_obs.txt
```

in a command line from the folder containing the script `symmetryDetection.py`. Note that on a Windows machine all slashes "/" have to be replaced by backslashes "\". The analysis reveals two symmetries, one pure scaling and one higher order transformation.

The optional argument `--pMax` specifies the highest power in the polynomial ansatz of the infinitesimals. For example, if the previous example is run with a highest power of one by

```
python symmetryDetection.py examples/simple.csv examples/simple_obs.txt --pMax 1
```

the second order transformation is not found.

With the argument `--ansatz`, a different polynomial ansatz can be chosen. The default value `uni` corresponds to a univariate polynomial ansatz, `par` to multivariate polynomials which also depend on the parameters and `multi` to an ansatz which depends on all variables. The example

$$\begin{tikzcd}
{} \arrow[r, "u"] & x_1 \arrow[r, "k_2"] \arrow[d, "k_1"'] & x_3 \arrow[r, "k_4"] & {} \\
 & x_2 \arrow[ur, "k_3"] \arrow[r, shift left, "k_5"] \arrow[d, "k_7"'] & x_4 \arrow[l, shift left, "k_6"] & \\
 & {} & &
\end{tikzcd}
.$$

from [Sedoglavic, 2002] with fixed input u and model equations

$$\begin{aligned}
\dot{x}_1 &= u - (k_1 + k_2)x_1 & x_2^{Obs} &= s_2 x_2 \\
\dot{x}_2 &= k_1 x_1 - (k_3 + k_6 + k_7)x_2 + k_5 x_4 & x_3^{Obs} &= s_3 x_3 \\
\dot{x}_3 &= k_2 x_1 + k_3 x_2 - k_4 x_3 & & \\
\dot{x}_4 &= k_6 x_2 - k_5 x_4 & &
\end{aligned}$$

does not admit transformations with univariate infinitesimals. However, there is a symmetry with multivariate infinitesimals which can be found by running

```
python symmetryDetection.py examples/sedoglavic.csv examples/sedoglavic_obs.txt
                                                    --ansatz par --input u --fixed u
```

The argument `--input u` specifies the variable u as an input and `--fixed u` results in no infinitesimal ansatz for this input, i.e. no transformation of u. To speed up the calculation, the first step of the algorithm can be done in parallel. To this end, the program is called with the option `--parallel PARALLEL` where `PARALLEL` is the maximal number of processes started. For example, one a quad core machine, the command

```
python symmetryDetection.py examples/sedoglavic.csv examples/sedoglavic_obs.txt
                                      --ansatz par --input u --fixed u --parallel 4
```

could be used. Not that when running on Windows, for small and simple models this can actually increase computation time because of the long startup time of new processes.

By the optional positional argument `predtiction_path`, a set of candidate predictions can be check. It is expected to have the same structure as the observation file (1) with one prediction per line. In case of the NFκB pathway, the predictions can be checked by running

```
python symmetryDetection.py examples/NFkB.csv examples/NFkB_obs.txt
                              examples/NFkB_pred.txt --input pIkk
```

The first prediction admits all transformation and therefore is possible despite the structural non-identifiability. In the second prediction which is the prediction of total phosphorylated IκBα, for the first transformation to be admitted, the predicted quantity PIKB needs a infinitesimal $\eta_{\mathrm{PIKB}} = -\mathrm{PIKB}$ which corresponds to a scaling transformation. Therefore, total phosphorylated IκBα can only be predicted on a relative scale. Similarly, the prediction of the IκBα mRNA is only possible on a relative scale due to transformations #1 and #4.

Initial conditions can be also checked by the argument `--initial INITIAL`. For example, this is necessary in the model of the JAK/STAT pathway analyzed in [Raue et al., 2014]. To start the analysis of that model, use the command

```
python symmetryDetection.py examples/JAK_STAT.csv examples/JAK_STAT_obs.txt
        --initial examples/JAK_STAT_init.txt --input u1 --fixed c1 c2 u1
```

By comparing the found symmetries with the results of [Raue et al., 2014] one sees that the same parameters are found to be non-identifiable. To see that that initial conditions can have a significant impact on the identifiability of the parameters, compare the results of

```
python symmetryDetection.py examples/NFkB.csv examples/NFkB_obs.txt
                    --initial examples/NFkB_init1.txt --input pIkk
```

with those of

```
python symmetryDetection.py examples/NFkB.csv examples/NFkB_obs.txt
                    --initial examples/NFkB_init2.txt --input pIkk
```

This initial values in the first file conserve all symmetries because it represents the steady state of the model. In the second file, the start concentration of NFκB$\circ$IκBα and nIκBα are fixed to a constant which removes two symmetries

If a transformation is admitted by a set of equations, its infinitesimals can always be multiplied by a common parameter factor and the symmetry conditions are still fulfilled. By default, these solutions of the systems are removed from the output. If they are also of interest, they can be displayed with the option `--allTrafos`.

The option `-t` or `--timeTrans` can be used to also search for symmetries including scalings of time. Here it is assumed that all differential equations, observations etc. do not explicitly depend on time. Furthermore the ansatz ignores cases in which the transformations of other variables/parameters could depend on time. In this case, scalings are the only possible transformation of time besides trivial translations which are always admitted for this variable.

Finally, if the colors and Unicode characters in the standard output lead to problems or if a more compact output is desired pretty printing can be switched of with the option `--notPretty`.

References

[Merkt et al., 2015] Merkt, B., Timmer, J., and Kaschek, D. (2015). Higher-order Lie symmetries in identifiability and predictability analysis of dynamic models. *Phys. Rev. E*, 92(1):12–20.

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

[Sedoglavic, 2002] Sedoglavic, A. (2002). A probabilistic algorithm to test local algebraic observability in polynomial time. *Journal of Symbolic Computation*, 33(5):735–755.