Contents

Configuration files

April 1, 2020

In order to use the tools provided in the LFNS toolbox, several configuration files containing information about the model, data and algorithm specifications need to be defined. In the following we give a detailed description about the syntax of these files.

Part I
Model

The models used by the LFNS toolbox are model of chemical reaction networks. The next two subsections are copied from the PhD thesis

Jan Mikelson. *Nested Sampling for biochemical reaction networks with applications to ERK signalling: A likelihood-free approach.* PhD thesis, ETH Zurich, 2019

1 Chemical Reaction Networks

The dynamics within biological cells are determined by the interaction between different proteins, metabolites and other biomolecules. For each considered model, the involved U biomolecules (or species) are denoted with $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_U$. The count of species $\boldsymbol{X}_u$ at a time t will be denoted with $X_u(t)$ and the full vector of the counts of all species with $X(t) = \{X_u(t)\}_{u=1,\ldots,U}$. These species interact through R reactions $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_R$ written as

$$\mathcal{R}_r : \qquad \sum_u^U p_{ru} \boldsymbol{X}_u \rightarrow \sum_u^U q_{ru} \boldsymbol{X}_u, \tag{1}$$

where p_{ru} is the numbers of molecules of species $\boldsymbol{X}_u$ consumed in reaction r, and q_{ru} is the number of molecules of species $\boldsymbol{X}_u$ produced by that reaction. These reactions usually represent biological processes within a cell, such as phosphorylation, degradation or translocation. These reactions fire according to propensities $\lambda_1(X(t)), \ldots, \lambda_R(X(t))$ that depend on the current state of the system $X(t)$ and a d-dimensional parameter vector θ.

Each reaction $\mathcal{R}_r$ is fully defined by their propensity $\lambda_r(X(t))$ and the corresponding stoichiometry vector ν_r, where ν_r indicates how many molecules of each species are

consumed and produced at each reaction

$$\nu_r = \begin{pmatrix} q_{r1} - p_{r1} \\ \vdots \\ q_{rU} - p_{rU} \end{pmatrix}.$$

The propensities have an intuitive interpretation, as they represent the probability at which each reaction $\mathcal{R}_r$ happens. The probability that a reaction $\mathcal{R}_r$ occurs in the time interval $[t, t+h]$ for some infinitesimal h is given by

$$\mathbb{P}\left(\text{Reaction } \mathcal{R}_r \text{ occurs in time interval}[t, t+h]\right) = \lambda_r\left(X(t)\right) h.$$

1.1 Propensities

In general, these propensity functions can take any arbitrary form[1]. In the following, we mention three frequently used forms of propensity functions for a state vector X.

Mass action kinetics These reactions describe the most simple interactions in a chemical reaction network. In this case, the propensity is directly proportional to the product of the concentrations of the involved species.

$$\lambda_r\left(X\right) = k_r \prod_{u=1}^{U} \begin{pmatrix} X_u \\ p_{ru} \end{pmatrix},$$

where k_r is some rate constant.

Hill kinetics Taking into account effects like ligand saturation, multiple binding sites, and general nonlinearity, hill kinetics allow for biological plausibility while avoiding mass action kinetics.

$$\lambda_r\left(X\right) = k_r \frac{X_{i_r}^{n_r}}{K_r + X_{i_r}^{n_r}},$$

for some index i_r, a rate k_r, the hill coefficient n_r and some constant K_r.

Michaelis-Menten kinetics Usually used for enzyme kinetics, Michaelis-Menten kinetics are a special case of Hill kinetics with $n_r = 1$.

$$\lambda_r\left(X\right) = k_r \frac{X_{i_r}}{K_r + X_{i_r}}.$$

The parameters such as k_r, K_r or n_r are usually encoded within the parameter vector θ.

[1] As long as they satisfy some basic growth conditions, as being non-negative or being zero whenever any involved species with a negative stoichiometry is zero.

1.2 Measurement model

In the context of systems biology, it is usually impossible to observe the involved species $\boldsymbol{X}$ directly. Instead, one must rely on noisy readouts from the considered system, such as fluorescent measurements or Western blot readouts. Modelling accounts for this by setting the measurement $Y(t)$ to be a P-dimensional random variable depending on the current state of the system $X(t)$ (also referred to as the latent state)

$$Y(t) \sim p(\cdot|X(t), \theta),$$

where p is some probability distribution. Note that in this formulation we allow for the measurement $Y(t)$ to also depend on the model parameters θ.

We assume that the variable Y is not observed at all times but only on T time points $t_1, \ldots, t_T$. For time point t_τ we also write $Y_\tau = Y(t_\tau)$ (and analogously $X_\tau = X(t_\tau)$ for the latent states). We denote with $Y = \{Y_\tau\}_{\tau=1,\ldots,T}$ all observations at all time points.

2 The model files

The LFNS toolbox reads the particular definition of the above described models from several text files that are being parsed by the toolbox. Three files are required for the full definition of a model:

The dynamics file This file contains the definition of the reactions in the model

The initial conditions file This file contains information about the initial states used for the simulation of the model.

The measurement model file This file contains informatin about the taken measurement and the definition of the likelihood function for each measurement.

2.1 General remarks about the model files

The model files are plain .txt files that are parsed by the LFNS toolbox (in particular by the file ParserReader.cpp in the namespace io). There are several implemented keywords available. The keywords are always written at the beginning of a line and are followed by a ":" and a line break. The keywords can also be found in the file ParserReader.h.

Species: After this keyword a list of the involved species follows, either separated by "," or " ".

Parameters: After this keyword a list of the involved parameters follows, either separated by "," or " ".

Random numbers: This keyword allows to define random variables that are used within the same .txt file. After this keyword the random numbers can be defined by writing the random number name, then a "=" and then the desired distribution for that random number. The currently available distributions are:

Normal(μ, σ) Creates a normal random number with mean μ and variance σ^2. Note that μ and σ need to be numeric values and not parameters!

Uniform(a, b) Creates a uniform random number between a and b. Note that a and b need to be numeric values and not parameters!

UniformInt(a, b) Creates an integer uniform random number between a and b. Note that a and b need to be numeric values and not parameters!

2.2 The dynamics file

The model dynamics file must contain the keyword "Species:" followed by a list of involved species in the next line (separated by "," or " "), the keyword "Parameters:" followed by a list of involved parameters in the next line (separated by "," or " ") and a list of the model reactions. The model reactions follow after the "Reachtions:" keyword and each reaction needs to be written in its own line. Each reaction consists of three parts

1. The first part defines the stoichiometry. It consists of the production species, followed by a "->" and a product species. So a reaction that converts the species A into the species B would look like

   ```
   A -> B
   ```

   If multiple species are involved (for instance in katalytic reaction) these species are combined using a "+". A catalytic reaction where the species A acts as a catalyst to convert specie B to A would be written as

   ```
   A + B-> A + A
   ```

   If a species gets created ex-nihilo or gets degraded the symbol "0" (zero) can be used, for instance

   ```
   0-> A
   ```

   would encode a reaction where the species A gets created ex-nihilo.

2. The second part needs to contain the keyword "Variables:" followed by a list of parameters associated with this reaction.

3. The third part contains the keyword "Propensity:" followed by a mathematical expression of the propensity including the involved species and parameters. For the definition of the propensity all parameters and species defined in the dynamics file under the "Parameters:" and "Species:" keyword can be used. For a Hill type propensity the full line could look like this

   ```
   A -> B          Variables:k,K          Propensity:k * A / (K + A)
   ```

Note that for the propensities the standard math notation can be used including symbols like "+", "-", "^", "/", "log", "log10", "sqrt", "exp", "_ pi" (containing the constant π), "binom" (the binomial coefficient with n and k), "ceil" and "floor". The list of all supported math operations can be found in the muParser description here and additionally defined functions in the ParserBaseObject.cpp file in the function "_ initializeParser".

Alternatively one can also write $\#ma$ after the "Propensity:" keyword to automatically use mass action kinetics. For example:

```
A -> B          Variables:k          Propensity:#ma
```

In this case the propensity will be parsed as "A*k". When using the $\#ma$ keyword the variable in the "Variables:" keyword will be used for the automated propensity generation!

A full dynamics file for simple gene expression could look like this:

```
Parameters:
k, gamma, k_P, gamma_P

Species:
mRNA, P

Reactions:
0 -> mRNA          Variables:k          Propensities:#ma
mRNA -> 0          Variables:gamma      Propensities:#ma
mRNA -> mRNA + P   Variables:k_P        Propensities:k_P*mRNA
P -> 0             Variables:gamma_P    Propensities:#ma
```

2.3 The initial conditions file

This file contains the initial conditions for the simulation of the model. It usually contains the "Parameters:" keyword, defining the involved parameters for the initial conditions (separated by "," or " "), the "Random numbers:" keyword, and the "Initial Values:" keyword. After the "Initial Values:" keyword, each line contains a species name, followed by a ":" and the corresponding initial value. An example of such an initial conditions file for the gene expression example is

```
Parameters:
mRNA_mu

Random numbers:
r_1 = Normal(0, 1)

Initial Values:
mRNA: mRNA_mu + r_1*0.1
P: 0
```

This file would create normally distributed initial mRNA counds, with a mean read from the parameter "mRNA_ mu" and standard deviation of 0.1, and zero initial protein.

2.4 The measurement model file

This file contains all the information about the simulation of the measurement for the model. The file contains a "Parameters:" keyword with the involved parameters, a "Species:" as well as "Random numbers:" keyword. Additionally it also contains the keyword "Measurement:" and the keyword "Loglikelihood:". These last two keywords define the measurement as well as the formula for the log-likelihood computation.

2.4.1 Measurement:

Any given number of measurements can be encoded, corresponding to various real-life experimental measurements such as fluorescent read-outs of the different involved species. The measurement keyword is followed by the formulas for each measurement, where each measurement needs to be written in its own line. The first entry in each line is a name for that measurement. This name can be freely chosen, but each measurement needs to have a corresponding line in after the "Loglikelihood:" keyword. After the measurement name a "=" follows and then the formula for the measurement, containing the species defined in the "Species:" keyword and parameters in after the "Parameters:" keyword.

2.4.2 Loglikelihood:

Each measurement needs to have an associated line after the "Loglikelihood:" keyword that contains a formula on how to compute the log-likelihood for this measurement. Currently it is assumed that the measurements are independent so that the total likelihood is just the product of the likelihoods for each measurement. Each line after the "Loglikelihood:" keyword begins with the name of the measurement, followed by a ":" and then the formula for the log-likelihood. In this formula the name of the measurement can be used as a variable. Here is an example of a full measurement model file for the gene expression model:

```
Parameters:
mRNA_scale, P_scale

Random numbers:
r_mRNA = Normal(0, 1)
r_P = Normal(0, 1)

Measurement:
mRNA_read  = mRNA_scale * mRNA + r_mRNA
P_read = P_scale * P + r_P

Loglikelihood:
mRNA_read: - (mRNA_read -  mRNA_scale * mRNA)^2 / (2) - log(sqrt(2 * _pi))
P_read: - (P_read -  P_scale * P)^2 / (2) - log(sqrt(2 * _pi))
```

In this case two measurements are simulated, one for the mRNA and one for the protein. Each measurement is assumed to be a normal random number centered around the scaled means of the corresponding species and with variance equal to 1. The log-likelihood functions are thus just the log-pdf of normal distributions.

Part II
The config.xml file

Once the model is defined as described above, an additional .xml file (we will refer to it as the config.xml file) needs to be created containing all the information about the model files. The config.xml file is an xml file with usual xml syntax. In the following we will describe the different blocks of this file and indicate which of these are minimally required for simulation, for likelihood computation and for running the full LFNS-inference. All these options are read through the class ConfigFileInterpreter.cpp.

3 Basic blocks

3.1 The <model> block

The <model> block is indicated with the "model" keyword and contains the information about the model type, and the location of the model files. In particular it contains the following entries:

<type> **(required)** This entry indicates how the model needs to be simulated. The options are DET (determinsitic ODE models), STOCH (stochastic models) and HYBRID (hybrid models). These options can be found in the file ModelSettings.h. If DET is chosen, an ODE simulator is used for simulation, if STOCH is chosen

the model gets simulated by the SSA and if HYBRID is chosen, some reactions will be simulated stochastically and some will be solved using an ODE simulator. To specify which species should be simulated stochastically, an additional field "stochspecies" (or "detspecies") needs to be provided to indicate the corresponding species. The remaining specie will automatically be considered deterministic (or stochastic respectively). Note that currently the hybrid model will only be simulated correctly if the stocahstic species do not depend on the determinsitic one (the reason for this is that in this case the stochastic simulator would need to integrate the ODE for the determinsitic cases to compute the stochastic reaction propensity and this has not been implemented yet).

`<model>` **(required)** This is the relative path to the model dynamics file.

`<initialvalue>` **(required)** This is the relative path to the initial states file.

`<measurement>` **(required)** This is the relative path to the measurement model file.

A full model block could look as following:

```
<model>
   <type>DET</type>
   <model>./gene_expression_dynamics.txt</model>
   <initialvalue>./gene_expression_initial_value.txt</initialvalue>
   <measurement>./gene_expression_measurement.txt</measurement>
</model>
```

Another example, where the model is simulated with a hybrid simulator where the species mRNA is simulated stochastically would look like this:

```
<model>
   <type>HYBRID</type>
   <model>./gene_expression_dynamics.txt</model>
   <initialvalue>./gene_expression_initial_value.txt</initialvalue>
   <measurement>./gene_expression_measurement.txt</measurement>
   <stochspecies>mRNA</stochspecies>
</model>
```

3.2 The `<Simulation>` block

The simulation block contains the information needed to simulate the defined model. It contains several options, most of which do not necessary need to be provided and in the case where they are not provided default values will be picked.

`<parameter>` **(required for simulation)** This entry contains a list of the particular values for the simulated parameters. It needs exactly as many entries as the model has parameters (in the dynamics file, the initial value file and measurement model

file), separated by "," or " ". The order of entries usually corresponds to the the order of parameters in the dynamics file, the initial value file and the measurement model file. However some options might change this order and one should always check the produced model summary file to verify that the parameters are assigned in the right order. Parameters that have been fixed through the `<fixedvalues>` (5) keyword do not need to be provided. This provided parameters can always we overwritten with the command line option `-p`.

`<experiments>` Provides a list of experiments to be simulated separated by "," or " ". This is useful if there are several experiments defined in the `<inputs>` fields and only a selected number of experiments needs to be simulated. If this field is not defined only one experiment without inputs will be performed and given the dummy name "0". This provided experiments can be overwritten with the command line option `-E`.

`<num_simulations>` This entry provides the number of simulations performed. It can be overwritten with the command line option `-n` and is set to 1 as default.

`<parameter_file>` Instead of one parameter in `<parameter>` or through the command line `-p`, the user can also provide a file with a list of parameters to be simulated. This file needs to have each parameter vector in a new row and each row should have as many entries as parameters are needed. The values can be separated by "," or " ". This entry can be overwritten with the command line option `-P`.

`<initialtime>` This provides the initial time for the simulation. A default value of 0 is assumed if this entry is not provided. This value can be overwritten with the command line options `-I`.

`<finaltime>` This provides the final time for the simulation. A default value of 100 is assumed if this entry is not provided. This value can be overwritten with the command line options `-F`.

`<interval>` This provides the read-out interval for the simulation (i.e. after what time the next output of the simulation should be written). A default value of 1 is assumed if this entry is not provided. This value can be overwritten with the command line options `-i`.

A first minimal example of the `<Simulation>` block for the above used gene expression model (not that it has a total of 7 parameters: `k, gamma, k_P, gamma_P` for the dynamics file, `mRNA_mu` for the initial conditions file and `mRNA_scale, P_scale` for the measurement model file) could look as follows

```
<Simulation>
    <parameter>1 0.1 1 0.1 10 1 1</parameter>
</Simulation>
```

Another example for the same model could look like this

```
<Simulation>
    <parameter>1 0.1 1 0.1 10 1 1</parameter>
    <initialtime>0</initialtime>
    <finaltime>250</finaltime>
    <interval>15</interval>
</Simulation>
```

3.3 The <LFNS> block

This block contains all the information needed to run the LFNS algorithm. All the default values can be found in the file LFNSSettings.h in the lfns namespace.

`<experiments>` **(required)** Provides a list of experiments to be used for the parameter inference separated by "," or " ". Each provided experiment must have a `<dataset>` (4.1) associated with it. In general each experiment will also have its own `<input>`.

`<N>` Provides the total number of LFNS particles. A default value of 1000 is assumed if this entry is not provided. This value can be overwritten with the command line options `-N`.

`<r>` Provides the number of LFNS particles resampled in each iteration. A default value of 100 is assumed if this entry is not provided. This value can be overwritten with the command line options `-r`.

`<H>` Provides the number of particle filter particles for the LFNS algorithm. A default value of 200 is assumed if this entry is not provided. This value can be overwritten with the command line options `-H`. Note that `H` should only be chose to be larger than 1 if the model needs a likelihood approximation (i.e. if the model is stochastic or has at least random initial conditions). If the model is purely deterministic H=1 is enough to compute the full likelihood.

`<dpgmmiterations>` Provides the number of iterations for the DP-GMM sampler for the LFNS algorithm. A default value of 50 is assumed if this entry is not provided. This value can be overwritten with the command line options `-d`.

`<epsilon>` Provides the tolerance for the LFNS algorithm. This is the value used for the Δ_{LFNS} termination criterion as defined in Mikelson and Khammash [2019]. A default value of 0.01 is assumed if this entry is not provided. This value can be overwritten with the command line options `-t`.

`<providedparameters>` This is an advanced option and is used for the case where a distribution for some of the model parameters already has been found and samples from this distribution have been written in a file (say `provided_params.txt`, each row needs to correspond to one parameter sample). In this case this file can be used

to infer the remaining parameters conditioned on the already provided parameter posterior. It is important to know that this option does not infer the joint posterior of all the parameters, but rather a conditional posterior. This entry needs to contain a list of the parameters for which a distribution is already provided in the same order as they are in the `provided_params.txt` file. This option requires an attribute pointing to the relative path of the provided parameter distribution. Here is an example:

```
<providedparameters file="provided_params.txt">k_P gamma_P</providedparameters>
```

A simple example of a full `<LFNS>` block can look like this:

```
<LFNS>
    <experiments>0</experiments>
    <N>1000</N>
    <H>1</H>
    <r>500</r>
    <epsilon>0.001</epsilon>
    <dpgmmiterations>100</dpgmmiterations>
</LFNS>
```

4 Defining different experiments

4.1 The <data> block

`<data>` The data block provides the information about the location of the data files for each of the experiments used for the LFNS inference. The `<data>` block can contain multiple `<dataset> (required for LFNS)` blocks defining each individual dataset.

`<dataset>` **(required for LFNS)** The `<dataset>` block contains the locations for the data files the files with the measurement timepoints for each experiment. It is understood that the files provided in each dataset block correspond to the experiment provided in the dataset block.

`<experiment>` **(required for LFNS)** Provides the name of the experiment corresponding to the data and time files in the same `<dataset>` block.

`<datafile>` **(required for LFNS)** Provides the relative path for the datafile of the corresponding experiment. Each measurement is expected to be in a separate row and each row should have exactly as many entries as there are measurement timepoints. A data file can contain one or multiple measurements (depending on the measurement model file) and can contain one or multiple timeseries trajectories.

`<timefile>` **(required for LFNS)** Provides the relative path for the timefile of the corresponding experiment and data file. It must have one row with exactly the same number of entries as each row of the data file.

This is an example of a `<data>` block for two experiments

```
<data>
   <dataset>
      <experiments>0</experiments>
      <datafile>./data_exp_0.txt</datafile>
      <timefile>./times_exp_0.txt</timefile>
   </dataset>
   <dataset>
      <experiments>ctrl</experiments>
      <datafile>./data_exp_ctrl.txt</datafile>
      <timefile>./times_exp_ctrl.txt</timefile>
   </dataset>
</data>
```

Since in real inference settings then number of experiments used for the inference can get rather large, it is unpractical to have a `<dataset>` block for each experiment. To avoid this it is possible to provide several entries for each entry of a `dataset` block. The example above can thus be written as

```
<data>
   <dataset>
      <experiments>0 ctrl</experiments>
      <datafile>./data_exp_0.txt ./data_exp_ctrl.txt</datafile>
      <timefile>./times_exp_0.txt ./times_exp_ctrl.txt</timefile>
   </dataset>
</data>
```

If the files `times_exp_0.txt` and `times_exp_ctrl.txt` are identical we can also write

```
<data>
   <dataset>
      <experiments>0 ctrl</experiments>
      <datafile>./data_exp_0.txt ./data_exp_ctrl.txt</datafile>
      <timefile>./times_exp_0.txt</timefile>
   </dataset>
</data>
```

In this case the file `times_exp_0.txt` will be used as `<timefile>` argument for both experiments.

4.2 The <inputs> block

<inputs> The **<inputs>** block defines the different experiments. This block allow to define to set parameters to certain values in various pulses in time. It provides possibilities to define the input parameters, the number, duration and strength of the pulses. The **<inputs>** block can contain several **<input>** block each defining particular inputs to the system.

<input> Each **<input>** block contains the information about each particular input.

<experiments> Provides the name of the experiment corresponding input defined in the **<input>** block.

<inputparam> The parameter affected by the defined input. The parameter defined will be set to zero (regardless of any previous provided value for this parameter) and will then attain the value provided in the **<strength>** entry of the **<input>** block whenever the pulse is active. If multiple **<input>** blocks are defined for the same parameter and multiple pulses are active at the same time, their strengths are added.

<period> The period at which the input pulse is applied.

<strength> The strength of the applied pulse.

<duration> The duration of the applied pulse.

<numpulses> The number of the applied pulses.

<startingtime> The time of the first applied pulse.

An example of such an input definition might look as follows

```
<inputs>
   <input>
      <experiments>0</experiments>
      <period>20</period>
      <strength>10</strength>
      <duration>2</duration>
      <numpulses>2</numpulses>
      <inputparam>k_P</inputparam>
      <startingtime>20</startingtime>
   </input>
   <input>
      <experiments>ctrl</experiments>
      <period>1000</period>
      <strength>2</strength>
      <duration>100</duration>
      <numpulses>1</numpulses>
      <inputparam>k_P</inputparam>
      <startingtime>20</startingtime>
   </input>
</inputs>
```

This block defines two experiments 0 and ctrl. The experiment 0 gives 2 pulses of strength 10 to theparameter `k_P`. This means the parameter `k_P` will be set to 0, except during the timpoints $[20, 22]$ and $[40, 42]$ during which it will be set to 10. Similarly for the experiment ctrl, the parameter `k_P` will be set to 2 starting from timepoint 20 till at the timepoint 120.

Similarly as for the `<data>` block, multiple inputs can be defined at once, so the above `<inputs>` block can also be written as

```
<inputs>
   <input>
      <experiments>0 ctrl</experiments>
      <period>20 1000</period>
      <strength>10 2</strength>
      <duration>2 100</duration>
      <numpulses>2 1</numpulses>
      <inputparam>k_P</inputparam>
      <startingtime>20</startingtime>
   </input>
</inputs>
```

We note that the input will be applied even when the input parameter is fixed as through the `<fixedparam>` entry. So to avoid confusion we recommend to fix each input parameter to 0 using the `<fixedparam>` entry.

5 Defining priors: the <parameter> block

The parameter block can be used to define the priors for the inference of the parameters (currently only Uniform and Log-Uniform distributions are available), define their bounds and fix parameters to particular values.

`<bounds>` The `<bounds>` block can contain multiple `<bound>` blocks that can each define the upper and lower bounds for a number of parameters.

- `<bound>` Each `<bound>` block contains three entries: `<parameters>`, `<lowerbounds>` and `<upperbounds>`
  - `<parameters>` This entry contains a list of parameters (separated by "," or " ") for which the bounds are defined.
  - `<lowerbounds>` The lower bound for all the parameters in the `<parameters>` entry. Alternatively a number of lower bounds equal to the numbers of parameters in `<parameters>` can be provided, where each number is interpreted to be the lower bound for the corresponding parameter.
  - `<upperbounds>` The upper bound for all the parameters in the `<parameters>` entry. Alternatively a number of upper bounds equal to the numbers of parameters in `<parameters>` can be provided, where each number is interpreted to be the upper bound for the corresponding parameter.

`<fixedvalues>` This block provides the possibility fix some of the model parameters to a particular value. If a model parameter is fixed, it will be treated as if it was hard coded into the program. It will not need to be provided as an input for the `simulate` command and will not be inferred by the LFNS algorithm. This can be useful when one would like to keep a general model but only wants to infer some of the parameters while keeping the other ones fixed.

- `<parameters>` A list of the parameter that should be fixed separated by "," or " ".
- `values` A list of the parameter values corresponding to the parameter names in the `<parameters>` entry separated by "," or " ".

`<scales>` This entry allows to set the scale of certain parameters to `linear` or `log`. In practice this indicates weather a uniform or a log-uniform prior is applied to the parameters. The `<scales>` block contains only one entry, with the names of the parameters and scale as an attribute. An example of a `<scales>` block is shown here:

```
<scales>
   <parameters scale="linear">k_P gamma_P</parameters>
</scales>
```

A full example of the `<parameter>` block is illustrated in the following

```
<parameters>
   <fixedparams>
      <parameters>mRNA_mu</parameters>
      <values>10</values>
   </fixedparams>
   <scales>
      <parameters scale="linear">mRNA_scale P_scale</parameters>
   </scales>
   <bounds>
      <bound>
         <parameters>k gamma k_P gamma_P</parameters>
         <lowerbounds>1e-5</lowerbounds>
         <upperbounds>1e1</upperbounds>
      </bound>
      <bound>
         <parameters>mRNA_scale P_scale</parameters>
         <lowerbounds>1</lowerbounds>
         <upperbounds>100</upperbounds>
      </bound>
   </bounds>
</parameters>
```

Bibliography

Jan Mikelson. *Nested Sampling for biochemical reaction networks with applications to ERK signalling: A likelihood-free approach.* PhD thesis, ETH Zurich, 2019.

Jan Mikelson and Mustafa Khammash. Likelihood-free nested sampling for biochemical reaction networks. *bioRxiv*, page 564047, 2019.