Lab Rotation Report
Topological Filtering

Simon Möller

April 8, 2015

Discriminating between various models of complex biological systems is a difficult task in computational biology. One approach in the study of chemical reaction networks is *topological filtering*. Starting from a model that incorporates all hypothetical reactions, this method finds a set of reduced models which are still in agreement with given measurement data. Reducing the root model corresponds to eliminating kinetic parameters that describe the involved reactions. Previously this elimination was done by setting the parameter to some small ε rather than 0. In this report we demonstrate that this can lead to qualitatively different results. We also analyze how quality and quantity of measurement data influence the results of the method for a small test network. Higher quality data leads to fewer viable models, whereas simply increasing the number of measurements does not have the same effect. Additionally, we provide the source code for the method as a MATLAB package and describe its usage.

1 Introduction

Computational models are an important tool to understand biological systems and to generate quantitative predictions that can be experimentally validated. Networks of chemical reactions are often modeled using ordinary differential equations. In such a model the temporal evolution of the state variables, representing the concentrations of the chemical species of interest, is determined by the network structure and reaction rates. These rates depend on kinetic parameters, which constitute parameters of the model. Namely, kinetic parameters can be assumed to be constant over time. However, values of these parameters are often unknown. In principle, measuring them precisely is inefficient (cf. [1]). Additionally, the structure of the network is also uncertain (i.e. it is unclear which reactions actually happen).

1.1 The topological filtering method

A common approach is to start with a simple, well-established core-model and then add possible hypotheses. In this study we are looking at a different approach called topological filtering that was introduced in [2]. Starting from a root model that incorporates *all* proposed reactions, the method tries to eliminate in an iterative fashion unnecessary model parameters, respectively reactions (model space exploration). This is done by an approximate verification of the model behavior for a wide range of values of its parameters (parameter space exploration).

We assume that some experimental data is given to which we can compare the output of the model for each set of parameters. Assuming that the errors between model and measurement data, ε, are normally distributed with mean 0, the likelihood of the observed data Y, given a model M and a parameter point θ is given by a

multivariate normal distribution

$$p(Y|\theta, M) = \frac{1}{\sqrt{(2\pi)^\rho |S|}} e^{-\frac{1}{2}\varepsilon^T S^{-1}\varepsilon}, \tag{1}$$

where ρ is the number of data points (i.e. number of observable species times number of time points) and S is the covariance matrix of these data points. To determine the viability of θ we check whether the minus log-likelihood function $E(\theta, Y, M) := -\ln[p(Y|\theta, M)]$ is smaller than the viability threshold:

$$E_0(\theta^*, M^*) := \ln(e^{\sqrt{2\rho}}\sqrt{(2\pi e)^\rho |S|}). \tag{2}$$

The threshold is derived as an expected value plus two standard deviations over all data points of the true parameter point and true model pair (θ^*, M^*). Derivation details of the parameter point viability criterion can be found in the supplementary material of [2]. Note that only ρ and S are needed to compute E_0.

We use the method described in [3] to explore the parameter space within given bounds of parameter values and obtain a representative sample of viable parameter space.

For each viable parameter point the parameters are checked for being nonessential one at a time, by setting their value to zero, or some small value ε, and by checking whether the 0 or ε-projected parameter point remains viable. For multiplicative parameters of reaction rates, such as mass action kinetics constants, projection of that parameter corresponds to an elimination of the respective reaction. For other parameters projection simplifies the respective reaction rate function. Subsequently, all parameters which are nonessential individually are projected simultaneously. Fig. 1 illustrates an example in which, for a given parameter point, parameters 1, 2 and 3 are nonessential when eliminated one at a time, whereas parameter 4 is essential. Hence, the models $M_{-\{1\}}$, $M_{-\{2\}}$, and $M_{-\{3\}}$ are viable (colored in green) and the algorithm moves on to checking their composition $M_{\{4\}}$ (i.e. only parameter 4 is not projected), skipping the second order models in between (colored in gray). If $M_{\{4\}}$ is found to be viable, it is saved as a possible reduced model and serves as a starting point for another round of topological filtering. If it is not viable for currently given parameter point, the method, originally, proceeds to the next parameter point. Moreover, since parameter 4 was found to be essential when eliminated individually, non of the models that eliminate 4 are explored for this parameter point (dashed boxes).

1.2 Motivation and results

The goal of this project is to highlight some of the shortcomings of the method itself as well as the previous implementation. One deviation between that implementation and the theoretical concept was that parameters were not projected to 0, but rather to some small value ε for numerical reasons. By using a simple, two-species network we will illustrate possible consequences of this. We also aim to provide an understanding of the method's uncertainty by analyzing a test network of four species. We analyze how the quality and quantity of measurements and parameter sampling influences the topological filtering outcome for the model.

In Section 4 we discuss further, in a more general manner, design choices for the method, with possible influences on its results.

Besides the theoretical work in this project, we also provide the implementation such that the method can be used as a MATLAB package. A description of the package and a short discussion of the runtime is given in Section 5.

2 Shortcomings of the parameter projection: a simple example

In the first case study we consider a very simple model that allows us to illustrate some special cases that can occur during the exploration of the model space.

Model description We consider two chemical species with concentration x_1 and x_2. The first species is produced and degraded with rates $u(t)$ and $k_1 x_1(t)$ respectively. It acts as a ligand that enhances the production

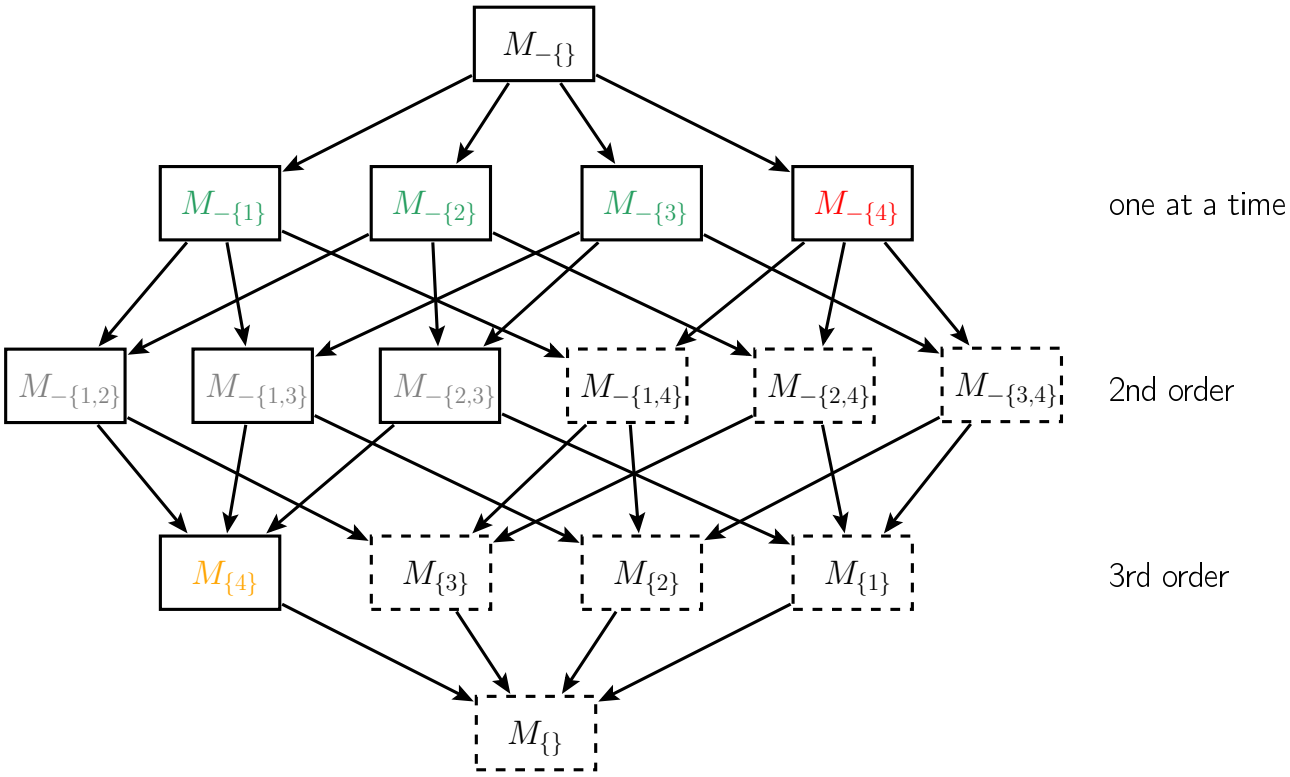


Figure 1: Illustration of the model space exploration. The subscripts denote which parameters have been deleted from (when preceded with a minus sign) or are present in the model. Green and red labels indicate viable and non-viable models, respectively. The yellow label represents an unchecked model. Gray labels and dashed boxes indicate models that are not being explored.

of the second species, a target protein, with rate $k_3 x_1(t)$. The second species is additionally produced at constant rate k_2 and degraded with rate $k_4 x_2(t)$:

$$\begin{aligned} \frac{dx_1(t)}{dt} &= -k_1 x_1(t) + u(t) \\ \frac{dx_2(t)}{dt} &= k_2 + k_3 x_1(t) - k_4 x_2(t), \end{aligned} \tag{3}$$

with initial conditions $x_1(0) = x_1^0, x_2(0) = x_2^0$. We consider here the special case with $u(t) = 0, x_1^0 > 0$, i.e. the ligand is only produced instantaneously at $t = 0$, and $x_2^0 = 0$. We get:

$$\begin{aligned} \frac{dx_1(t)}{dt} &= -k_1 x_1(t) \\ \frac{dx_2(t)}{dt} &= k_2 + k_3 x_1(t) - k_4 x_2(t). \end{aligned} \tag{4}$$

The steady state of this system is

$$\begin{aligned} x_1^* &= 0 \\ x_2^* &= k_2/k_4. \end{aligned} \tag{5}$$

On the other hand, if we consider the reduced model with $k_1 = 0$, i.e. with a non-degradable ligand, we find that

$$\begin{aligned} x_1(t) &= x_1^0 = \text{const.} > 0 \\ x_2^* &= \frac{k_3 x_1^0 + k_2}{k4}. \end{aligned} \tag{6}$$

Experimental setup We assume that we can experimentally observe only the concentration x_2 at one time point and denote this observable with y_1. Thus, in Eq. (2) we have $\rho = 1$. Defining $S = y_1 \delta$ and using a sample parameter point and measurement error from Table 1 we find that the viability criterion is $E_0 = 2.49$.

Table 1: Parameter values used for the simulation.

Name	Value
k_1	0.04
k_2	8
k_3	1
k_4	1
x_1^0	10
x_2^0	0
y_1	10
δ	0.05

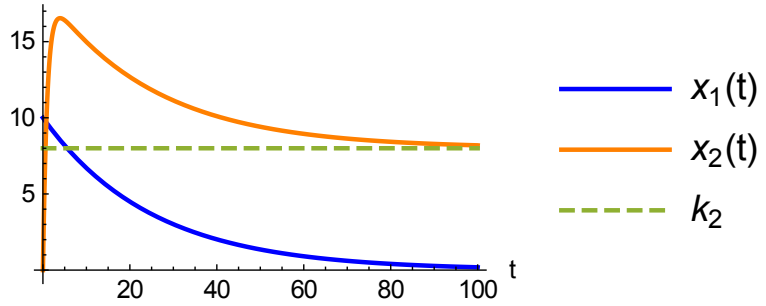


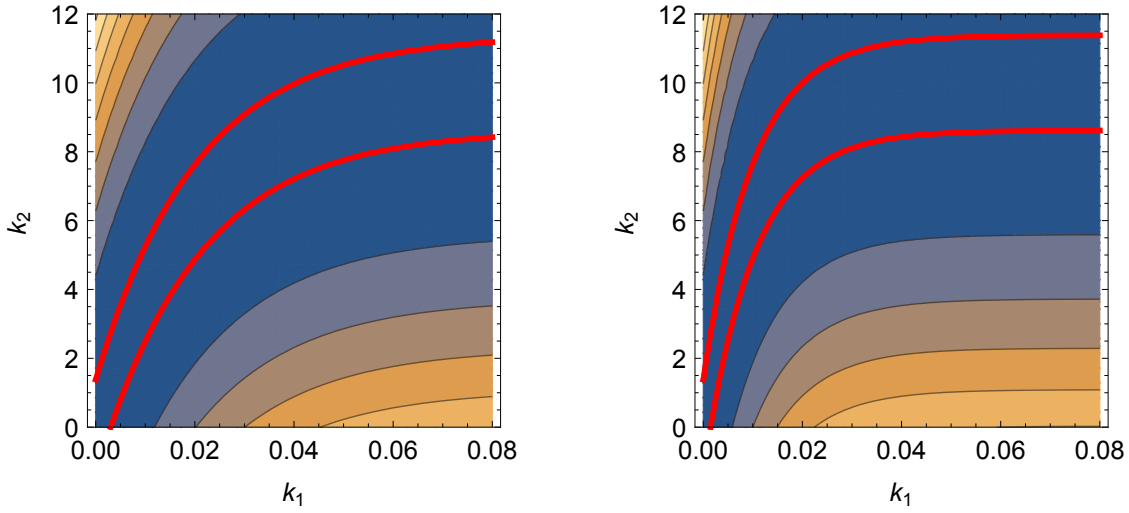
Figure 2: Time course of model 1 with the parameters given in Table 1. The concentration of the ligand, x_1, decreases exponentially, while the concentration of the protein, x_2, first increases steeply and then slowly approaches its steady state from above (k_2, shown as a dashed line).

Fig. 2 shows the time course of x_1 and x_2 for the parameters specified in Table 1.

Assume now that k_3 and k_4 are fixed at given values, and we only explore parameter space for k_1 and k_2. Fig. 3 depicts the viable parameter space at three different time points. The threshold contour is shown in red. As time progresses the horizontal stripe of viable space around $k_2 = y_1 = 10$ extends towards the k_2-axis. This is because we have $x_2^* = k_2$ for $k_1 > 0$ (Eq. (5)), i.e. $x_2(t)$ approaches the value of k_2 as t goes to infinity. For the reduced model with $k_1 = 0$ we have we have $x_2^* = x_1^0 + k_2 = y_1 + k_2$ (Eq. (6)), i.e. the origin is always a viable parameter point.

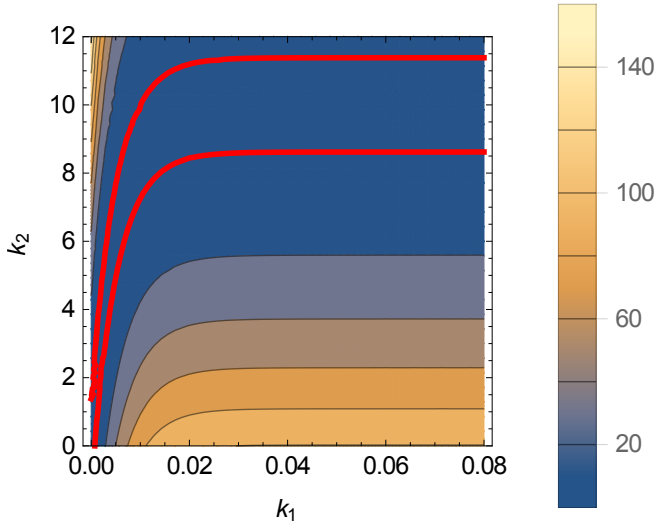
Observations Using above setup, Fig. 4 illustrates two cases which show how a projection to some small ε can lead to qualitatively very different results than a projection to 0. Fig. 4a illustrates one of the shortcomings of the previous implementation. Assume 0.1 as the lower bound of k_1 for parameter sampling. To check whether k_1 is essential, the viable parameter points are projected to some small value $k_1 = \varepsilon_1$ rather than to $k_1 = 0$. Fix $\varepsilon_1 = 0.01$, i.e. one order of magnitude smaller than the lower boundary. As the figure shows, the ε-projection of the viable parameter point still lies within the viable region, while a projection to $k_1 = 0$ would not thus giving rise to a false positive. For any ε_1 and any lower bound for k_1 one can fix a large enough measurement time point such that this case would occur.

Analogously, Fig. 4b shows (for different values of the sampling boundary and ε_1) how two false positive predictions of one at a time ε-projections can lead to a true positive prediction when combined. The one at a time projections of k_1 and k_2 to ε_1 and ε_2, respectively, are incorrectly classified as viable. A subsequent check of the simultaneous elimination of both parameters would, however, correctly classify the resulting model as viable. Using a projection to 0, this reduced model would not have been identified as the one at a time projections would not have been viable.



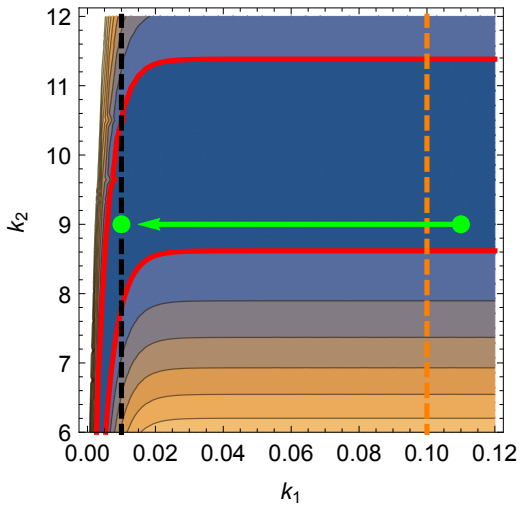
(a) $t = 50$

(b) $t = 100$

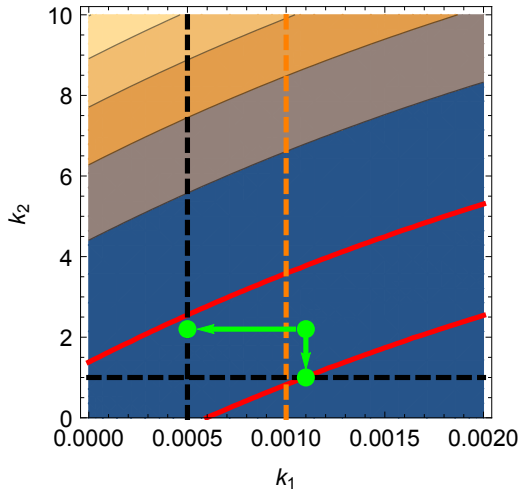


(c) $t = 200$

Figure 3: Contour plot of the cost function at three different time points using the parameters in Table 1. The contour for the threshold at $E_0 = 2.49$ is depicted in red.



(a) While the ε-projection still lies within the viable space, a projection to $k_1 = 0$ would not.



(b) The two ε-projections are incorrectly classified as viable thus predicting correctly that the combination of eliminating k_1 and k_2 is viable as well.

Figure 4: Contour plots of the cost function for $t = 250$ and a sampled parameter point in different scenarios. The projection to some ε is shown as a black, dashed line, the lower boundary for sampling of k_1 is depicted by the orange, dashed line.

The current implementation allows the user to specify for each parameter to which value it should be projected.

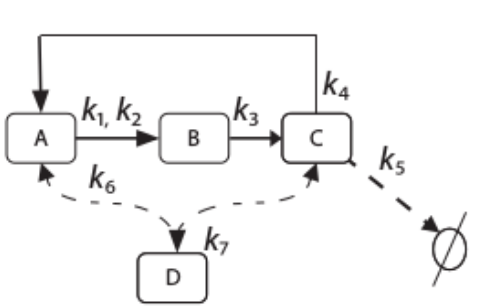
3 Uncertainty assessment in the method

Since the parameter exploration in the topological filtering method is random, different results can be produced for the same model and data set. The goal of this part of the experiment is to assess the reliability of the results. Additionally we want to find out how the set of viable models depends on the the quantity and quality of the measurement data.

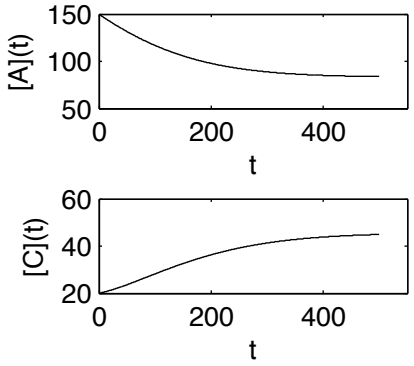
Model description We use a small test network that was described and analyzed in [2] (see Fig. 5a). In this network a substrate A is converted through an intermediate species B into a product C in a two-step process, with the first step being modeled using the two-parameter Michaelis-Menten kinetics rate law. The product can be converted back to substrate or degrade. A fourth species D is a complex of the substrate and the product, introducing reversible inhibitory mechanism for the substrate-product conversion cycle. Seven kinetic parameters are describing the reactions. The true model, used to generated data, excludes C degradation and D species reactions. The mathematical details of the model can be found in the supplementary material of [2].

This model illustrates one of the caveats that need to be kept in mind for the model space exploration: The parameter k_7, standing for D dissociation back into A and C, can not be eliminated individually, even though it was set to 0 to generate the data. It can only be removed in combination with the parameter of the reverse reaction k_6. This shows the necessity for "recursive search", i.e. using already reduced models to further explore model space. In this case parameter k_6 can be eliminated individually and subsequently we can also eliminate k_7 (in fact, the value of k_7 is of no importance once k_6 is eliminated since the concentration of D will remain 0).

Experimental setup To assess the influence of the quality and quantity of the measured data on the results we used synthetic datasets. We generated the datasets by simulating the model for the same parameters as in [4]. In that setup, $k_5 = k_6 = k_7 = 0$, i.e. degradation of C and the inhibition by D do not take place. We based our



(a) Example network with four chemical species and seven kinetic parameters. Dashed lines represent reactions that were eliminated when we generated the artificial data. The figure was taken from [2].



(b) Time course of the concentrations of species A and C for a simulation of the model using the parameters given in [4].

Figure 5

analysis on the work in [2] and considered species A and C as observable. The time course of the concentration for those two species is shown in Fig. 5b.

In the first part of the experiment we took the results at 20 uniformly distributed time points between 25 and 500 seconds. We then set the standard deviation for species $x \in A, C$ at each measurement point t to

$$\sigma_x(t) = \delta x(t) + 0.02 x_{\max}, \tag{7}$$

where δ is the relative standard deviation, $x(t)$ is the concentration of either species A or C at time t and $x_{\max}$ is the maximum concentration of the respective species over the entire course of the simulation. We generated datasets for $\delta \in \{0.05, 0.1, 0.15\}$. For the second part of the experiment, we fixed $\delta = 0.1$ and varied the number of time points to 10, 20, and 30, again creating three datasets.

For all six datasets, we performed ten runs of topological filtering with $S = \operatorname{diag}\big(\sigma_x(t_1), \ldots, \sigma_x(t_p)\big)$. We counted how often each of the reduced models was found to be viable. The parameters for the parameter space exploration were fixed to `nmont` = 100 and `nelip` = 30. We explored all parameters in the range $[10^{-4}, 10^4]$. Parameter k_2 corresponds to a Michaelis-Menten constant and was projected to $\varepsilon = 10^{-5}$, while all other parameters were projected to 0.

We also investigated how the parameter space sampling quality affects the outcome of the method. For $\delta = 0.1$ and 20 time points, we chose `nmont` $\in \{10^2, 10^3, 10^4\}$ always setting `nelip` = `nmont`/10, performing 10 runs of topological filtering.

Reproduction of previous results This test network has been analyzed using the topological filtering method before for data with 20 time points [2]. Seven reduced viable models were reported. We discovered additional models in our experiments. The discussion on this model in [2] was, however, very brief. In particular, it was not stated explicitly how the noise for the artificial data was generated which can explain the deviation between the findings. Also, the topological filtering method is not exact so varying outcomes can occur.

Observations Changing the standard deviation results in a change of the covariance matrix S which in turn affects the viability threshold E_0 (cf. Eq. (2)) as well as the likelihood function p (cf. Eq. (1)). For the three different relative standard deviations δ, we find $E_0 = 126.62$, 146.65, and 159.92, respectively. Therefore it is expected that more models will be viable for larger values of δ which is indeed the case as illustrated in Fig. 6. Note that positive counts below 10 indicate that the respective model was only found to be viable in some of the runs. Better quality of measurements, leads to a smaller viable space for the reduced models and thus the

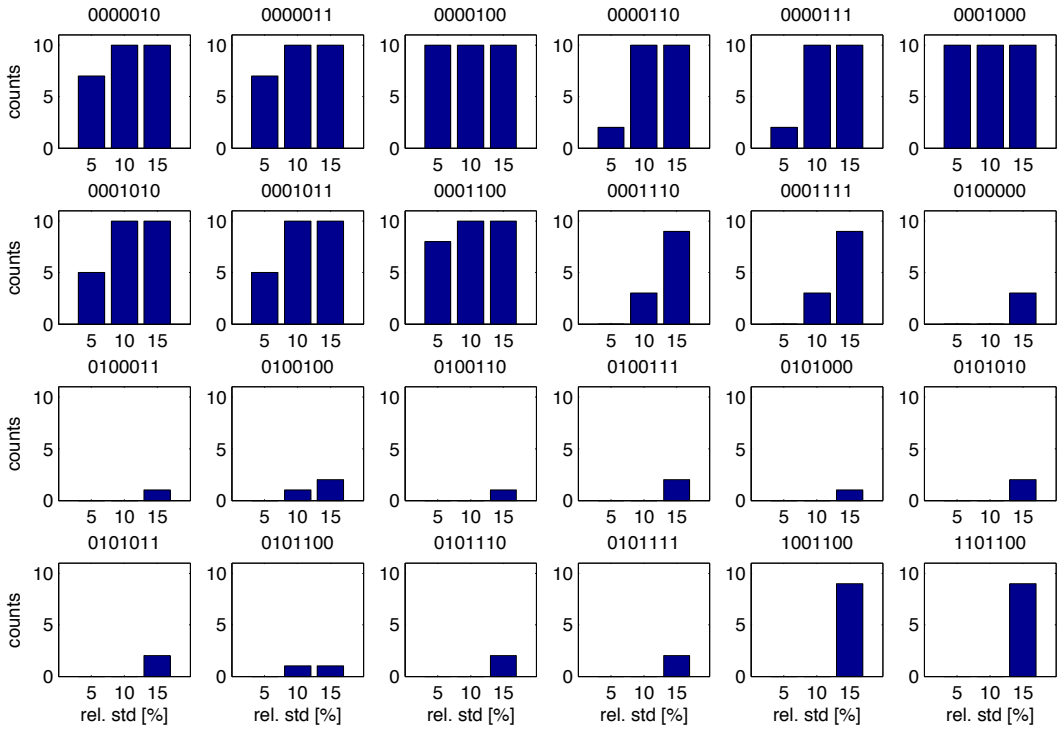


Figure 6: For three different values of the relative standard deviation, δ, we performed ten runs of topological filtering and counted how often each reduced model was discovered as viable. The title string of each subplot encodes the reduced model as a binary string with a 1 in the ith position indicating that the ith parameter has been eliminated.

likelihood of sampling a parameter in that space decreases (cf. Fig. 6, models `0001110` and `0001111`, or models `1001100` and `1101100`). Model `1101100` corresponds to a chemical reaction network that differs substantially from the one that was used to generate the data. Fig. 7 shows, however, that the resulting time course of concentrations does resemble the original data under high enough measurement uncertainty.

Fig. 8 shows the results for varying the number of time points. Again, increasing the number increases the threshold E_0, in this case to 75.07, 146.65, and 217.62, respectively. The results are, however, different from the previous case. For each model, the number of times it is found viable is similar for all obtained values of E_0. The changes in the shape of the viable parameter space due to additional time points appear to offset the increase in the threshold. We do clearly see that some models are found with far greater reliability than others. Increasing the number of sampled parameter points as well as repeated sampling during the recursive search would probably reduce this difference.

It should be noted that in our experiment the measurements were *exact* and we simply fixed the standard deviation to a certain value. In a real experiment, the measurement will be noisy and in such a situation adding more measurements will most likely affect the outcome of the method.

The results for the experiment in which we varied the number of sampled parameter points are shown in Fig. 9. Exploring the viable parameter space in more detail increases the chances to find viable reduced models which occupy a small viable subspace of the sampled parameter space. Examples for such models are `0001110` and `0101111` that were only discovered for `nmont` $= 10^4$. Other models (such as `0100000` in which only k_2 has been reduced) are found in every run for all values of `nmont`. In general, increasing the number of runs or the number of explored parameters increases the chances of finding additional reduced models that are viable at the cost of higher computation time (compare section 5.2).

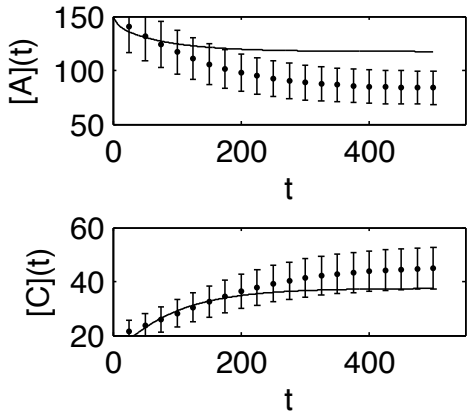


Figure 7: Example time course for the reduced model `1101100` (i.e. parameters k_1, k_2, k_4, and k_5 have been eliminated) together with artificial measurement for $\delta = 0.15$ for which this model was found to be viable. The remaining parameters were set to $k_3 = 0.0117, k_4 = 0.0004$, and $k_7 = 0.0599$ resulting in a cost of $E = 159.74$ barely under the threshold $E_0 = 159.92$.

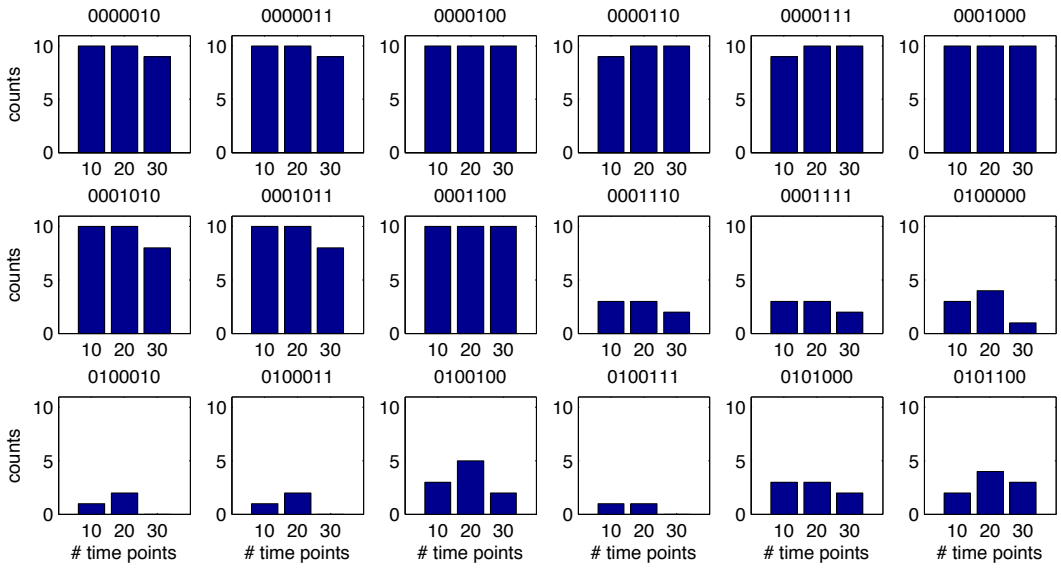


Figure 8: Similar to Fig. 6, but this time we varied the number of time points to three different values.

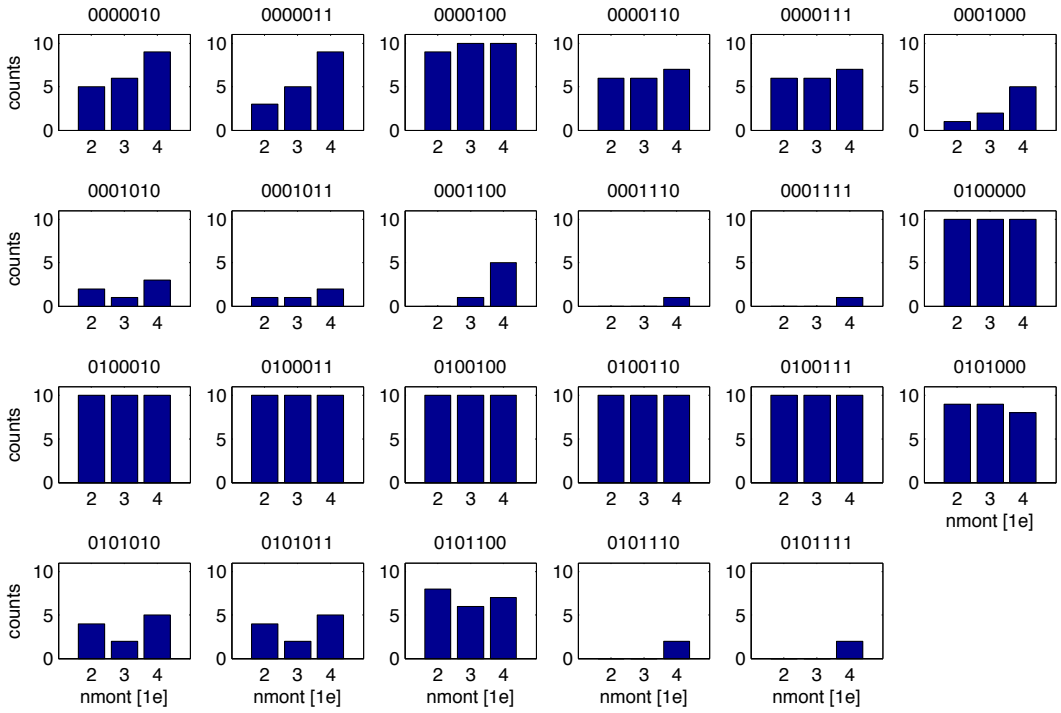


Figure 9: Similar to Fig. 6, but this time we varied the number of sampled parameters in the Monte Carlo exploration, `nmont`, to $10^2, 10^3$, and 10^4.

4 Further considerations for the method

The topological filtering method is a heuristic tool to find viable models without exploring the entire model space. Such approaches always seek to find a balance between computational burden an accuracy. For topological filtering there are several design choices that may influence the result.

4.1 Backtracking and recursive search

Fig. 1 illustrates the entire model space for a root model with four parameters. Given a set of viable parameter points, a subset of this space is explored for each of them. In the example, models $M_{-\{1\}}$, $M_{-\{2\}}$, and $M_{-\{3\}}$ are viable for at least some parameter points. For those points, the algorithm checks the composed model $M_{\{4\}}$. If that model is found to be nonviable for some parameter point, we have the option to perform *backtracking*: This checks the models that are composed of a subset of the reduced parameters (i.e. $M_{-\{1,2\}}$, $M_{-\{2,3\}}$, and $M_{-\{1,3\}}$). This can be done for each parameter point as soon as $M_{\{4\}}$ is found nonviable, or it can be seen as a last resort, once $M_{\{4\}}$ has been checked for *all* parameter points.

Conversely, if $M_{\{4\}}$ is found viable for some parameter point, we have the option of using it as a starting point for *recursive search*: In this procedure, $M_{\{4\}}$ becomes the starting point for another iteration of topological filtering. In the example, the only model left to check is the fully reduced model $M_{\{\}}$. Exploration of the model space from the root model had found that the fourth parameter was essential when eliminated individually and therefore did not explore $M_{\{\}}$. However, now that the other three parameters have already been eliminated, parameter four may not be essential any longer (just like in the ABCD example of the previous section in which k_7 could only be eliminated together with k_6).

4.2 Sampling frequency

Another important decision concerns the question how often the viable parameter space should be sampled for points from which then reduced models are explored. In particular one can sample only once using the root model, or every time topological filtering is started again during the recursive search, or even for each reduced model (when considering all parameter points at once). In particular, new parameter points could also be sampled during backtracking.

5 Code

5.1 MATLAB package

The code for carrying out the topological filtering method is contained in a MATLAB package. It requires the HYPERSPACE [3] and the SBTOOLBOX2 [5] package to run properly.

For the most part, this package performs the same analysis as done in [2]. In fact, we generalized the code that was used in that study. Also, the option for backtracking was added and recursive search can be carried out automatically.

5.1.1 Quickstart

To analyze your own model you need to create the following files:

1. A description of your model following the SBTOOLBOX2 specifications.
2. A description of the experimental data in a txt file that includes (always separated by whitespace)
   a) the time points of the measurements in the first line
   b) the names of the observed species in the second line
   c) the average measurements at each time point, for each species in a new line
   d) the measurement error as either
      i. two numbers: the relative error and relative base error (e.g. 0.1 0.02, `modeString='relative'`)
      ii. the standard deviation of each measurement, again as one line per species (`modeString='stds'`)
3. A description of the model parameters in a txt file in which every line looks like:

```
<parameter name> <projection value>
        <lower bound for exploration> <upper bound for exploration>
```

   e.g.

```
k1 0        0.001 100
k2 0.0001 0.001 10
```

Copy `setupTemplate.m` into the folder containing your files and adjust it accordingly. Additional parameters that can be adjusted in the beginning of that file and their meaning are listed in Table 2. After execution is finished, the workspace variables will be saved in the file specified in `fname`. One of those variables is the matrix `allViableProjections` which contains in its rows all viable projections that were found as boolean vectors (`true` if the parameter was reduced). Additionally, the vector `viableProjectionCounts` contains in the ith row the number of runs in which the ith projection in `allViableProjections` was found viable.

Table 2: Parameters of the package that can be adjusted.

Name	Meaning
fname	filename for results (i.e. workspace variables will be saved there)
numRuns	number of runs that should be performed
nmont	maximum number of integrations in Monte Carlo sampling (cf.[3])
nelip	maximum number of integrations in ellipsoids sampling (cf.[3])
backtrack	flag that determines, whether backtracking should be performed
recursive	flag that determines, whether the reduced models should be used as a starting point for another run of topological filtering

5.1.2 Passing additional arguments

For some experiments it may be convenient to have more control over the code. The program therefore computes values for the following variables itself only if they have not been set externally before:

- `p0`: This is the starting point used for the exploration of parameter space. If not provided `simplexSB` optimization from the SBTOOLBOX2 package is used to find it.
- `expData`: This is a MATLAB structure that needs to contain the following fields:
  - `timePoints`: column vector containing the time points at which measurements were taken
  - `observables`: column cell array containing the names of the observable species as strings
  - `measurements`: a matrix in which each row represents the measurements for one species at each time point

  Additionally it needs to contain either
  - `S`: a $\rho \times \rho$ covariance matrix made up of k (number of time points) blocks of size $l \times l$ (l is number of species) that are placed along the diagonal. Each such block contains the covariance between the measurements for all species at that time point.

  or
  - `logDetS, invS`: these two values represent $\ln|S|$ and S^{-1}, respectively. The first one is a scalar, the last one a matrix.
- `logDetS, invS`: As in the description for `expData`; these parameters can also be supplied by themselves.

5.1.3 Overwriting functions

All source code is contained in the `source` directory of the package. If you wish to overwrite specific function calls (e.g. to adjust the `evalModel` function to fit your experiment specifically) make sure they have higher precedence than the one in the `source` directory.

5.2 Computation time

We recorded the runtime for the experiment described in Section 3 for the three different values of `nmont`. We find that the runtime scales exponentially with the logarithm of `nmont` (Fig. 10).

For a more realistic test, we repeated the analysis of a model described in [2] that analyses 17 parameters in a network of 30 reactions. Performing one run of topological filtering with backtracking disabled and recursive search enabled and using `nmont` $= 10^4$ and `nelip` $= 10^3$ took 30 hours on ETH's Euler cluster (Intel Xeon E5, 2.7 GHz). In this setup of sampling method we were able to replicated results reported in [2], i.e. parameters of

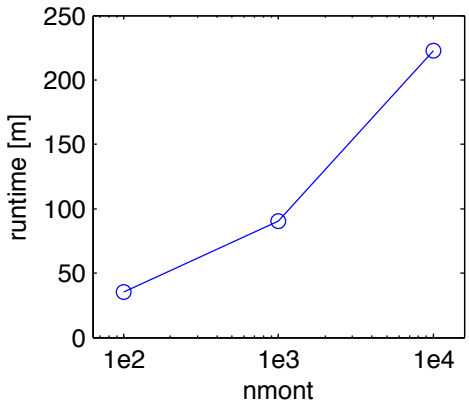


Figure 10: Runtime for the ABCD model for varying number of sampled parameters running MATLAB on a Linux Desktop PC with an Intel i5, 3.4 GHz.

reactions r_1-r_5 were found to be always essential; namely, parameters $V_{\max}$ for all 5 reactions and additionally K_{M} for r_1, r_3, and r_5.

6 Discussion

This report provides further insights into the topological filtering method. We have shown using a small example network how the projection value for a parameter can influence the resulting viability. For another small network we have demonstrated that the quality of available measurement data impacts the outcome of the method, while the number of measurements did not have such an impact, despite viability threshold varying more. This illustrates the interplay between the cost function and the viability threshold. Increasing the number of sampled parameters was proven to increase the number of viable models that are discovered. We pointed out further design choices that may affect the outcome of the model such as backtracking from nonviable and recursive search from viable models. Finally, we described how our implementation of the method in a MATLAB package can be used and reported the runtime for some examples.

In all experiments, we have sampled the parameter space only once for the initial root model. While this greatly reduces computational burden in comparison to sampling the space in every iteration of the recursive search, further investigation should clarify how far this limits the reduced models that can be explored. Also, future studies could help to understand how exactly the shape of the viable parameter space (i.e. the cost function and the threshold) is influenced by changing quality and quantity of measurement data and how this, in turn, affects the efficiency of parameter sampling.

The provided implementation should allow others to perform topological filtering for their model, but there is still room for further improvements. For instance, by using MATLAB's symbolic toolbox reaction rates could be simplified after the elimination of a parameter. For example, this would allow to project Michealis-Menten constant to 0 without risking numerical errors, because the reaction rate $r = v_{\max}[S]/(K_{\mathrm{M}} + [S])$ would simply reduce to $r = v_{\max}$ for $K_{\mathrm{M}} = 0$. In the current implementation a projection to 0 may cause numerical problems for small concentration of S because the fraction is not simplified. Also, support for parallel computation has the potential to greatly reduce the runtime of the program. Exploration of the parameter space, the recursive search and the iteration over all viable points are steps that could be done in parallel.

References

[1] Ryan N Gutenkunst, Joshua J Waterfall, Fergal P Casey, Kevin S Brown, Christopher R Myers, and James P Sethna. Universally sloppy parameter sensitivities in systems biology models. *PLoS Comput Biol*, 3(10):e189,

10 2007.

[2] Mikael Sunnåker, Elias Zamora-Sillero, Reinhard Dechant, Christina Ludwig, Alberto Giovanni Busetto, Andreas Wagner, and Joerg Stelling. Automatic generation of predictive dynamic models reveals nuclear phosphorylation as the key msn2 control mechanism. *Science Signaling*, 6(277):ra41–ra41, 2013.

[3] Elias Zamora-Sillero, Marc Hafner, Ariane Ibig, Joerg Stelling, and Andreas Wagner. Efficient characterization of high-dimensional parameter spaces for systems biology. *BMC systems biology*, 5(1):142, 2011.

[4] Hans-Michael Kaltenbach, Sotiris Dimopoulos, and Jörg Stelling. Systems analysis of cellular networks under uncertainty. *FEBS Letters*, 583(24):3923–3930, 2009.

[5] SBTOOLBOX2 Web page. `http://www.sbtoolbox2.org/main.php?display=documentationSBT&menu=overview`. Last accessed 08-April-2014.