An Advanced Environment for Hybrid Modeling and Parameter Identification of Biological Systems

Sabrina Proß Bernhard Bachmann
University of Applied Sciences Bielefeld
Am Stadtholz 24, 33609 Bielefeld, Germany
http://www.fh-bielefeld.de/ammo

Abstract

Biological systems are often very complex so that an appropriate formalism is needed for modeling their behavior. Hybrid Petri nets, consisting of time-discrete as well as continuous Petri net elements, have proven to be ideal. This formalism was implemented based on the Modelica language. Several Petri net components are structured within an advanced Petri net library. A special sub-library contains so-called wrappers for specific biological reactions to simplify the modeling procedure.
The Petri net models developed with the Dymola tool can be connected to Matlab Simulink to use all the Matlab power for parameter identification, sensitivity analysis and stochastic simulation.
This paper illustrates the usage of the Petri net library, the coupling to Matlab Simulink and further processing of the simulation results with algorithms in Matlab. In addition, the application is demonstrated by modeling the metabolism of Chinese Hamster Ovary Cells.
Keywords: Biological Systems; Petri nets; Parameter Identification

1 Introduction

The procedure of modeling and simulation is displayed in Figure 1. The beginning of the process is founded by a set of objectives and purposes which are translated together with current knowledge of the system into a list of specific hypotheses.
The next stage is to find a formalism which can model the defined hypotheses by specific mathematical equations. The modeling of biological systems demand often a combination of continuous and discrete equations, a differential equation system sole is often not sufficient. Examples are gene regulation and processes where the organism switches from substance production to consumption or vice versa when special environmental conditions occur. This kind of process takes place within the metabolism of Chinese Hamster Ovary Cells (CHO-Cells) which switches from lactate and ammonium production to consumption after a specific change of environmental conditions. In addition, the antibody production starts only when special environmental conditions occur. The concrete mechanism is part of section 5.

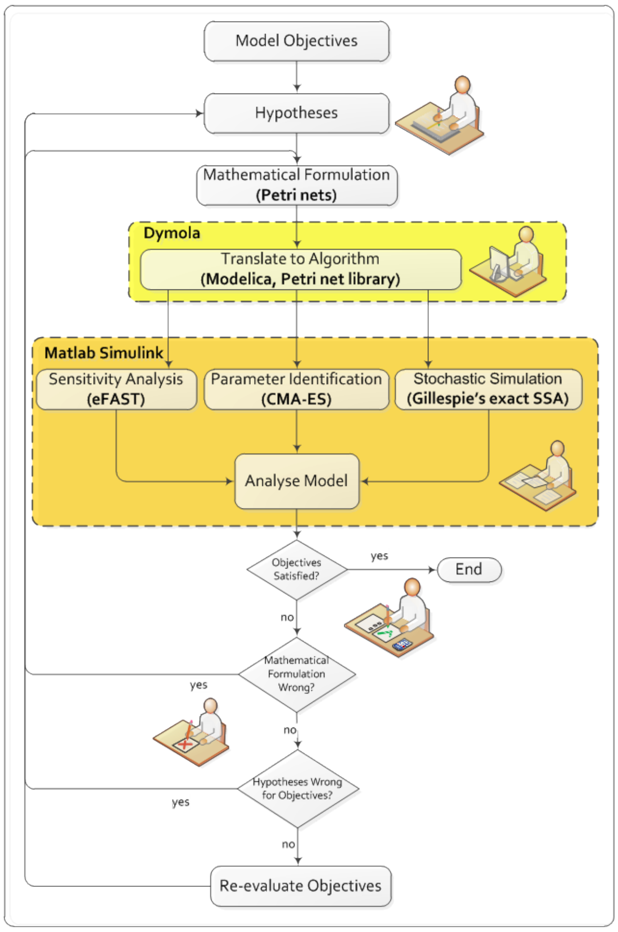

Model Objectives
Hypotheses
Mathematical Formulation (Petri nets)
Dymola
Translate to Algorithm (Modelica, Petri net library)
Matlab Simulink
Sensitivity Analysis (eFAST)
Parameter Identification (CMA-ES)
Stochastic Simulation (Gillespie's exact SSA)
Analyse Model
Objectives Satisfied?
yes
End
no
Mathematical Formulation Wrong?
yes
no
Hypotheses Wrong for Objectives?
yes
no
Re-evaluate Objectives

Figure 1: The modeling procedure

The hybrid Petri net concept, consisting time-discrete as well as continuous Petri net elements, fulfills all the required biological conditions and is thusly applicable for the realization of the CHO-model. The biological pools, like e.g. metabolites, genes, proteomes and signals are represented by places. The reactions between them can be modeled by transitions. This transfer of biological systems to Petri nets was first introduced by Reddy [1]. An introduction in the basic Petri net concepts is given in section 2.

Since the Modelica language provides all necessary features to implement the Petri net formalism it has been chosen to develop an advanced Petri net library, whereby the Petri net component models consist of differential, algebraic and discrete equations. Section 3 gives an introduction to the Petri net library.

On the basis of an established Petri net model, several calibration and analysis methods can be applied by coupling Dymola with Matlab Simulink. Section 4 gives an introduction in selected methods for parameter identification, sensitivity analysis and stochastic simulation as well as their realization within Matlab Simulink.

The modeling procedure is done if the model satisfies all determined objectives. Otherwise the mathematical formulation is wrong or the hypotheses are not correct for the desired objectives. In this case the modeling procedure has to be restarted at the respective stage.

2 Petri Nets

The Petri net formalism for graphical modeling of concurrent and nondeterministic processes was first introduced by Carl Adam Petri in 1962 [2]. A Petri net is mathematically a directed, 2-colored and bipartite graph. The property 2-colored implies the division in two unique node sets which are called *transitions* and *places* and only places can be connected to transitions or transitions to places according to the bipartite attribute. The places are represented graphically by circles and transitions by rectangles. A place models a state, for example of an object or a condition, while a transition models the change of states, for example activities or events.

Every place can contain an integer number of *tokens*. These tokens are represented graphically by little, black dots or numbers inside the places. A concrete determination of the token number of a place is called *state of the place* and a concrete determination of the token numbers of every place is called the *state of the Petri net*. Furthermore, the directed edges can have integer weightings which are written at the edges.

Following all places in the previous area of a transition are called *previous places* and all in the past area are called *past places*. Similarly, the transitions in the previous area of a place are called *previous transitions* and all in the past area are called *past transitions* (see Figure 2).

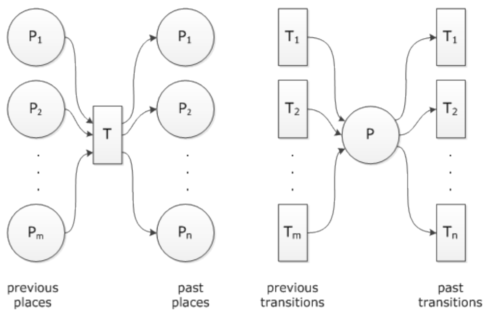


Figure 2: Previous and past places, and previous and past transitions

A transition is *ready-to-fire* if all previous places have at least as much tokens as the edge weightings. A ready-to-fire transition *fires* by removing as much tokens as the edge weightings from all previous places and by adding as much tokens as the edge weightings to all past places.

Figure 3 shows at the top an example of a Petri net where the transitions $T1$ and $T2$ are ready-to-fire and the others not. The Petri net at the bottom displays the new state after firing transition $T1$ and $T2$.

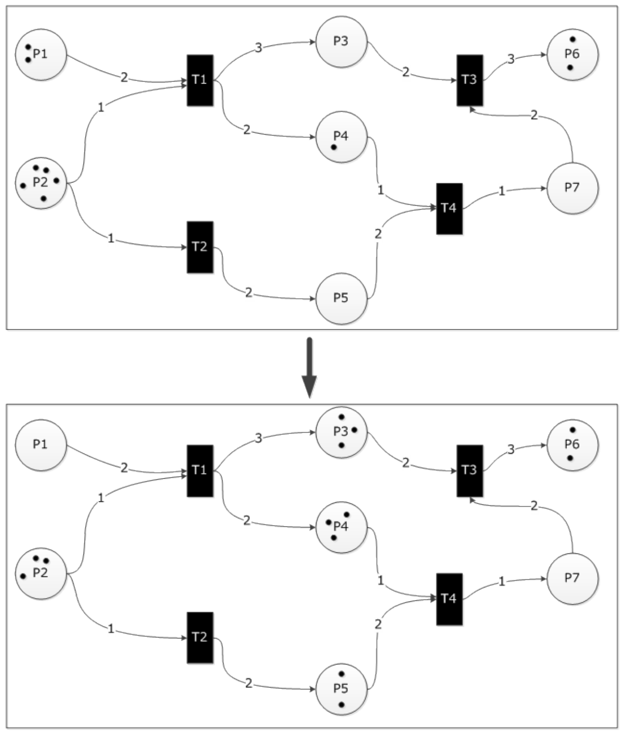


Figure 3: Petri net example, top: transitions *T1* and *T2* are ready-to-fire, bottom: new state of the Petri net after firing transition *T1* and *T2*

In the last years, the basic Petri net concept, described above has been more and more extended in order to model different kind of applications (e.g. biological systems). The first extension is that every place in a Petri net has a lower and upper limit of

tokens. These Petri nets are called *Petri nets with capacities*.
Biological applications demand not alone capacitated places but also limitations especial for each edge from places to transitions. In this conjunction, the lower bound of an edge is called *threshold*, the upper bound is called *inhibition* and the Petri net is called *Petri net with edge bounds*.
The fixed edge weightings can be replaced by dynamical ones which may depend on the current token number of a place. In this manner, not only integers can be written at the edges but also the name of a place. This Petri net extension is called *self-modified Petri net* and was first introduced by Valk [3].
These self-modified Petri nets can be further expanded to *functional Petri nets* by allowing functions as edge weightings which may depend on token numbers of several places [4].
For the simulation of a Petri net it is necessary to associate time with its behavior. One possibility to do this is that every transition gets a delay. A delay is the time period that the respective state change takes. This Petri net concept is called *timed Petri net*.
This concept can be modified to *stochastic Petri nets* by random delays, i.e. the fixed values are replaced by random numbers that change at every activation point in time. The delays are exponentially distributed random numbers, whereby the characteristic parameter λ can depend on token numbers of several places (see e.g. [5], [6]).
Biochemical reactions occur in most cases continuously. In order to model these reactions, the discrete Petri net concept has to be transferred to a continuous one [7]. The most serious difference between discrete and *continuous Petri nets* is that token numbers are real and that transitions fire continuously. A function is assigned to every edge of a continuous Petri net depending on token numbers of several places just like functional Petri nets. These functions specify the speed of the firing process and are the right side of differential equations. A continuous Petri net is an ordinary differential equation system whose structure can change within time.
Additionally, the modeling of biological systems demands often a combination of discrete and continuous processes. Hybrid Petri nets which contain discrete as well as continuous Petri net elements accomplish this [8].
The following connections are allowed within hybrid Petri nets

- ✓ discrete place → discrete transition
- ✓ discrete transition → discrete place
- ✓ continuous place → continuous transition
- ✓ continuous transition → continuous place
- ✓ continuous place → discrete transition
- ✓ discrete transition → continuous place

Not allowed are the connections

- ✗ discrete place → continuous transition
- ✗ continuous transition → discrete place

3 Petri Net Library

The advanced Petri net library described in this paper bases on the previous ones developed in Modelica ([9], [10], [11]). The improvements are:

- o Discrete Petri nets
  - Edges can have integer or functional weightings depending on token numbers of several places
  - Edges can have integer bounds (threshold and inhibition values)
  - If a place has a bottleneck, the connected transitions are enabled randomly with different probabilities
- o Continuous Petri nets
  - Generalization of the discrete Petri net concept to the continuous one
  - Edges can have functional weightings depending on token numbers of several places
  - Places can have minimum and maximum capacities and edges can have bounds (threshold and inhibition values)
- o Hybrid Petri nets
  - Combination of discrete and continuous Petri net elements to hybrid Petri nets

The advanced Petri net library is structured in five sub-libraries: Discrete, Continuous, Stochastic, Reactions and Global. Additionally, there are packages for Interfaces, Constants, Functions and Blocks which are used within component models (see Figure 4). The Petri net elements of the library are represented by the icons in Figure 5.

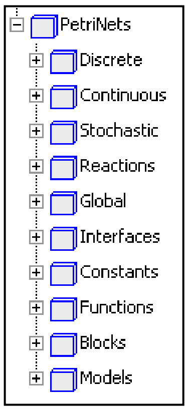
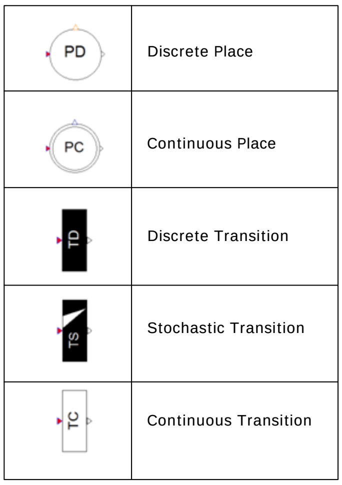


Figure 4: Structure of the Petri Net library



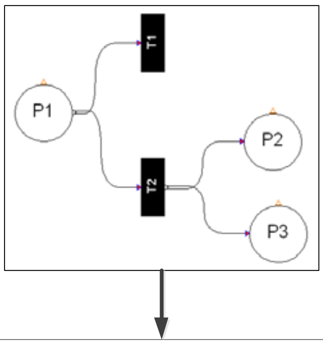
PD	Discrete Place
PC	Continuous Place
TD	Discrete Transition
TS	Stochastic Transition
TC	Continuous Transition

Figure 5: Icons of the Petri net library

The implementation details of places and transitions can be found in [12]. This Petri net library has been improved concerning the connectors of places and transitions since new modeling features are available in Modelica 3.2. The components have been upgraded by using the `connectorSizing` annotation [13].

```
parameter Real nPast=1 annotation
      (Dialog(connectorSizing=true);
parameter Real nPre=1 annotation
      (Dialog(connectorSizing=true);
```

The parameters `nPast` and `nPre` are used as dimension size of the vectors of connectors. The Dymola tool set these parameters automatically, i.e. they appear not in the property dialog. If a new connection is drawn, the respective parameters are incremented by one and a new `connect`-equation is created for the new highest index. Figure 6 shows a Petri net example as component diagram and the corresponding `connect`-equations that are created automatically by drawing a line from place to transition. In regard to this upgrade, some of the parameters of the place and transition models have to be entered as vectors since they belong to their edges. Exemplary, the vector parameter `add` of a transition which contains all weightings of the edges to its past places (cp section 3.2). The weighting of the edge from transition $T2$ in Figure 6 to place $P2$ is supposed to be 5 and the weighting of the edge from $T2$ to $P3$ is supposed to be 8. Then the parameter `add` has to be `add={5,7}`. The first entry in the `add`-vector corresponds to the first connection starting from $T2$ indexed with [1] and the second entry corresponds to the second connection indexed with [2].
The drawback of this concept is that the knowledge about the indices of the connections is needed to guarantee the right assignment of the edge weightings. When a connection is added or deleted one must keep attention that the entries of the vector parameters are still in the right order. But on the other hand the big advantage is that components can have an arbitrary amount of previous and past, which simplifies the modeling process enormously.



```
connect(P1.outTransition[1], T1.inPlaces[1]);
connect(P1.outTransition[2], T2.inPlaces[1]);
connect(T2.outPlaces[1], P2.inTransition[1]);
connect(T2.outPlaces[2], P3.inTransition[1]);
```

Figure 6: The `connectorSizing` annotation

3.1 Place Model

Table 1 contains the parameters which can be set in all places (discrete, stochastic and continuous) and Table 2 shows those that are only part of discrete places. The parameter `enablePast` is explained in Figure 7.

Table 1: Parameters of both places (discrete and continuous)

Identifier	Description	Type/ Default
startTokens	The number of tokens that the place contains at the beginning of the simulation. In the discrete case nonnegative integer and in the continuous case nonnegative real numbers can be entered.	scalar/ zero
minTokens	The minimum number of tokens that the place must always contain. In the discrete case nonnegative integer and in the continuous case nonnegative real numbers can be entered.	scalar/ zero
maxTokens	The maximum number of tokens that the place can contain. In the discrete case nonnegative integer and in the continuous case nonnegative real numbers can be entered.	scalar/ infinite

Table 2: Parameters only of discrete places

Identifier	Description	Type/ Default
enablePast	Enabling probabilities of the past transitions. If a place has not enough tokens to enable all its past transitions, a random decision must be made whereby the respective transition is chosen with the entered probability. The sum of all enabling probabilities has to be one (see Figure 7).	vector/ each entry $\frac{1}{nPast}$
enablePre	Enabling probabilities of the previous transitions. If a place cannot gain tokens from all its previous transitions due to its maximum capacity, a random decision must be made whereby the respective transition is chosen with the entered probability. The sum of all enabling probabilities has to be one.	vector/ each entry $\frac{1}{nPre}$

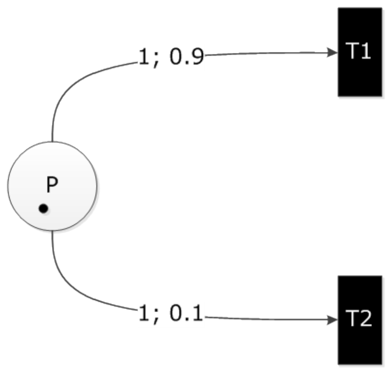


Figure 7: Petri net example for the `enablePast`-parameter: The place P has one token, not enough to fire in T1 and T2 simultaneously since both edge weightings are one. A random decision is applied where T1 is chosen with the probability 0.9 and $T2$ with the probability 0.1. The sum of all enabling probabilities of a place has to equal one.

3.2 Transition Model

The parameters available in all transition models are summarized in Table 3. Table 4 contains the parameters, which are only part of the discrete transition. Those that can be only set in stochastic transitions are shown in Table 5.

Table 3: Parameters of all transitions (discrete, stochastic and continuous)

Identifier	Description	Type/ Default
sub	Weightings of edges starting from previous places. In the discrete and stochastic case nonnegative integers and functions can be entered and in the continuous one nonnegative real numbers and functions are allowed. With the ".t"-notation one can access the tokens of a place for the edge weightings, e.g. `sub={2.9*P1.t}`.	vector/ each entry 1
add	Weightings of edges ending in past places. In the discrete and stochastic case nonnegative integers and functions can be entered and in the continuous one nonnegative real numbers and functions are allowed. With the ".t"-notation one can access tokens of a place for edge weightings, e.g. `add={0.45*P1.t}`.	vector/ each entry 1
inhibition	Upper bound of edges staring from previous places. Nonnegative integers can be entered in the discrete and stochastic case and nonnegative real numbers in the continuous case.	vector/ each entry infinite
threshold	Lower bound of edges starting from previous places. Nonnegative integer can be entered in the discrete and stochastic case and nonnegative real numbers in the continuous case.	vector/ each entry zero
con	Condition which has to be true so that the transition can become active and can fire, e.g. `time>9.7`.	scalar/ true

Table 4: Parameter only of a discrete transition

Identifier	Description	Type/ Default
delay	The time that a discrete transition waits after its activation before it fires.	scalar/ 1

Table 5: Parameters only of a stochastic transition

Identifier	Description	Type/ Default
c	Constant for the pre-defined lambda functions (see parameter `lambdaFunc`)	scalar/ 1
lambdaFunc	Pre-defined function for the lambda calculation; choice between Stochas-	scalar/ Stochastic mass

	tic mass action hazard function and Stochastic level hazard function (see [14])	action
lambda	User-defined function for lambda instead of the pre-defined lambda functions (Stochastic mass action hazard function and Stochastic level hazard function)	scalar

3.3 Reactions Sub-Library

The Petri net models of the Discrete, Continuous and Stochastic sub-libraries can be wrapped into models for different kinds of biological reactions to simplify the modeling process. These model components are organized in the sub-library Reactions which is also divided in several sub-libraries for different reaction types. Till now there are:

- Reaction kinetics
- Enzyme kinetics
- Growth kinetics
- Culture strategies
- Process activations

The detailed wrapping process is explained in [15].

3.4 Animation in Dymola

The simulation results can be displayed by either plots of selected token numbers or by an animation. By the latter the degree of redness changes during time according to the token number of the place, i.e. a red place has many tokens and a white place is empty. The redness degree can be scaled from 0 to 100 by the green Settings-box which is a component of the Global sub-library (see Figure 8).

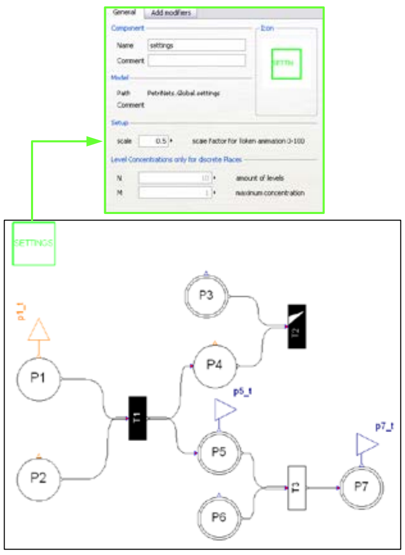

General | Add modifiers
Component
Name: settings
Comment:
Icon
Model
Path: PetriNets.Global.settings
Comment
Setup
scale: 0.5 — scale factor for token animation 0-100
Level Concentrations only for discrete Places
N: 10 — amount of levels
M: 1 — maximum concentration

Figure 8: Scaling of the redness degree by the Settings-component of the global sub-library

This animation is realized by the annotation `DynamicSelect` [13]

```
annotation(fillColor = DynamicSelect
({255,255,255},if scale<100 then
{255,255-2.55*tokenscale,255-
2.55*scale} else {255,0,0})
```

Figure 9 shows the redness change of a Petri Net example during time. This animation offers a good way for analyzing large and complex Petri Nets.

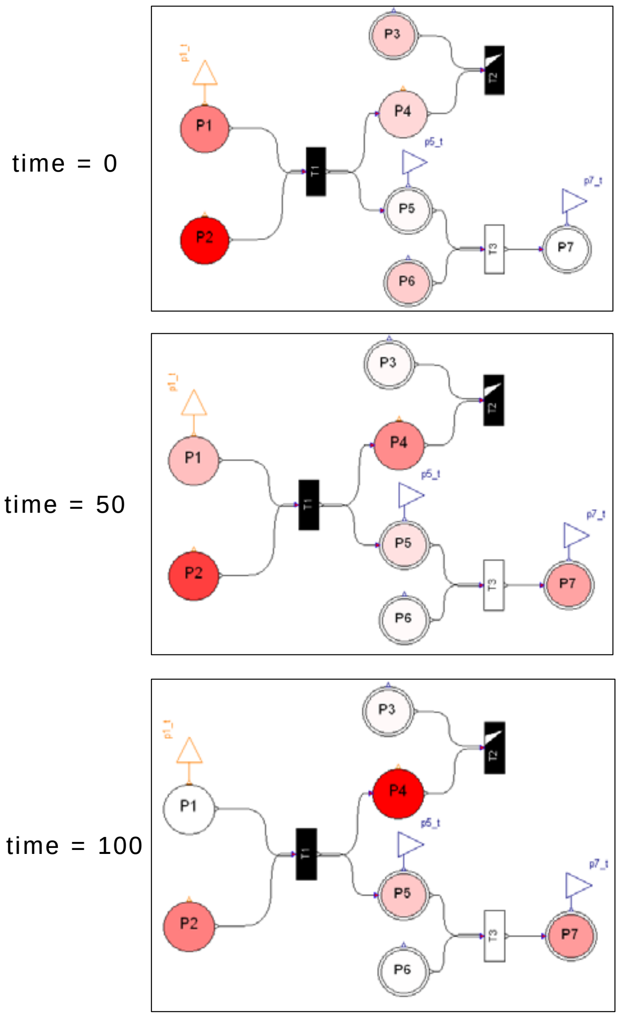


Figure 9: Animation of a Petri net in Dymola; the token distribution of the Petri net example, top: at the beginning of the simulation, middle: after a simulation of 50 time units, bottom: after a simulation of 100 time units; the degree of redness corresponds to the token numbers, i.e. a red place has many tokens and a white place is empty

4 Model Calibration and Analysis

Once a Petri net model is established, the next step according to Figure 1 is to get further insight in the model parameter characteristics by sensitivity analysis, parameter identification and stochastic simulation. For this, a connection between the Petri net model in Dymola and Matlab Simulink has to be created to benefit from the power of Matlab.

4.1 Connection Dymola and Matlab Simulink

For parameter optimization, sensitivity analysis and stochastic simulation, it is necessary to simulate the model several times with different parameter settings. Dymola offers a possibility to connect a Modelica model to Matlab by a Simulink interface (DymolaBlock) and a set of Matlab m-files [16]. Figure 10 displays at the top a Petri net modeled by the Petri net library in Dymola and at the bottom the corresponding Simulink model. If the token number of a place over the time is needed in Matlab for further calculations, one has to create a connector above the respective place. This is an orange `IntegerOutput` connector in the case of a discrete place or a blue `RealOutput` connector if it is a continuous place. In the Petri Net example of Figure 10 the token numbers of the places $P1$, $P5$ and $P7$ are needed in Matlab, whereby $P1$ is a discrete place with an `IntegerOutput` connector and $P5$ and $P7$ are continuous with a `RealOutput` connector. The DymolaBlock in Simulink generates a connector for all places connected with an output connector in Dymola. These connectors can then be connected via a bus to an outport so that these simulation results are saved in a matrix and are available in the Matlab environment for further calculations. In the same manner it is also possible that Petri net models get inputs from Matlab via a connection between a Simulink source and a Modelica `IntegerInput` or `RealInput` connector.
To connect a Dymola-model with Simulink, one has to enter the model name and its path in the property dialog of the DymolaBlock (see Figure 11). After that, the model can be complied and the parameters can be set. The parameters can be also set within Matlab by special m-files.

```
[p,x0,pnames,x0names]=loaddsin(
                    'example.txt');
p=setParameterByName(pnames,p,
                    'param1',25);
setParametersFDsin('ex/example1',
                    pnames,p,x0names,x0)
```

For a detailed description see [16]. After the parameter setting the Simulink model can be simulated by the prompt

```
sim(model,timespan,options,ut).
```

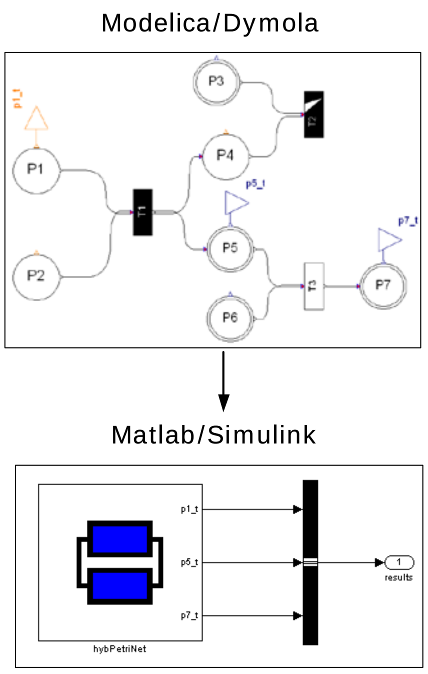
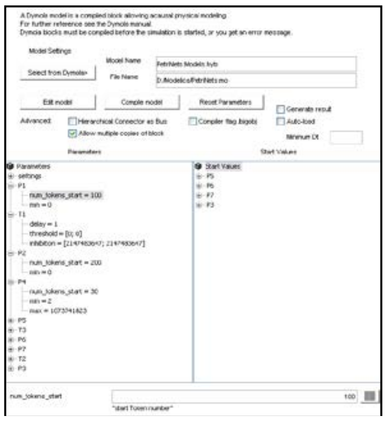


Figure 10: Connecting Dymola and Matlab Simulink by a Simulink Interface (DymolaBlock), top: Petri net modeled by the Petri net library in Dymola, bottom: Simulink interface of the Dymola-model in Matlab Simulink


A Dymola model is a compiled block allowing acausal physical modeling.
For further reference see the Dymola manual.
Dymola blocks must be compiled before the simulation is started, or you get an error message.

Model Settings
Model Name: PetriNets.Models.hyb
File Name: D:/Modelica/PetriNets.mo

Advanced: ☐ Hierarchical Connector as Bus ☑ Allow multiple copies of block ☐ Compiler flag /bigobj ☐ Generate result ☐ Auto-load
Minimum DT

```
Parameters
+ settings
- P1
  num_tokens_start = 100
  min = 0
- T1
  delay = 1
  threshold = [0; 0]
  inhibiton = [2147483647; 2147483647]
- P2
  num_tokens_start = 200
  min = 0
- P4
  num_tokens_start = 30
  min = 2
  max = 1073741823
+ P5
+ T3
+ P6
+ P7
+ T2
+ P3
```

```
Start Values
+ P5
+ P6
+ P7
+ P3
```

num_tokens_start 100
"start Token number"

Figure 11: Parameter dialog of the DymolaBlock in Simulink

4.2 Sensitivity Analysis

The goal of the sensitivity analysis is to apportion the uncertainty in model output to the different sources of uncertainty in the model input (e.g. model parameter) [17]. It can uncover technical errors in the model, identify critical regions in the input space, establish priorities for research, simplify models and defend against falsifications of the analysis [18]. The techniques can be divided in local and global methods. The *local sensitivity analysis* concentrates on the local impact of the input factors on the model and is usually performed by computing the partial derivatives of the output functions with respect to the input factors. In contrast, the *global sensitivity analysis* considered the influence of the input factors according to a given range of variation and a probability density function.

The Petri net component models contain among others discrete equations (e.g. `when`-equation). Following, several events are detected within the simulation of a Petri net so that the function of token development over time is mostly not differentiable and not continuous. This engender that the local sensitivity analysis methods cannot be applied and one has to access the global methods. Various global sensitivity analysis methods are available (see e.g. [19]). Some of them were implemented in Matlab and can be chosen with respect to the model structure.

For the example in section 5 the *extended Fourier Amplitude Sensitivity Test* (eFAST) has been applied. The eFAST is a variance-based method, i.e. it uses the variance as indicator of the importance of the input factors. The enormous advantage of this method is its applicability independent of any assumptions about the model structure. That means it works for models with a linear as well as a nonlinear relationship between input and output and it is unimportant if this relationship is monotonic or not. The Fourier Sensitivity Amplitude Test (FAST) was developed by Cukier and others in the 1970s ([20], [21], [22]) and extended by Saltelli and others in 1999 [23]. The main idea behind the FAST is to convert the m-dimensional integral of the mean value of the output Y

$$\langle Y \rangle = \int \dots \int Y(\boldsymbol{x}) \cdot P(\boldsymbol{x})\, d\boldsymbol{x}$$

into a one-dimensional integral in s by using the transformation functions, called *search curves*

$$x_i = G_i(\sin(\omega_i s)),$$

where $\boldsymbol{x}$ is the n-dimensional vector of input factors, $P(\boldsymbol{x}) = \prod_{i=1}^{m} P_i(x_i)$ is the product of their probability density functions, $s \in (-\pi, \pi)$ is the *search variable* and $\{\omega_i\}$ is a set of integer angular frequencies. If the frequencies ω_i and the search curves $G_i(\sin(\omega_i s))$ are chosen appropriate, the expectation of Ycan be approximated by

$$E(Y) = \frac{1}{2\pi} \int_{-\pi}^{\pi} Y(s) ds,$$

where $Y(s) = Y\big(G_1(\sin(\omega_1 s)), \dots, G_n(\sin(\omega_n s))\big)$. The variance of Y can be approximated by the Fourier coefficients A_k and B_k [24]

$$Var(Y) \approx 2 \sum_{k=1}^{\infty} A_k^2 + B_k^2,$$

where

$$A_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} Y(s) \cos(js)\, ds$$
$$B_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} Y(s) \sin(js)\, ds\,.$$

The expressions $E(Y)$ and $Var(Y)$ provides a way to estimate the expectation and the variance of the output Y. Additionally, the application of the FAST method demands a definition of the frequencies ω_i, the search curves G_i and the number of sufficient points at which the model is evaluated to allow a numerical approximation of the expectation and the variance. For this it is referred to ([20], [21], [24], [23], [19]).

The contribution of factor x_i to the variance of Y can then be approximated by the partial variance

$$D_i = 2 \sum_{p=1}^{M} (A_{p\omega_i}^2 + B_{p\omega_i}^2),$$

where M is the maximum harmonic that is considered [22]. The ratios

$$S_i = \frac{D_i}{Var(Y)}$$

provide a way to rank the input factors according to their contribution to the variance of output Y, called *first-order sensitivity coefficients*. An implementation of the FAST method can be found in [25].

This method has been improved by Saltelli and others [23] to the extended FAST method (eFAST) which computes the total (all-effects) contribution of each input factor to the output variance. This is done by assigning a usually high frequency ω_i to an investigated factor x_i and a set of almost identical and usually low, but different from ω_i, frequencies $\omega_{\sim i}$ to all remaining factors. The partial variance of the complementary set can be computed by

$$D_{\sim i} = 2 \sum_{p=1}^{M} (A_{p\omega_{\sim i}}^2 + B_{p\omega_{\sim i}}^2),$$

whereby $D_{\sim i}$ measures the effect of any orders that do not involve the factor x_i. The *total sensitivity coefficients* are then given by

$$TS_i = 1 - \frac{D_{\sim i}}{Var(Y)}.$$

For the choice of the frequencies and a detailed description of the eFAST method, it is referred to [23].

4.3 Parameter Identification

The parameter identification deals with methods that estimate the unknown model parameter to adapt the model behavior as good as possible to the reality e.g. to the measured experimental data. The objective of the optimization procedure can be for example the least squares

$$f(\boldsymbol{x}) = \sum_{j=1}^{n}\sum_{l=1}^{r}\left(\frac{y_j(t_l,\boldsymbol{x}) - d_j(t_l)}{d_j(t_l)}\right)^2 \rightarrow \min \qquad \text{Eq. 1}$$

where $\boldsymbol{x}$ is the input vector, $y_j(t_l,\boldsymbol{x})$ is the j-th model output corresponding to the input $\boldsymbol{x}$ at time t_l, $d_j(t_l)$ is the measurement of the j-th output at time t_l, n is the number of measured outputs and r is the number of measured points in time. Several numerical methods are well known to solve this problem. An overview can be found in [26]. However, if the underlying model is a Petri net the standard methods like Gauss-Newton or Levenberg-Marquardt are inoperative due to the non-differentiability and non-continuity of the model output. Global optimization methods has to be use that work without derivatives. For an overview of global optimization methods it is referred to [27]. In the example in section 5 a special evolution strategy, the *Covariance Matrix Adaption Evolution Strategy* (CMA-ES), is applied to estimate the unknown model parameters. Evolution strategies consist in general of the following steps:

1. *Initialization*: a specific number of individuals is generated by a random procedure.
2. *Recombination*: one or more parents produce one or more offspring. Several methods are documented (see e.g. [28]).
3. *Mutation*: minor change of the offspring.
4. *Selection*: a specific number of the best individuals form the parents of new generation. Several methods can be found in [27]. The next iteration begins with the recombination.

The parameter estimation with the CMA-ES algorithm bases on the mentioned steps above and additionally two main principles plays an important role [29].

The first is to increase the probability of a successful mutation according to the maximum likelihood principle. Therefore, the mean of the distribution is updated such that the likelihood of previously successful candidate solutions is maximized. Furthermore, the covariance matrix of the distribution is updated such that the likelihood of previously realized successful steps to appear again is increased. These updates can be interpreted as a natural gradient descent and consequently the CMA conducts an iterated principal component analysis of successful steps while retaining all principle axes. The covariance matrix adaption is to learn about the second order model of the underlying objective function.

The second is to record two paths of time evolution of the distribution mean of the strategy. Such a path contains important information about the correlation between consecutive steps. The first path is used for the covariance matrix adaption procedure and the second is used for a step-size control. A detailed description of the algorithm and its implementation can be found in [29].

4.4 Stochastic Simulation

Stochastic simulation is to simulate many realizations of a stochastic model (stochastic Petri net) and to study the arising results. One method is Gillespie's algorithm [30] which was created to simulate chemical and biochemical reaction systems efficiently and accurately. It is a modification of the Monte Carlo method. The elementary steps according to an underlying stochastic Petri net model are:

1. *Initialization*: Initialize the number of tokens in the stochastic Petri net, reaction constants, and the random number generators.
2. *Monte Carlo step*: Generate random number to determine the next transitions to fire as well as their delays. The delays are exponentially distributed random numbers, whereby the characteristic parameter λ is proportional to token numbers of the previous places.
3. *Update*: Update token numbers based on the firing transitions.
4. *Iterate*: Go back to step 2 unless the simulation time has been exceeded.

5 Example: Modeling the metabolism of Chinese Ovary Cells

The Chinese Hamster Ovary (CHO) Cells produce antibodies which are part of many pharmaceuticals [31]. Additionally, they release the waste-products lactate and ammonium which can inhibit their growth and antibody production when specific concentrations are exceeded ([32], [33], [34]).

Experiments were performed by growing the CHO-Cells in shaking flaks, whereby they were fed with the nutrients glucose and glutamine. They produced by conversion of these nutrients antibodies, ammonium and lactate. By the latter ones it is assumed that they cannot only be produced by the CHO-Cells but also consumed when the environmental conditions are appropriate ([35], [36]). Figure 12 displays these coherencies and Figure 13 represents the experimental data of CHO-Cells growing in shaking flaks.

The experiments were performed by the University of Applied Sciences Bielefeld, Biotechnology department [37]. The cells grow till day 4 (exponential growing phase) afterwards they pass over to the stationary phase for 2 days where approximately as much cells grow as die. Finally, more cells die than new ones grow thus the curve of living cells decreases and the curve of death cells increases (death phase).

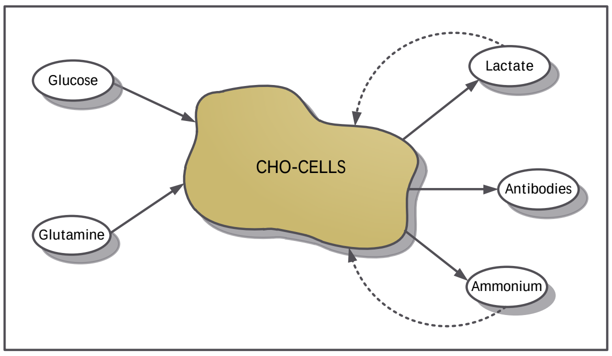


Figure 12: The main metabolism of the CHO-cells

The nutrient Glucose is exhausted at the end of the experiment and the waste-product lactate is produced till day 4 and afterwards it is consumed. They convert it back to pyruvate which enters the citric acid cycle (TCA-cycle) [36]. Here, it is assumed that they start the lactate consumption when a specific lactate concentration is exceeded. Additionally, the ammonium concentration decreases after 4 days and the glutamine concentration increases. In this conjunction, it seems likely that the CHO-cells can convert ammonium back to glutamine when the glutamine concentration falls below a specific value.
The Antibody production starts first after 2.5 days and does not stop until the end of the experiment. At this point the supposition is that the cells start the production first when the glucose becomes limiting.

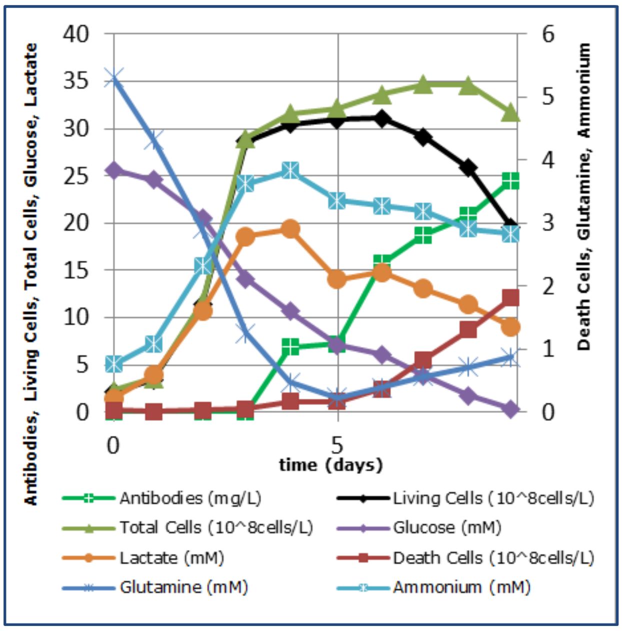


Figure 13: Experimental data of CHO-Cells growing in shaking flaks

A continuous Petri Net models the dynamics of the CHO-cells (see Figure 14). This Petri Net covers a lot of different differential equation systems. Which of them is chosen depends on the environmental conditions. At the beginning of the experiment, it represents the following ODEs

$$\frac{dX_t}{dt} = \mu \cdot X_v \quad \text{Eq. 2}$$

$$\frac{dX_d}{dt} = \mu_d \cdot X_v \quad \text{Eq. 3}$$

$$\frac{dX_v}{dt} = (\mu - \mu_d) \cdot X_v \quad \text{Eq. 4}$$

$$\frac{dGlc}{dt} = -q_{glc} \cdot X_v \quad \text{Eq. 5}$$

$$\frac{dGlu}{dt} = -q_{glu} \cdot X_v - k_{sd} \cdot Glu \quad \text{Eq. 6}$$

$$\frac{dLac}{dt} = q_{lac} \cdot X_v \quad \text{Eq. 7}$$

$$\frac{dAmm}{dt} = q_{amm} \cdot X_v + k_{sd} \cdot X_v \quad \text{Eq. 8}$$

$$\frac{dAb}{dt} = 0 \quad \text{Eq. 9}$$

$$\begin{gathered} X_t(0) = X_{t0},\ X_d(0) = X_{d0},\ X_v(0) = X_{v0}, \\ Glc(0) = Glc_0, Glu(0) = Glu_0, \\ Lac(0) = Lac_0, Amm(0) = Amm_0, \\ Ab(0) = Ab_0 \end{gathered} \quad \text{Eq. 10}$$

where X_t is the concentration of total cells (10^8 cells/L), X_d is the concentration of death cells (10^8 cells/L), X_v is the concentration of living cells (10^8 cells/L), Glc is the glucose concentration (mM), Glu is the glutamine concentration (mM), Lac is the lactate concentration (mM), Amm is the Ammonium concentration (mM), Ab is the Antibody concentration (mg/L), μ is the specific growth rate (1/d), μ_d is the specific death rate (1/d), q_{glc} is the specific glucose uptake rate (mmol/10^8 cells/d), q_{glu} is the specific glutamine uptake rate (mmol/10^8 cells/d), k_{sd} is the constant for the spontaneous degradation of glutamine, q_{lac} is the specific lactate production rate (mmol/10^8 cells/d), q_{amm} is the specific ammonium production rate (mmol/10^8 cells/d) and X_{t0}, X_{d0}, X_{v0}, Glc_0, Glu_0, Lac_0, Amm_0 and Ab_0 are the initial concentrations.
The conversion from glutamine to ammonium can take place in two different ways: the CHO-cells can perform it ($q_{glu} \cdot X_v$, $q_{amm} \cdot X_v$) and it can occur within the medium by spontaneous decomposition ($k_{sd} \cdot Glu$) [38].
No antibodies are produced at the beginning of the experiment hence the differential equation is set to

zero. After a specific change of the environmental conditions, the antibody production starts and Eq. 9 has to be changed to

$$\frac{dAb}{dt} = q_{ab} \cdot X_v \qquad \text{Eq. 11}$$

where q_{ab} is the antbody production rate (mg/10^8 cells/d). The supposition is that the decreasing glucose concentration initiates the antibody production. In terms

$$\frac{dAb}{dt} = \begin{cases} 0, & Glc \geq 14\,mM \\ q_{ab} \cdot X_v, & Glc < 14\,mM \end{cases} \qquad \text{Eq. 12}$$

A similar switching situation occurs by the lactate concentration. At the beginning the dynamics are represented by Eq. 7 and after a specific change of the environmental conditions, especially the lactate concentration passes a threshold, the dynamics are described by

$$\frac{dLac}{dt} = \begin{cases} q_{lac} \cdot X_v - q_{lacs} \cdot X_v, Lac \geq 19\,mM \\ q_{lac} \cdot X_v, Lac < 19\,mM \end{cases} \qquad \text{Eq. 13}$$

where q_{lacs} is the specific lactate consumption rate (mmol/10^8 cells/d). The glutamine consumption and production, respectively, leads to the following switching equation, whereby the change is initiated by the decreasing glutamine concentration

$$\frac{dGlu}{dt} = \begin{cases} -q_{glu} \cdot X_v - k_{sd} \cdot Glu, Glu \geq 0.4\,mM \\ -q_{glu} \cdot X_v - k_{sd} \cdot Glu + q_{glus} \cdot X_v, Glu < 0.4\,mM \end{cases} \qquad \text{Eq. 14}$$

where q_{glus} is the specific glutamine production rate (mmol/10^8 cells/d) and the corresponding dynamics for the ammonium concentration are

$$\frac{dAmm}{dt} = \begin{cases} q_{amm} \cdot X_v + k_{sd} \cdot Glu, Glu \geq 0.4\,mM \\ q_{amm} \cdot X_v + k_{sd} \cdot Glu - q_{amms} \cdot X_v, Glu < 0.4\,mM \end{cases} \qquad \text{Eq. 15}$$

where q_{amms} is the specific ammonium consumption rate (mmol/10^8 cells/d).

Figure 14 displays the Petri net modeling the discussed conditions above (Eq. 2-Eq. 5, Eq. 10, Eq. 12-Eq. 15). All places and transitions are continuous. Table 6 contains the places and their corresponding substances and Table 7 summarizes the information of the transitions. The orange Activation-boxes are wrappers of the Reactions sub-library and they work like a discrete switch. When the token number of the connected place exceeds the entered value of the parameter `tres` or fall below the entered value of the parameter `inhi`, the connected transition becomes active and remains active until one of the connected places becomes empty in contrast to the threshold and inhibition values of the transitions. Everything inside the brown cell mass occurs within the cells and outside of this picture are the reaction for the spontaneous decomposition of glutamine and the substances that the cells release to the medium. The total amount of cells, the sum of living cells and death cells, is modeled by an algebraic equation `Xt_t=Xv.t+Xd.t.`

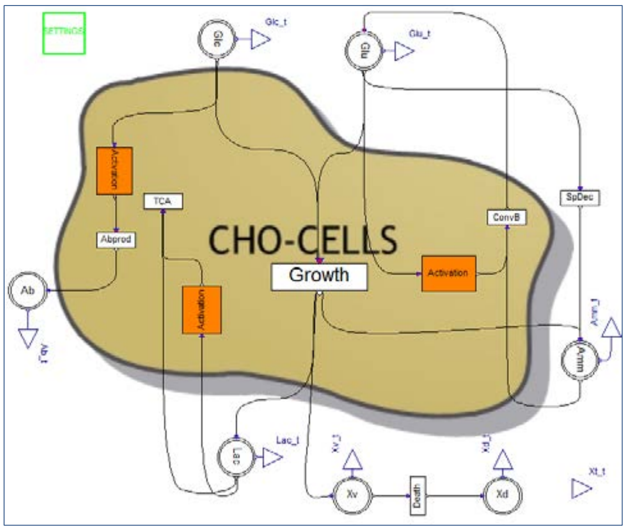


Figure 14: Petri net model of the CHO metabolism in Figure 12

Table 6: Places of the CHO-Model in Figure 14 and the corresponding substances

Place	Substance
Xv	Concentration of living CHO-Cells
Xd	Concentration of death CHO-Cells
Glc	Glucose concentration
Glu	Glutamine concentration
Lac	Lactate concentration
Amm	Ammonium concentration

The experimental data of Figure 13 are approximated by smoothing splines to get further insight to the relations between the respective specific rates. The rates at the beginning of the simulation can be calculated by the following equations

$$\mu = \frac{1}{X_v} \cdot \frac{dX_{total}}{dt} \qquad \text{Eq. 16}$$

$$\mu_d = \frac{1}{X_v} \cdot \frac{dX_{death}}{dt} \qquad \text{Eq. 17}$$

$$q_{glc} = -\frac{1}{X_v} \cdot \frac{dGlc}{dt} \qquad \text{Eq. 18}$$

$$q_{glu} = -\frac{1}{X_v} \cdot \left(\frac{dGlu}{dt} + k_{sd} \cdot Glu\right) \qquad \text{Eq. 19}$$

$$q_{lac} = \frac{1}{X_v} \cdot \frac{dLac}{dt} \qquad \text{Eq. 20}$$

$$q_{amm} = \frac{1}{X_v} \cdot \left(\frac{dGlu}{dt} - k_{sd} \cdot Glu\right) \qquad \text{Eq. 21}$$

The specific antibody production rate can be calculated after day 2.5 when the cells start with the production

$$q_{ab} = \frac{1}{X_v} \cdot \frac{dAb}{dt} \qquad \text{Eq. 22}$$

The relations analysis yields the following equation structures for the specific rates

$$\mu = \mu_{max} \cdot \frac{Glu}{K_{Glu} + Glu} \qquad \text{Eq. 23}$$

$$\mu_d = \mu_{dmax} \cdot \frac{KD_{Glc}}{KD_{Glc} + Glc} \qquad \text{Eq. 24}$$

$$q_{glc} = \frac{1}{Y_{X,Glc}} \cdot \mu \qquad \text{Eq. 25}$$

$$q_{glu} = \frac{1}{Y_{X,Glu}} \cdot \mu \qquad \text{Eq. 26}$$

$$q_{lac} = Y_{Lac,Glc} \cdot q_{glc} \qquad \text{Eq. 27}$$

$$= Y_{lac,Glc} \cdot \frac{1}{Y_{X,Glc}} \cdot \mu$$

$$q_{amm} = Y_{Amm,Glu} \cdot q_{glu} \qquad \text{Eq. 28}$$

$$= Y_{Amm,Glu} \cdot \frac{1}{Y_{x,Glu}} \cdot \mu$$

$$q_{ab} = k_{ab} \qquad \text{Eq. 29}$$

$$q_{lacs} = k_{lacs} \qquad \text{Eq. 30}$$

$$q_{amms} = k_{amms} \qquad \text{Eq. 31}$$

$$q_{glus} = Y_{Glu,Amm} \cdot q_{amms} \qquad \text{Eq. 32}$$

with the parameters μ_{max} (1/d) as maximum specific growth rate and K_{Glu} as constant of the Monod kinetics, μ_{dmax} (1/d) as maximum specific death rate, KD_{Glc} as constant of the death kinetics (mM), $Y_{X,Glc}$ (10^8 cells/mmol), $Y_{X,Glu}$ (10^8 cells/mmol), $Y_{Lac,Glc}$ (mol/mol), $Y_{Amm,Glu}$ (mol/mol) and $Y_{Glu,Amm}$ (mol/mol) as yield coefficients, k_{ab} (mg/10^8 cells) as constant of the antibody production, k_{lacs} (mmol/10^8 cells) as constant of the lactate consumption and k_{amms} as constant of the ammonium consumption.

For performing a sensitivity analysis and afterwards a parameter optimization for the 13 model parameters, the Petri net model in Dymola (Figure 14) has to be connected to Matlab via a Simulink interface as described in section 4.1. The simulation results of all token numbers are needed in Matlab, thus all places have a blue `RealOutput` connector (Figure 14) so that a corresponding port at the Simulink interface is provided.

Table 7: Transitions of the CHO-Model in Figure 14 and the corresponding reactions and the edge weighting functions (Eq. 2-Eq. 5, Eq. 10, Eq. 12-Eq. 15)

Transition	Reaction	Weightings		Conditions
Growth	Cell growth	Glc → Growth	$q_{glc} \cdot Xv.t$	
		Glu → Growth	$q_{glu} \cdot Xv.t$	
		Growth → Xv	$\mu \cdot Xv.t$	
		Growth → Lac	$q_{lac} \cdot Xv.t$	
		Growth → Amm	$q_{amm} \cdot Xv.t$	
Death	Cell death	Xv → Death	$\mu_d \cdot Xv.t$	
		Death → Xd	$\mu_d \cdot Xv.t$	
TCA	Lactate consumption	Lac → TCA	$q_{lacs} \cdot Xv.t$	Activation Box thres=19
Abprod	Antibody production	Abprod → Ab	$q_{ab} \cdot Xv.t$	Activation Box inhi=14
SpDec	Spontaneous decomposition of glutamine to ammonium	Glu → SpDec	$k_{sd} \cdot Glu.t$	
		SpDec → Amm	$k_{sd} \cdot Glu.t$	
ConvB	Conversion of ammonium back to glutamine	Amm → ConvB	$q_{amms} \cdot Xv.t$	Activation Box inhi=0.4
		ConvB → Glu	$q_{glus} \cdot Xv.t$	

Before the 13 parameters of the model are estimated a global sensitivity analysis is performed to get further insight in the parameter characteristics. This analysis is the basis of the following parameter optimization since less sensitive parameters can be fixed during the optimization process to increase the chance of a converging optimization algorithm. The global sensitivity analysis is performed by Matlab with eFAST method explained in section 4.2. Therefore, the model is simulated several times with different parameter settings and each time the objective function in Eq. 1 is evaluated. The eFAST-method measures the contribution of each parameter to the variance of this objective function, whereby the parameters are varied in a specific range. If a parameter contributes less to the variance, this parameter cannot be identified within an optimization procedure and has to be fixed on the other hand if a parameter contributes much to the variance of the objective function this parameter is identifiable.

The results of the global sensitivity analysis, i.e. the contribution of each parameter to the variance of the objective function, are displayed in Figure 15. It becomes clear that 7 of 13 parameters contribute 91 % of the variance so that 6 parameters

$$Y_{X,Glu}, \mu_{dmax}, KD_{glc}, k_{lacs}, Y_{Lac,Glc}, Y_{Amm,Glu}$$

can be fixed during the optimization process and 7

$$k_{sd}, Y_{Glu,Amm}, k_{amms}, Y_{X,Glc}, \mu_{max}, K_{Glu}, k_{ab}$$

have to be optimized.

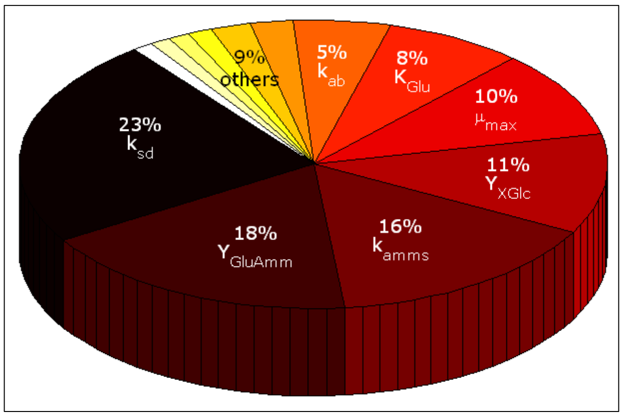


Figure 15: Model variance contribution of every parameter according to the eFAST method

The parameter optimization is performed by CMA-ES method explained in section 4.3. The optimization procedure takes place in Matlab via a Simulink interface. Figure 16 displays the results of this optimization procedure which show a good agreement with the experimental data.

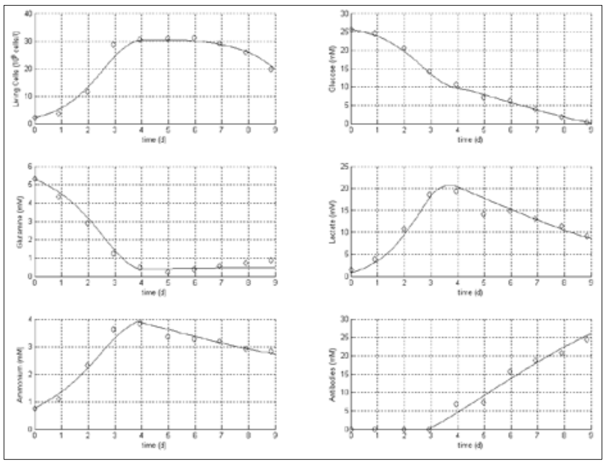


Figure 16: Results of the parameter optimization procedure

To achieve a good model of the CHO-metabolism, it is also possible to choose a stochastic approach, i.e. a stochastic Petri net model and a stochastic simulation according to Gillespie's algorithm as described in section 4.4. The edge weightings of the continuous approach in Table 7 are now the dynamic values of the characteristic parameter λ of the exponential distribution by which the delay of the stochastic transition is chosen randomly at every activation point in time (cp. Section 2). The transformation of the parameters of the continuous to the stochastic model is well studied and can be found in [39]. Figure 17 displays the CHO-metabolism modeled by a stochastic Petri net, whereby the places are discrete and the transitions are stochastic. The tokens represent here different concentration levels like it is presented in [14]. One token equals to 0.5 (mM, 10^8Cells/l, mg/l), thus there are $N+1=90+1$ different levels since the maximum concentration (M) is set to 45. The values of M and N can be entered in the green settings-box which has to be a part of every model and can be found in the Global-library. This stochastic Petri net model is also connected to a Simulink interface in Matlab so that the stochastic simulation can take place within an m-file. The results are displayed in Figure 18 where 500 Simulation are accomplished and the means were built with 10 simulations, respectively.

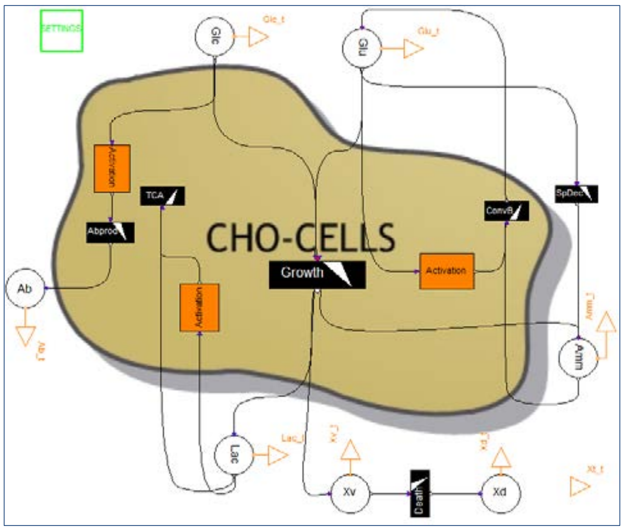


Figure 17: Stochastic Petri Net of the main CHO-metabolism in Figure 12

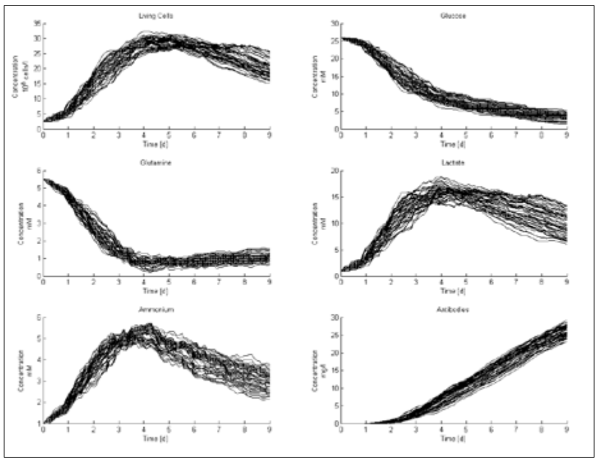


Figure 18: The stochastic simulation results according to Gillespie's algorithm of the stochastic Petri Net model in Figure 17

6 Conclusions

The Petri net library in Modelica is a good instrument for hybrid modeling of biological systems. The advantages of this approach are:

- The object-oriented modeling language Modelica is able to model discrete places and transitions as well as stochastic and continuous ones. The places and transitions are models that easily can be changed, modified, or expanded so that further Petri net extensions can be implemented fast.
- The language allows the realization of hybrid models by combining discrete and continuous processes. The hybrid simulation with discrete events and the solution of continuous differential equations is then performed by Dymola or by another Modelica-tool.
- The Reactions sub-library offers a fast and simple way to build up a model and further reactions can be easily added.
- The hierarchical modeling concept of Modelica enables a structuring of the models on different levels which is useful when the model is complex and used by different persons with different aims.
- The Petri net animation of Dymola offers a way to get insight of the token distribution of large and complex Petri nets.
- The coupling of Dymola-models and Simulink-models allows the simulation of a model many times and use the arising simulation results for subsequent calculations so that stochastic simulation, sensitivity analysis and parameter identification in Matlab is possible.
- The Petri net library can be integrated in other Petri net modeling tools by parsing the Petri net of the respective tool (e.g. XML-format) to Modelica-text and simulate it via a batch process where the simulation results are saved in a data file.

In this manner the new Petri net library in combination with Matlab Simulink leads to a complete environment for hybrid modeling of biological systems.

References

[1] V.N. Reddy, M.L. Mavrovouniotis, M.N. Liebman (Eds.), Petri net representations in metabolic pathways: Proc Int Conf Intell Syst Mol Biol, 1993.

[2] C.A. Petri, Communication with automata, Rome Air Development Center, Research and Technology Division, 1966.

[3] R. Valk, Self-modifying nets, a natural extension of Petri nets, Automata, Languages and Programming (1978) 464–476.

[4] R. Hofestädt, S. Thelen, Quantitative modeling of biochemical networks, In Silico Biology 1 (1998) 39–53.

[5] F. Bause, P.S. Kritzinger, Stochastic Petri Nets, Vieweg, 2002.

[6] M. Heiner, D. Gilbert, R. Donaldson, Petri nets for systems and synthetic biology, Formal Methods for Computational Systems Biology (2008) 215–264.

[7] D. Gilbert, M. Heiner, From Petri nets to differential equations-an integrative approach for biochemical network analysis, Petri Nets and Other Models of Concurrency-ICATPN 2006 (2006) 181–200.

[8] A. Doi, S. Fujita, H. Matsuno, M. Nagasaki, S. Miyano, Constructing biological pathway models with hybrid functional Petri nets, In Silico Biology 4 (2004) 271–291.

[9] P.J. Mosterman, M. Otter, H. Elmqvist, Modeling Petri nets as local constraint equations for hybrid systems using Modelica, Citeseer, Reno, Nevada, Proceedings of SCS Summer Simulation Conference, 1998, pp. 314–319.

[10] S.M. Fabricius, Extensions to the Petri Net Library in Modelica, ETH Zurich, Switzerland (2001).

[11] M. Otter, K.E. Årzén, I. Dressler (Eds.), StateGraph-a Modelica library for hierarchical state machines: 4th International Modelica Conference, 2005.

[12] S. Proß, B. Bachmann, A Petri Net Library for Modeling Hybrid Systems in OpenModelica, Como, Italy, Modelica Conference proceedings, 2009.

[13] Modelica Association, Modelica - A Unified Object-Oriented Language for Physical Systems Modeling Language Specification Version 3.2 (2010).

[14] D. Gilbert, M. Heiner, S. Lehrack (Eds.), A unifying framework for modelling and analysing biochemical pathways using Petri nets: Proceedings of the 2007 international conference on Computational methods in systems biology, Springer-Verlag, 2007.

[15] S. Proß, B. Bachmann, R. Hofestädt, K. Niehaus, R. Ueckerdt, F.J. Vorhölter, P. Lutter, Modeling a Bacterium's Life: A Petri-Net Library in Modelica, Como, Italy, Modelica Conference proceedings, 2009.

[16] Dynasim AB, Dymola-Dynamic Modeling Laboratory-User Manual Volume 2, Lund/Sweden, 2010.

[17] A. Saltelli, Sensitivity analysis in practice: a guide to assessing scientific models, John Wiley & Sons Inc, 2004.

[18] A. Saltelli, M. Ratto, T. Andres, Global sensitivity analysis: the primer, John Wiley & Sons Ltd, 2008.

[19] A. Saltelli, K. Chan, E.M. Scott, Sensitivity analysis, Wiley New York, 2000.

[20] R.I. Cukier, C.M. Fortuin, K.E. Shuler, A.G. Petschek, J.H. Schaibly, Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. I Theory, The Journal of Chemical Physics 59 (1973) 3873–3876.

[21] J.H. Schaibly, K.E. Shuler, Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. II Applications, The Journal of Chemical Physics 59 (1973) 3879–3888.

[22] R.I. Cukier, J.H. Schaibly, K.E. Shuler, Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. III. Analysis of the approximations, The Journal of Chemical Physics 63 (1975) 1140–1149.

[23] A. Saltelli, S. Tarantola, K.P. Chan, A quantitative model-independent method for global sensitivity analysis of model output, Technometrics 41 (1999) 39–56.

[24] R.I. Cukier, H.B. Levine, K.E. Shuler, Nonlinear sensitivity analysis of multiparameter model systems, Journal of Computational Physics 26 (1978) 1–42.

[25] M. Koda, G.J. Mcrae, J.H. Seinfeld, Automatic sensitivity analysis of kinetic mechanisms, Int. J. Chem. Kinet. 11 (1979) 427–444.

[26] J. Nocedal, S.J. Wright, Numerical optimization, Springer-Verlag New York Inc, New York, Berlin, Heidelberg, 1999.

[27] T. Weise, Global Optimization Algorithms – Theory and Application, 2009. http://www.it-weise.de/.

[28] T. Bäck, Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms, Oxford University Press, USA, 1996.

[29] N. Hansen, The CMA evolution strategy: a comparing review, Towards a new evolutionary computation (2006) 75–102.

[30] D.T. Gillespie, Exact stochastic simulation of coupled chemical reactions, The journal of physical chemistry 81 (1977) 2340–2361.

[31] Birch, JR, A.J. Racher, Antibody production, Advanced drug delivery reviews 58 (2006) 671–685.

[32] N. Kurano, C. Leist, F. Messi, S. Kurano, A. Fiechter, Growth behavior of Chinese hamster ovary cells in a compact loop bioreactor. 2. Effects of medium components and waste products, Journal of biotechnology 15 (1990) 113–128.

[33] M.S. Lao, D. Toth, Effects of ammonium and lactate on growth and metabolism of a recombinant Chinese hamster ovary cell culture, Biotechnology progress 13 (1997) 688–691.

[34] S.S. Ozturk, M.R. Riley, B.O. Palsson, Effects of ammonia and lactate on hybridoma growth, metabolism, and antibody production, Biotechnol. Bioeng. 39 (1992) 418–431.

[35] Y.S. Tsao, A.G. Cardoso, R.G. Condon, M. Voloch, P. Lio, J.C. Lagos, B.G. Kearns, Z. Liu, Monitoring Chinese hamster ovary cell culture by the analysis of glucose and lactate metabolism, Journal of biotechnology 118 (2005) 316–327.

[36] A. Provost, G. Bastin, S.N. Agathos, Y.J. Schneider, Metabolic design of macroscopic bioreaction models: application to Chinese hamster ovary cells, Bioprocess and biosystems engineering 29 (2006) 349–366.

[37] J. Link, Charakterisierung der Prozessparameter tierischer Zellkulturen in Schüttelinkubatoren. Bachelor thesis, Bielefeld, 2010.

[38] S.S. Ozturk, B.O. Palsson, Chemical decomposition of glutamine in cell culture media: effect of media type, pH, and serum concentration, Biotechnology progress 6 (1990) 121–128.

[39] D.J. Wilkinson, Stochastic modelling for systems biology, Chapman & Hall/CRC, 2006.