On Robustness Computation and Optimization in BIOCHAM-4

François Fages, Sylvain Soliman

On Robustness Computation and Optimization in BIOCHAM-4

François Fages[1] and Sylvain Soliman[1]

Inria Saclay - Ile de France, Lifeware group
Francois.Fages@inria.fr, Sylvain.Soliman@inria.fr

Abstract. BIOCHAM-4 is a tool for modeling, analyzing and synthesizing biochemical reaction networks with respect to some formal, yet possibly imprecise, specification of their behavior. We focus here on one new capability of this tool to optimize the robustness of a parametric model with respect to a specification of its dynamics in quantitative temporal logic. More precisely, we present two complementary notions of robustness: the statistical notion of model robustness to parameter perturbations, defined as its mean functionality, and a metric notion of formula satisfaction robustness, defined as the penetration depth in the validity domain of the temporal logic constraints. We show how the formula robustness can be used in BIOCHAM-4 with no extra cost as an objective function in the parameter optimization procedure, to actually improve the model robustness. We illustrate these unique features with a classical example of the hybrid systems community and provide some performance figures on a model of MAPK signalling with 37 parameters.

1 Introduction

Computational systems biology aims at gaining a system level understanding of high-level biological processes from their biochemical realm. Formal methods from Computer Science have been soon introduced at the heart of this effort to go beyond mathematical modeling and master the complexity of cell processes. In particular, model-checking techniques have been used to analyze Boolean gene regulatory networks and signaling or control protein reaction networks. They have also been generalized, in particular in the hybrid systems community, to quantitative temporal logics such as MTL, MITL, STL, or in our case FO-LTL($\mathbb{R}_{\text{lin}}$) [14] for dealing with numerical constraints, fitting quantitative models to the time series data observed with increasing accuracy in biological experiments [16,9], controling cell processes in real-time, or designing circuits in synthetic biology [1,13].

One striking feature of natural biological processes is their robustness with respect to both external perturbations and their intrinsic stochasticity. Measuring and optimizing robustness is thus an important topic in biological modeling and a useful yet rare feature of modeling tools in this domain. In [11], Kitano gives a definition of robustness of a biological system with respect to a dynamical property and a parameter perturbation law, as the mean functionality of the

system. Such a statistical definition of robustness can be evaluated in a computational model by sampling the parameters' space according to their distribution laws, and checking the property on simulation traces, i.e. by runtime verification or monitoring. Since the capability of generating simulation traces is the only requirement, this can be done in a very general setting of non-linear hybrid systems.

In [7,14], different notions of robustness are proposed with the definition of a continuous degree of satisfaction of a temporal logic property on a trace. Such a satisfaction degree gives quantitative information on the satisfaction of the formula, as the distance to, and penetration depth within, the validity domain of the formula. In the simplest example of a threshold formula $x < c$, that satisfaction degree can be defined for instance as the value $c - x$, with negative values representing distance to satisfaction, and positive values the margins achieved for robust satisfaction.

In this paper, we present some unique features of BIOCHAM-4[1] for defining hybrid systems by reaction networks with rates and events, and focus on the use of quantitative temporal logic language to specify the dynamical properties of the system, observed or wished, verify them in the model, compute parameter sensitivity indices, measure robustness, synthesize model parameters and optimize robustness. We deal with the two complementary notions of robustness mentioned above, i.e. the statistical notion of model robustness to parameter change, and the formula satisfaction robustness. We show how the formula robustness can be used in BIOCHAM-4 with no extra cost as an objective function in the parameter optimization procedure to actually improve the robustness of the model with respect to parameter perturbations. We illustrate these features with a classical example of non-linear hybrid system, the bouncing ball example, and provide some performance figures on the Huang and Ferrell's model of MAPK signaling [10] with 37 parameters.

2 BIOCHAM Models

A BIOCHAM model is composed of a (multi)set of reactions with rate functions, and/or influences with forces, plus possibly events. Such models can be interpreted in a hierarchy of differential, stochastic, Petri net and Boolean semantics [4]. We will focus here on the differential semantics. They can be imported from model repositories such as BioModels using the BIOCHAM interface to SBML (SBML-qual for influence models), or from ODE models [3] through an interface to the XPP format.

This article comes with a set of examples (MAPK signaling, cell cycle, bouncing ball) available online under the form of BIOCHAM-4 Jupyter notebooks[2].

[1] `http://lifeware.inria.fr/biocham4`
[2] `https://lifeware.inria.fr/wiki/Main/Software#CMSB18`

3 Behavior Specifications in FO-LTL($\mathbb{R}_{\text{lin}}$)

3.1 Validity Domains of FO-LTL($\mathbb{R}_{\text{lin}}$) Constraints

In BIOCHAM, quantitative temporal properties of the behavior of a system can be formally specified in a first-order version of Linear Time Logic with free variables, linear constraints over $\mathbb{R}$, and quantifiers, named FO-LTL($\mathbb{R}_{\text{lin}}$). The grammar is:
$\phi ::= c \mid \neg\phi \mid \phi \Rightarrow \psi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \exists x\ \phi \mid \forall x\ \phi \mid \mathbf{X}\,\phi \mid \mathbf{F}\,\phi \mid \mathbf{G}\,\phi \mid \phi\,\mathbf{U}\,\phi \mid \phi\,\mathbf{W}\,\phi$
where c denotes linear constraints over state variables (including *Time*) and free variables. The *validity domain* $\mathcal{D}_{\pi,\phi} \in \mathbb{R}^k$ of an FO-LTL($\mathbb{R}_{\text{lin}}$) formula ϕ containing $k \geq 1$ variables on a finite trace $\pi = (s_0, \ldots, s_n)$ can be defined by recursively bottom-up from the validity domain of the basic constraints to the complex FO-LTL($\mathbb{R}_{\text{lin}}$) formulae by intersection, union and complementation [5]. An FO-LTL($\mathbb{R}_{\text{lin}}$) formula is false if one validity domain is empty, valid if the validity domains of all variables are $\mathbb{R}$, and satisfiable otherwise.

For instance, the formula $\mathbf{F}(A \geq 0.2)$ where A is a state variable expresses that the concentration of molecule A gets greater than 0.2 at some time point in the trace ($\mathbf{F}$). If needed, the precise time values where the concentration of A gets greater than the threshold value can be expressed by introducing a free variable t with an equality constraint to the real time variable, $\mathbf{F}(A \geq 0.2 \wedge t = \text{Time})$. Constraints between time variables can then relate the time of different events. The maximum value of a state variable A can be specified and set in a variable v by the formula $\mathbf{G}(A \leq v) \wedge \mathbf{F}(A \geq v)$, a local maximum (or plateau) by $\mathbf{F}(A \leq v \wedge \mathbf{X}(A = v \wedge \mathbf{X}\,A \leq v))$. FO-LTL($\mathbb{R}_{\text{lin}}$) formulae are very expressive and can be used to define complex oscillation properties, with pseudo-period constraints defined by delay constraints between the local maxima, and phase constraints defined by delays between the peaks of different state variables [6].

3.2 Implementation

The recursive definition of the validity domain of an FO-LTL($\mathbb{R}_{\text{lin}}$) formula on a finite trace is implemented in BIOCHAM-4 by generating a C program that implements the necessary loops, starting from the last time point to the first, and considering the subformulae in the bottom-up order, i.e. first from the linear constraints at the leaves, to the root of the syntactic tree. The call to the C compiler is responsible for slower response time than the previous interpreted implementation on small examples, but faster on large examples. The Parma Polyhedra Library (PPL)[3] is used to solve the linear constraints and simplify the representation of validity domains by finite lists of polyhedra.

Bound constraints, i.e. constraints of the form $x \leq c$ or $x \geq c$ where x is a variable and c a constant, define boxes as a particular kind of polyhedra. In that case, the validity domains are finite union domains of boxes, since they are obtained by intersection, union, complementation and projection of boxes.

[3] `http://www.bugseng.com/ppl`

However, it is worth noticing that even in the case of bound constraints, the validity domain of a temporal formula can contain an exponential number of polyhedra in the number of free variables in the FO-LTL($\mathbb{R}_{\text{lin}}$) formula [5]. The issue of trace simplification, such as keeping in the trace only the time points that are local extrema for one state variable, is also important to reduce computation time and justified in a number of practical cases [15].

4 Formula Robustness and Satisfaction Degree

In [13], the continuous satisfaction degree in the interval $[0, 1]$, of an FO-LTL($\mathbb{R}_{\text{lin}}$) formula ϕ on a trace π, was defined as the distance between the validity domain of the free variables $x_1, \ldots, x_k$ in ϕ and some objective values in $\mathbb{R}^k$ given as a valuation σ of the variables $x_1, \ldots, x_k$, i.e. a vector $\boldsymbol{v}_\sigma = (v_1, \ldots, v_k)$. This definition enforces that the violation degree is in $[0, +\infty)$ (0 when the formula $\sigma(\phi)$ is satisfied) and the satisfaction degree in $(0, 1]$ (1 when the formula is satisfied). Those notions can be generalized in order to take into account how *robustly* a formula is satisfied, by taking into account the penetration depth of the objective values in the validity domain, similarly to the space-time robustness defined in [2] for STL:

Definition 1. *The* violation degree $vd(\pi, \phi, \sigma) \in (-1, +\infty)$ *of an FO-LTL*($\mathbb{R}_{lin}$) *formula* ϕ *in a numerical trace* π *with respect to an objective valuation* σ *is*

$$\min_{\boldsymbol{v} \in D_{\pi,\phi}} d(\boldsymbol{v}, \boldsymbol{v}_\sigma) \text{ if } \boldsymbol{v}_\sigma \notin \mathcal{D}_{\pi,\phi}, \ \frac{1}{1 + \min_{\boldsymbol{v} \notin D_{\pi,\phi}} d(\boldsymbol{v}, \boldsymbol{v}_\sigma)} - 1 \text{ if } \boldsymbol{v}_\sigma \in \mathcal{D}_{\pi,\phi}.$$

The satisfaction degree $sd(\pi, \phi, \sigma) \in (0, +\infty)$ *of* ϕ *in* π *w.r.t.* σ *is* $\dfrac{1}{1 + vd(\pi, \phi, \sigma)}$

The first case is the same as [14], i.e. the distance in $[0, +\infty)$ between $\boldsymbol{v}_\sigma$ and the domain $\mathcal{D}_{\pi,\phi}$. In the second case, we get a negative number related to the distance between $\boldsymbol{v}_\sigma$ and the outside of the domain $\mathcal{D}_{\pi,\phi}$. The satisfaction degree is defined as in [14] but now provides a notion of formula robustness when its value is greater than 1. Indeed $sd(\pi, \phi, \sigma)$ is bounded by the radius of $\mathcal{D}_{\pi,\phi}$ and describes in the space of the variables $x_1, \ldots, x_k$ by how much one can change the objective $\boldsymbol{v}_\sigma$ while keeping $\sigma(\phi)$ satisfied on π.

In our implementation in BIOCHAM-4, the distance to, and penetration depth within, validity domains are not computed exactly, but approximated using the notions of generators and constraints in PPL. This is indeed sufficient to guide the search of parameter values and optimize formula robustness, using the Covariance Matrix Adaptive Evolutionary Strategy CMA-ES [8] as black-box continuous optimization tool with satisfaction degree as objective function.

5 Model Robustness and Parameter Sensitivity

The more classical notion of *model robustness* defined in [11] as the mean functionality with respect to a set of parameter perturbations P can be instanciated

in our setting for a FO-LTL($\mathbb{R}_{\text{lin}}$) formula ϕ and some objective σ as

$$R_{P,\phi,\sigma} = \int_{p \in P} \min(1, sd(\pi(p), \phi, \sigma)) prob(p) dp$$

In BIOCHAM this integral is evaluated by sampling (log-)normally-distributed parameter values given by the user. This sampling stops when a user-given number of samples is reached, or when the relative sample-standard-deviation becomes smaller than a given threshold. However, the computation may be computationally expensive with tens of numerical simulations needed and is generally not usable inside a search for model parameters.

6 BIOCHAM commands

The bouncing ball is a classical example of the hybrid systems community with which we can illustrate our approach. One single simulation is already quite time consuming (about 2s) because of the checking of the bouncing event condition. The FO-LTL($\mathbb{R}_{\text{lin}}$) formula `F(x<h /\ F(x>h))` gives the height of the first bounce in the free variable h (~ 4 in the companion notebook). Computing the validity domain with the command

```
validity_domain(F(x<h / F(x>h)))
```

and satisfaction degree

```
satisfaction_degree(F(x<h / F(x>h)), [h -> 3.5])
```

on such a simulation trace is slightly faster (about 0.1s). Now, the command

```
robustness(F(x<h / F(x>h)), [x0, K, D], [h->3.5]).
```

computes a value (0.884 in 160s) that quantifies how robustly is satisfied the formula with an objective for h of 3.5 when parameters x_0, K, D are perturbed (with default Gaussian distribution). On the other hand, the formula robustness necessitates a single simulation to get the value $sd(\pi, \phi, \sigma) = 1.4664$. This low computational cost makes it possible to use formula robustness as an optimization criterion within the parameter search procedure. One can ask BIOCHAM-4 to search for parameter values such that the violation degree vd is below -0.5 (i.e., $sd \geq 2$):

```
search_parameters(F(x<h /\ F(x>h)), [5<=x0<=10, 0.5<=K<=1.0, -0.1<=D<=0.0],
 [h->3.5], cmaes_stop_fitness: -0.5).
```

BIOCHAM-4 finds in 2.5s a solution with $x_0 = 9.555$, $K = 0.801$, $D = -0.0585$ such that $vd = -0.572$ and $sd = 2.339$. Computing again the model robustness as before we now get (in 101s) an improved model robustness of 0.930.

It is worth noting however that, in theory, increasing the formula robustness does not necessarily improve the model robustness. Indeed, the formula robustness may increase with a parameter set that approaches a frontier where the formula is no longer satisfied, in which case the model robustness with respect to parameter perturbation may decrease. In many practical cases however, though the systems we tackle are highly non-linear, improving the formula robustness also did improve the model robustness.

7 Evaluation on MAPK Signaling Network

In [10], the authors present the dynamics of a signaling network, namely the Mitogen Activated Protein Kinase (MAPK) cascade, with a model encompassing 22 species and fully relying on Mass-Action kinetics with 37 parameters. This model remains today a reference for the MAPK cascade as it properly describes the 3-level global structure and the 2-step nature of each level. Though it was supposed that explicit feedback reactions were necessary for obtaining oscillations in this model, it was demonstrated by Qiao et al. in 2007 [12] that this is actually not necessary. The authors explored blindly the space of 36 parameters and then established a bifurcation diagram in the 37th parameter in order to find oscillations.

The FO-LTL($\mathbb{R}_{\text{lin}}$) formula used in [14] to measure model robustness is $\mathbf{F}(\text{PPK} \geq up \wedge \mathbf{F}(\text{PPK} \leq up - amplitude))$, where PPK represents the output of the signaling cascade, i.e. the concentration of the doubly-phosphorylated kinase at the third level. By using that formula with an objective of for instance 0.5 for *amplitude*, one can actually search parameter values in a directed way, guided by continuous satisfaction degree. In order to ignore simple overshoots, one can consider the refined formula $\mathbf{F}(\text{PPK} \geq up \wedge \mathbf{F}(\text{PPK} \leq up - a_1 \wedge \mathbf{F}(\text{PPK} \geq up - a_1 + 0.5 * a_2)))$, with objectives of 0.5 and 1 for respectively a_1 and a_2. BIOCHAM-4 is able to find a solution for the 37 parameters in a few minutes (averaged to 5 minutes over 100 runs on a 3.6GHz Intel Core i7 machine) and to optimize formula robustness with no extra cost. This is also shown on a cell cycle model in the companion notebook mentionned above.

8 Conclusion

Building models that are robust to parameter variations is a key issue in systems biology and synthetic biology, because of both their fluctuating environment and the stochastic nature of biochemical reactions. BIOCHAM-4 is one of the very few tools to implement a metric notion of robustness for dynamical properties and optimize it by the parameter search procedure with no extra cost. Although not true in all generality for non-linear hybrid systems, optimizing formula robustness does improve in practice the statistical notion of model robustness with respect to parameter variations. The integration of formula robustness in the notion of satisfaction degree thus provides a simple and effective approach to the design of robust parameterization of reaction networks.

Acknowledgment: This work benefited from partial support from the ANR project HYCLOCK contract DS0401.

References

1. Courbet, A., Amar, P., Fages, F., Renard, E., Molina, F.: Computer-aided biochemical programming of synthetic microreactors as diagnostic devices. Molecular Systems Biology 14(4) (2018)

2. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: FORMATS 2010. Lecture Notes in Computer Science, vol. 6246, pp. 92–106. Springer-Verlag (2010)
3. Fages, F., Gay, S., Soliman, S.: Inferring reaction systems from ordinary differential equations. Theoretical Computer Science 599, 64–78 (Sep 2015)
4. Fages, F., Martinez, T., Rosenblueth, D., Soliman, S.: Influence networks compared with reaction networks: Semantics, expressivity and attractors. IEEE/ACM Transactions on Computational Biology and Bioinformatics (Feb 2018)
5. Fages, F., Rizk, A.: On temporal logic constraint solving for the analysis of numerical data time series. Theoretical Computer Science 408(1), 55–65 (Nov 2008)
6. Fages, F., Traynard, P.: Temporal logic modeling of dynamical behaviors: First-order patterns and solvers. In: del Cerro, L.F., Inoue, K. (eds.) Logical Modeling of Biological Systems, chap. 8, pp. 291–323. John Wiley & Sons, Inc. (2014)
7. Fainekos, G., Pappas, G.: Robustness of temporal logic specifications. In: Int. Workshop on Formal Approaches to Software Testing and Runtime Verification, FATES/RV'06. Lecture Notes in Computer Science, vol. 4262, pp. 178–192. Springer-Verlag (2006)
8. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
9. Heitzler, D., Durand, G., Gallay, N., Rizk, A., Ahn, S., Kim, J., Violin, J.D., Dupuy, L., Gauthier, C., Piketty, V., Crépieux, P., Poupon, A., Clément, F., Fages, F., Lefkowitz, R.J., Reiter, E.: Competing G protein-coupled receptor kinases balance G protein and β-arrestin signaling. Molecular Systems Biology 8(590) (Jun 2012)
10. Huang, C.Y., Ferrell, J.E.: Ultrasensitivity in the mitogen-activated protein kinase cascade. PNAS 93(19), 10078–10083 (Sep 1996)
11. Kitano, H.: Towards a theory of biological robustness. Molecular Systems Biology 3, 137 (2007)
12. Qiao, L., Nachbar, R.B., Kevrekidis, I.G., Shvartsman, S.Y.: Bistability and oscillations in the huang-ferrell model of mapk signaling. PLoS Computational Biology 3(9), 1819–1826 (Sep 2007)
13. Rizk, A., Batt, G., Fages, F., Soliman, S.: A general computational method for robustness analysis with applications to synthetic gene networks. Bioinformatics 12(25), il69–il78 (Jun 2009)
14. Rizk, A., Batt, G., Fages, F., Soliman, S.: Continuous valuations of temporal logic specifications with applications to parameter optimization and robustness measures. Theoretical Computer Science 412(26), 2827–2839 (2011)
15. Traynard, P., Fages, F., Soliman, S.: Trace simplifications preserving temporal logic formulae with case study in a coupled model of the cell cycle and the circadian clock. In: CMSB'14: Proceedings of the twelth international conference on Computational Methods in Systems Biology. pp. 114–128. No. 8859 in Lecture Notes in BioInformatics, Springer-Verlag (Sep 2014)
16. Traynard, P., Feillet, C., Soliman, S., Delaunay, F., Fages, F.: Model-based investigation of the circadian clock and cell cycle coupling in mouse embryonic fibroblasts: Prediction of reverb-alpha up-regulation during mitosis. Biosystems 149, 59–69 (Nov 2016)
17. Tyson, J.J.: Modeling the cell division cycle: cdc2 and cyclin interactions. Proceedings of the National Academy of Sciences 88(16), 7328–7332 (Aug 1991)

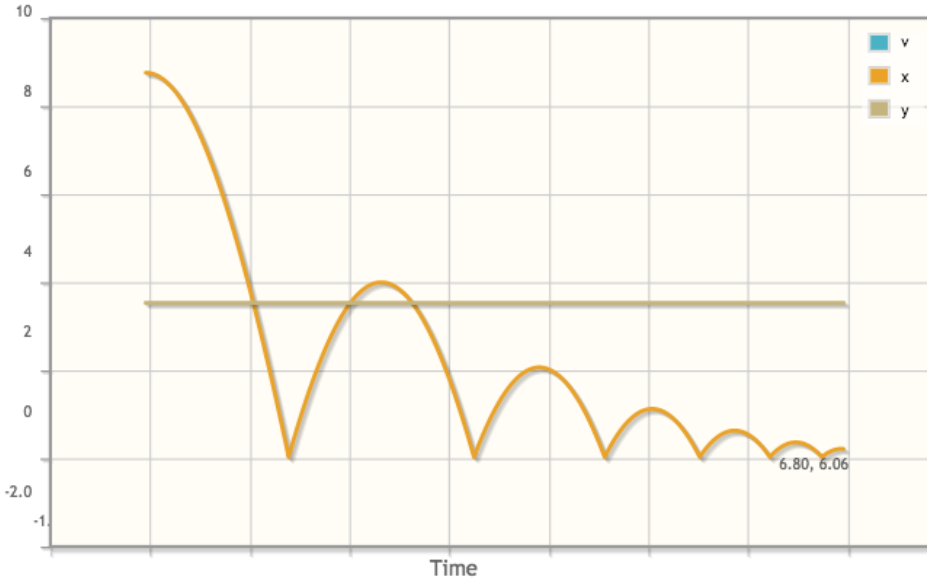
Appendix A Bouncing Ball Example

The bouncing ball with air friction is a classical example of the hybrid system community. It can be modeled in BIOCHAM by one first reaction with mass action (MA) law kinetics where the velocity catalyzes the position, a second reaction where the gravity force catalyzes the velocity, a third reaction for air friction where the velocity autocalyzes its degradation, and one event for elastic bouncing by reversing the sign of one parameter:

```
biocham: load(ball.bc).

biocham: list_model.
MA(c)for _=[v]=>x.
g/c for v=>_.
MA(D*c)for v=[v]=>_.
present(x,x0).
present(y,3.5).
parameter(
  x0 = 8.725,
  D = -0.05,
  K = 0.75,
  g = 9.8,
  c = 1.0
).
add_event(x<=0, c = -K*c).

biocham: list_ode.
[0] d(v)/dt= -1*g/c-D*c*v^2
[1] d(x)/dt=c*v
[2] d(y)/dt=0
```

In a hybrid system numerical simulations are relatively time consuming due to the necessity of precisely checking the satisfaction time of events

```
biocham: numerical_simulation. plot.
```



Model robustness to parameter perturbations is estimated by sampling the parameter space. This is time consuming since it involves making one simulation for each parameter set.

The FO-LTL formula used in this example defines the amplitude range of x in the domain of the free variable h The amplitude 3.96 exceeds here the objective of 3.5, the satisfaction degree is thus greater than one showing some formula robustness.

However the model robustness w.r.t. (default) parameter perturbations is below one (0.88) showing that some parameter perturbations destroy the amplitude objective.

```
biocham: validity_domain(F(x < h /\ F(x > h))).
h<3.96646/\h> -1.0e-6

biocham: satisfaction_degree(F(x < h /\ F(x > h)), [h -> 3.5]).
1.466440

biocham: robustness(F(x < h /\ F(x > h)), [x0, K, D], [h -> 3.5]).

Time: 158.027 s
Robustness degree: 0.88414
```

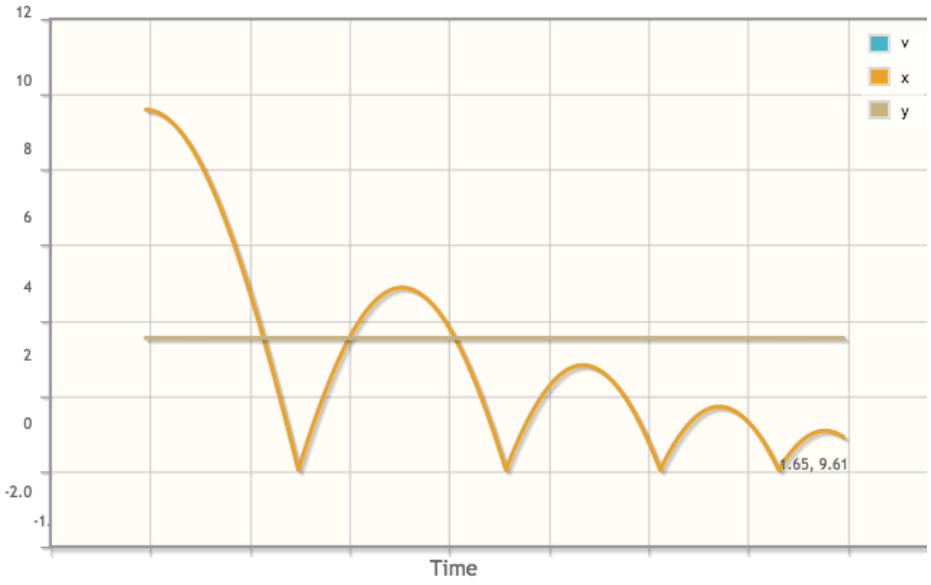
With a satisfaction degree greater than 1 the formula is already satisfied with some margin. However, formula robustness can be further optimized by parameter optimization with formula robustness as objective function with no extra cost and much faster computation time than for estimating model robustness.

It gives an amplitude of 4.84 which obviously improves the satisfaction degree of the formula.

This is shown to also improve the model robustness (0.93) to parameter perturbations as expected in many examples.

```
biocham: search_parameters(
            F(x < h /\ F(x > h)),
            [5 <= x0 <= 10, 0.5 <= K <= 1.0 , -0.1 <= D <= 0.0],
            [h -> 3.5], cmaes_stop_fitness: -0.5).

Time: 27.491 s
Stopping reason: Fitness: function value -5.72e-01 <= stopFitness
Best satisfaction degree: 2.339136
[0] parameter(x0=9.555198250463423)
[1] parameter(K=0.8014006370116868)
[2] parameter(D= -0.058468364144163905)
```

```
biocham: numerical_simulation. plot.
```



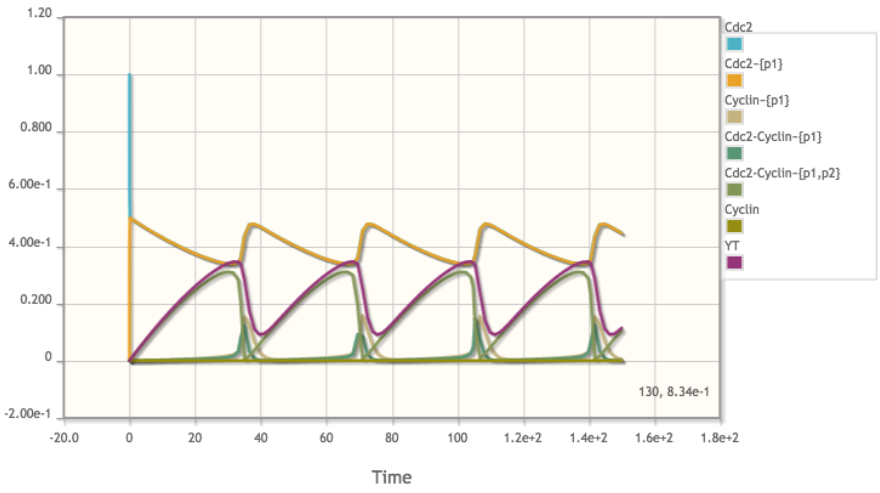

```
biocham: validity_domain(F(x < h /\ F(x > h))).
h<4.83916/\h> -1.0e-6

biocham: satisfaction_degree(F(x < h /\ F(x > h)), [h -> 3.5]).
2.339120
```

```
biocham: robustness(F(x < h /\ F(x > h)), [x0, K, D], [h -> 3.5]).

Time: 100.736 s
Robustness degree: 0.930547
```

Appendix B Cell Cycle Example

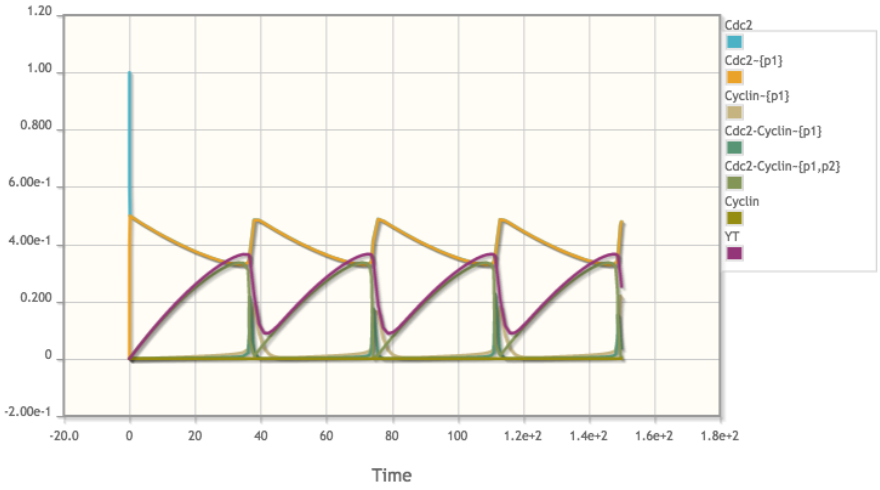
We consider here the simple model of cell division cycle of Tyson [17] and the search of parameter values optimizing the robustness of some amplitude property of the oscillations.

```
biocham: load(library:examples/cell_cycle/Tyson_1991.bc).
biocham: option(time:150).
biocham: parameter(k8=100).
biocham: numerical_simulation. plot.
```




```
biocham: satisfaction_degree(exists(max, exists(min,
             F(Cdc2-Cyclin~{p1} > max /\ F(Cdc2-Cyclin~{p1} < min /\
             F(Cdc2-Cyclin~{p1} > max /\ F(Cdc2-Cyclin~{p1} < min))))
             /\ max - min > amp)), [amp -> 0.19]).
0.974450

biocham: seed(0).
biocham: robustness(exists(max, exists(min,
             F(Cdc2-Cyclin~{p1} > max /\ F(Cdc2-Cyclin~{p1} < min /\
             F(Cdc2-Cyclin~{p1} > max /\ F(Cdc2-Cyclin~{p1} < min))))
             /\ max - min > amp)), [k4, k6], [amp -> 0.19]).

Time: 7.494 s
```

```
Robustness degree: 0.978333

biocham: parameter(k4=550, k6=2).

biocham: numerical_simulation. plot.
```




```
biocham: satisfaction_degree(exists(max, exists(min,
             F(Cdc2-Cyclin~{p1} > max /\ F(Cdc2-Cyclin~{p1} < min /\
             F(Cdc2-Cyclin~{p1} > max /\ F(Cdc2-Cyclin~{p1} < min))))
             /\ max - min > amp)), [amp -> 0.19]).
1.009960

biocham: robustness(exists(max, exists(min, F(Cdc2-Cyclin~{p1} > max /\
             F(Cdc2-Cyclin~{p1} < min /\ F(Cdc2-Cyclin~{p1} > max /\
             F(Cdc2-Cyclin~{p1} < min)))) /\ max - min > amp)),
             [k4, k6], [amp -> 0.19], robustness_coeff_var: 0.1).

Time: 10.956 s
Robustness degree: 0.996442
```

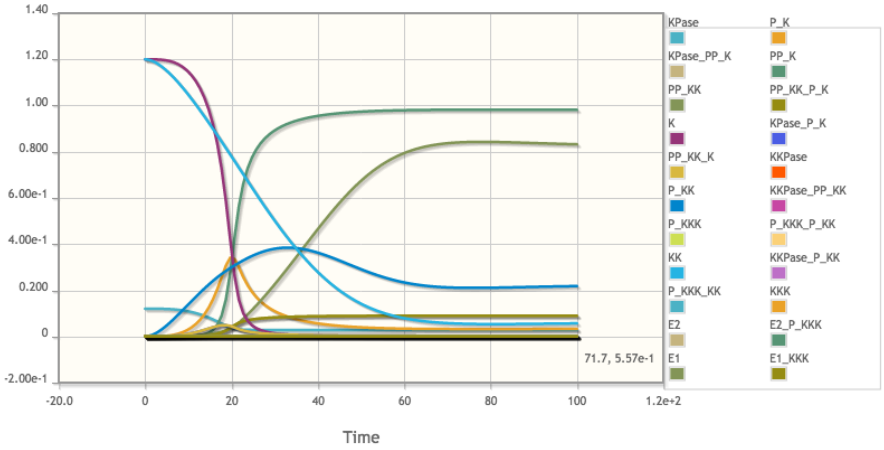
Appendix C MAPK Signaling Network Example

Here we consider the model of Huang and Ferrell of the Mitogen Activated Protein Kinase (MAPK) signaling network [10]. We illustrate the search of parameter values exhibiting oscillations with some apmlitude properties and the optimization of the robustness of those properties.

```
biocham: load(huang_ferrell.bc).
```

```
biocham: list_model.
MA(a1)for KKK+E1=>E1_KKK.
MA(d1)for E1_KKK=>KKK+E1.
MA(k1)for E1_KKK=>E1+P_KKK.
MA(a2)for P_KKK+E2=>E2_P_KKK.
MA(d2)for E2_P_KKK=>P_KKK+E2.
MA(k2)for E2_P_KKK=>E2+KKK.
MA(a3)for KK+P_KKK=>P_KKK_KK.
MA(d3)for P_KKK_KK=>KK+P_KKK.
MA(k3)for P_KKK_KK=>P_KK+P_KKK.
MA(a4)for P_KK+KKPase=>KKPase_P_KK.
MA(d4)for KKPase_P_KK=>P_KK+KKPase.
MA(k4)for KKPase_P_KK=>KK+KKPase.
MA(a5)for P_KK+P_KKK=>P_KKK_P_KK.
MA(d5)for P_KKK_P_KK=>P_KK+P_KKK.
MA(k5)for P_KKK_P_KK=>PP_KK+P_KKK.
MA(a6)for PP_KK+KKPase=>KKPase_PP_KK.
MA(d6)for KKPase_PP_KK=>PP_KK+KKPase.
MA(k6)for KKPase_PP_KK=>P_KK+KKPase.
MA(a7)for K+PP_KK=>PP_KK_K.
MA(d7)for PP_KK_K=>K+PP_KK.
MA(k7)for PP_KK_K=>P_K+PP_KK.
MA(a8)for P_K+KPase=>KPase_P_K.
MA(d8)for KPase_P_K=>P_K+KPase.
MA(k8)for KPase_P_K=>K+KPase.
MA(a9)for P_K+PP_KK=>PP_KK_P_K.
MA(d9)for PP_KK_P_K=>P_K+PP_KK.
MA(k9)for PP_KK_P_K=>PP_KK+PP_K.
MA(a10)for PP_K+KPase=>KPase_PP_K.
MA(d10)for KPase_PP_K=>PP_K+KPase.
MA(k10)for KPase_PP_K=>P_K+KPase.
present(E1,E1_tot).
present(E2,E2_tot).
present(KKK,KKK_tot).
present(KK,KK_tot).
present(K,K_tot).
present(KPase,KPase_tot).
present(KKPase,KKPase_tot).
parameter(
  a1 = 1000.0,
  d1 = 150.0,
  k1 = 150.0,
  a2 = 1000.0,
  d2 = 150.0,
  k2 = 150.0,
```

```
  a3 = 1000.0,
  d3 = 150.0,
  k3 = 150.0,
  a4 = 1000.0,
  d4 = 150.0,
  k4 = 150.0,
  a5 = 1000.0,
  d5 = 150.0,
  k5 = 150.0,
  a6 = 1000.0,
  d6 = 150.0,
  k6 = 150.0,
  a7 = 1000.0,
  d7 = 150.0,
  k7 = 150.0,
  a8 = 1000.0,
  d8 = 150.0,
  k8 = 150.0,
  a9 = 1000.0,
  d9 = 150.0,
  k9 = 150.0,
  a10 = 1000.0,
  d10 = 150.0,
  k10 = 150.0,
  KKK_tot = 0.003,
  KK_tot = 1.2,
  K_tot = 1.2,
  E2_tot = 0.0003,
  KKPase_tot = 0.0003,
  KPase_tot = 0.12,
  E1_tot = 3.0e-5
).

biocham: option(time: 100).

biocham: numerical_simulation. plot.
```

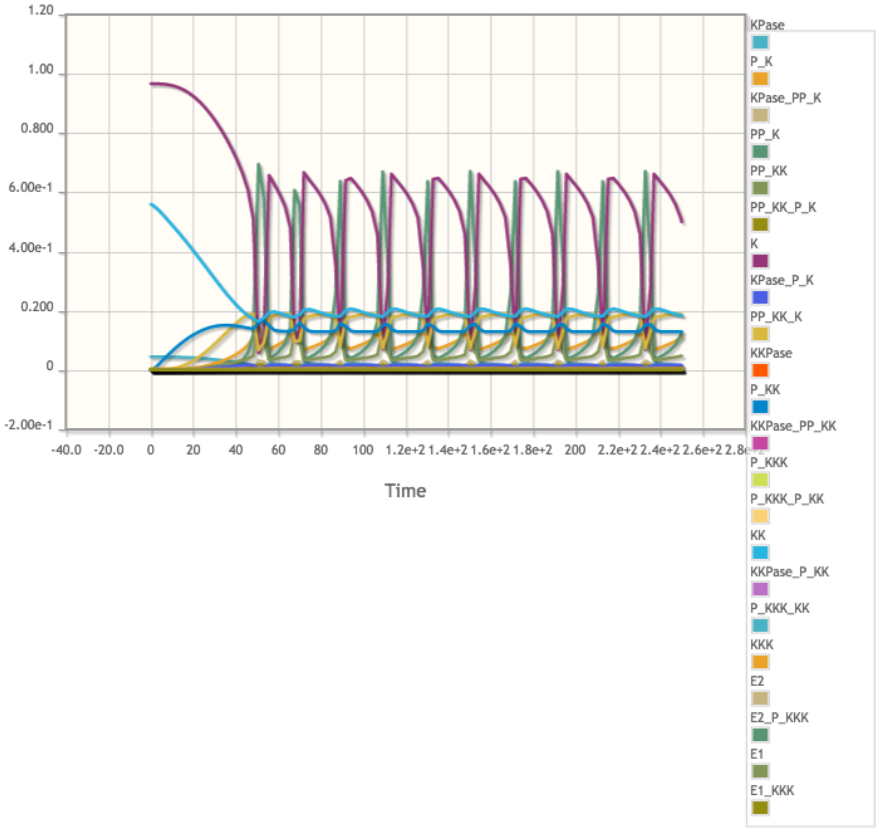


Let us search for parameter values exhibiting oscillations, using the same range of values as Qiao et al. [12] as given in their Table S1. We also add a range for $E1_{tot}$ which was not searched by Qiao et al. but used instead as parameter in a bifurcation diagram.

```
biocham: timeout 600
Timeout set to 600

biocham: seed(0).
biocham: search_parameters(
   exists(up, F(PP_K >= up /\ F(PP_K <= up - amplitude /\ F(PP_K >= up)))),
   [
      6.0e-4 <= KKK_tot <= 1.5e-2,
      0.24 <= KK_tot <= 6,
      0.24 <= K_tot <= 6,
      6.0e-5 <= E2_tot <= 1.5e-3,
      6.0e-5 <= KKPase_tot <= 1.5e-3,
      2.4e-2 <= KPase_tot <= 0.6,
      200 <= a1 <= 5000,
      200 <= a2 <= 5000,
      200 <= a3 <= 5000,
      200 <= a4 <= 5000,
      200 <= a5 <= 5000,
      200 <= a6 <= 5000,
      200 <= a7 <= 5000,
      200 <= a8 <= 5000,
      200 <= a9 <= 5000,
      200 <= a10 <= 5000,
      30 <= d1 <= 750,
      30 <= d2 <= 750,
      30 <= d3 <= 750,
```

```
      30 <= d4 <= 750,
      30 <= d5 <= 750,
      30 <= d6 <= 750,
      30 <= d7 <= 750,
      30 <= d8 <= 750,
      30 <= d9 <= 750,
      30 <= d10 <= 750,
      30 <= k1 <= 750,
      30 <= k2 <= 750,
      30 <= k3 <= 750,
      30 <= k4 <= 750,
      30 <= k5 <= 750,
      30 <= k6 <= 750,
      30 <= k7 <= 750,
      30 <= k8 <= 750,
      30 <= k9 <= 750,
      30 <= k10 <= 750,
      1.0e-8 <= E1_tot <= 1.0e-4
   ],
   [amplitude -> 0.5],
   cmaes_log_normal: yes
).

Time: 514.492 s
Stopping reason: Fitness: function value -1.16e-01 <= stopFitness (1.00e-04)
Best satisfaction degree: 1.131039
[0] parameter(KKK_tot=0.005219419568782974)
[1] parameter(KK_tot=0.559412542917763)
[2] parameter(K_tot=0.9666273387435539)
[3] parameter(E2_tot=0.0004921189852601114)
[4] parameter(KKPase_tot=0.0006001092025538948)
[5] parameter(KPase_tot=0.04465600529486992)
[6] parameter(a1=749.1374184681108)
[7] parameter(a2=1425.4851100575816)
[8] parameter(a3=4405.483511310206)
[9] parameter(a4=1787.822432703679)
[10] parameter(a5=913.8138837808307)
[11] parameter(a6=1077.2238852170678)
[12] parameter(a7=1518.449110421418)
[13] parameter(a8=4657.351620325106)
[14] parameter(a9=982.6905013194673)
[15] parameter(a10=2049.2558154978096)
[16] parameter(d1=111.03155089393204)
[17] parameter(d2=60.17987177300847)
[18] parameter(d3=43.700829649966245)
```

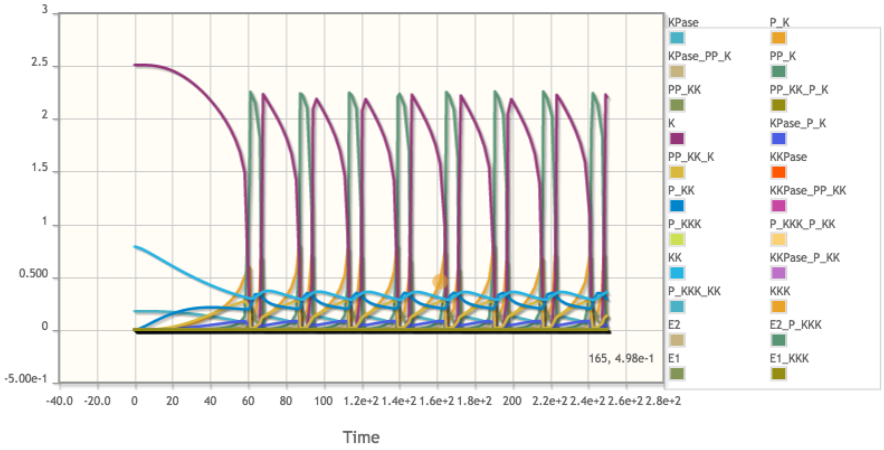
```
[19] parameter(d4=377.2918369043323)
[20] parameter(d5=78.36000105258077)
[21] parameter(d6=347.6471162031489)
[22] parameter(d7=159.8293350890864)
[23] parameter(d8=73.81187639431515)
[24] parameter(d9=272.7696740285378)
[25] parameter(d10=49.08766017222723)
[26] parameter(k1=195.06871032844694)
[27] parameter(k2=119.97711682617926)
[28] parameter(k3=85.71227545201387)
[29] parameter(k4=659.8294248420139)
[30] parameter(k5=478.3419887728013)
[31] parameter(k6=359.0640863995503)
[32] parameter(k7=35.460554156809266)
[33] parameter(k8=391.76536853346914)
[34] parameter(k9=430.2858803359159)
[35] parameter(k10=256.23653403655294)
[36] parameter(E1_tot=2.8989794237322926e-5)

biocham: numerical_simulation(time: 250). plot.
```




```
biocham: seed(0).
biocham: robustness(
    exists(up, F(PP_K >= up /\ F(PP_K <= up - amplitude /\ F(PP_K >= up)))),
    [
        a1, a2, a3, a4, a5, a6, a7, a8, a9, a10,
        d1, d2, d3, d4, d5, d6, d7, d8, d9, d10,
        k1, k2, k3, k4, k5, k6, k7, k8, k9, k10
    ],
    [amplitude -> 0.5]).

Time: 10.38 s
Robustness degree: 0.837798

biocham: seed(0).
biocham: search_parameters(
   exists(up, F(PP_K >= up /\ F(PP_K <= up - amplitude /\ F(PP_K >= up)))),
   [
      6.0e-4 <= KKK_tot <= 1.5e-2,
```

```
      0.24 <= KK_tot <= 6,
      0.24 <= K_tot <= 6,
      6.0e-5 <= E2_tot <= 1.5e-3,
      6.0e-5 <= KKPase_tot <= 1.5e-3,
      2.4e-2 <= KPase_tot <= 0.6,
      200 <= a1 <= 5000,
      200 <= a2 <= 5000,
      200 <= a3 <= 5000,
      200 <= a4 <= 5000,
      200 <= a5 <= 5000,
      200 <= a6 <= 5000,
      200 <= a7 <= 5000,
      200 <= a8 <= 5000,
      200 <= a9 <= 5000,
      200 <= a10 <= 5000,
      30 <= d1 <= 750,
      30 <= d2 <= 750,
      30 <= d3 <= 750,
      30 <= d4 <= 750,
      30 <= d5 <= 750,
      30 <= d6 <= 750,
      30 <= d7 <= 750,
      30 <= d8 <= 750,
      30 <= d9 <= 750,
      30 <= d10 <= 750,
      30 <= k1 <= 750,
      30 <= k2 <= 750,
      30 <= k3 <= 750,
      30 <= k4 <= 750,
      30 <= k5 <= 750,
      30 <= k6 <= 750,
      30 <= k7 <= 750,
      30 <= k8 <= 750,
      30 <= k9 <= 750,
      30 <= k10 <= 750,
      1.0e-8 <= E1_tot <= 1.0e-4
   ],
   [amplitude -> 0.5],
   cmaes_log_normal: yes,
   cmaes_stop_fitness: -0.15
).

Time: 471.696 s
Stopping reason: Fitness: function value -6.34e-01 <= stopFitness (-1.50e-01)
Best satisfaction degree: 2.732226
```

```
[0] parameter(KKK_tot=0.0060507291510886125)
[1] parameter(KK_tot=0.7875358601949335)
[2] parameter(K_tot=2.510549952091971)
[3] parameter(E2_tot=0.00010491974393664447)
[4] parameter(KKPase_tot=0.0009464415540164032)
[5] parameter(KPase_tot=0.17733172243018358)
[6] parameter(a1=449.33724955834504)
[7] parameter(a2=1763.108020587385)
[8] parameter(a3=1778.8986750288468)
[9] parameter(a4=740.4823792799979)
[10] parameter(a5=927.8977619477777)
[11] parameter(a6=1427.793075587396)
[12] parameter(a7=2016.450839358622)
[13] parameter(a8=772.4947746518594)
[14] parameter(a9=693.7324782560232)
[15] parameter(a10=2295.2367896275964)
[16] parameter(d1=86.61481481015618)
[17] parameter(d2=139.3843547761347)
[18] parameter(d3=156.6366644888986)
[19] parameter(d4=55.04542633721323)
[20] parameter(d5=99.37085045832148)
[21] parameter(d6=73.50787251962154)
[22] parameter(d7=32.023376170744086)
[23] parameter(d8=187.02659561104193)
[24] parameter(d9=191.35504930722655)
[25] parameter(d10=596.1181951904716)
[26] parameter(k1=81.71448150522714)
[27] parameter(k2=718.4001911364844)
[28] parameter(k3=266.3789179550008)
[29] parameter(k4=106.44352863825837)
[30] parameter(k5=57.53060904576804)
[31] parameter(k6=373.06203392951795)
[32] parameter(k7=32.7900781193166)
[33] parameter(k8=104.5069465073101)
[34] parameter(k9=345.14862441826324)
[35] parameter(k10=43.87097095054329)
[36] parameter(E1_tot=1.8850158833237074e-5)

biocham: numerical_simulation(time: 250). plot.
```




```
biocham: seed(0).
biocham: robustness(
    exists(up, F(PP_K >= up /\ F(PP_K <= up - amplitude /\ F(PP_K >= up)))),
    [
        a1, a2, a3, a4, a5, a6, a7, a8, a9, a10,
        d1, d2, d3, d4, d5, d6, d7, d8, d9, d10,
        k1, k2, k3, k4, k5, k6, k7, k8, k9, k10
    ],
    [amplitude -> 0.5]).

Time: 5.165 s
Robustness degree: 0.866939
```