Quantitative Temporal Logics for Systems Biology

Towards Systems Biology

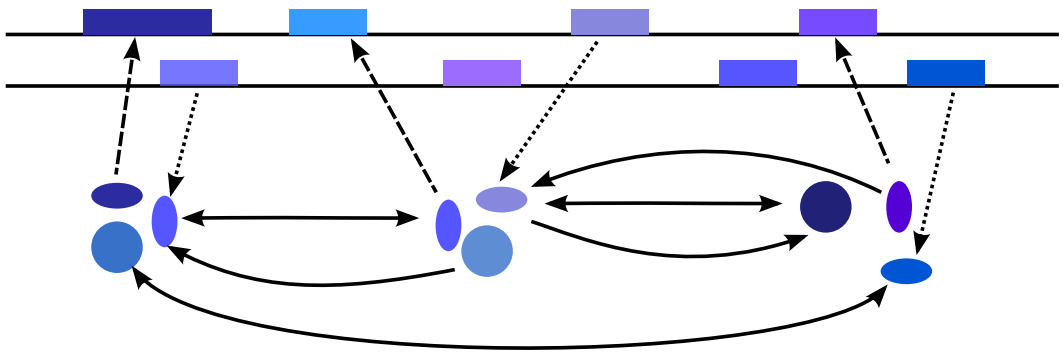
Alexandre Donzé

Laboratoire Verimag, Grenoble

June 1st, 2011

Biological networks

- Understanding a biological process through interactions between its elements
- Biological networks represents metabolism, gene regulation, signal transduction, protein interactions, etc



genes network

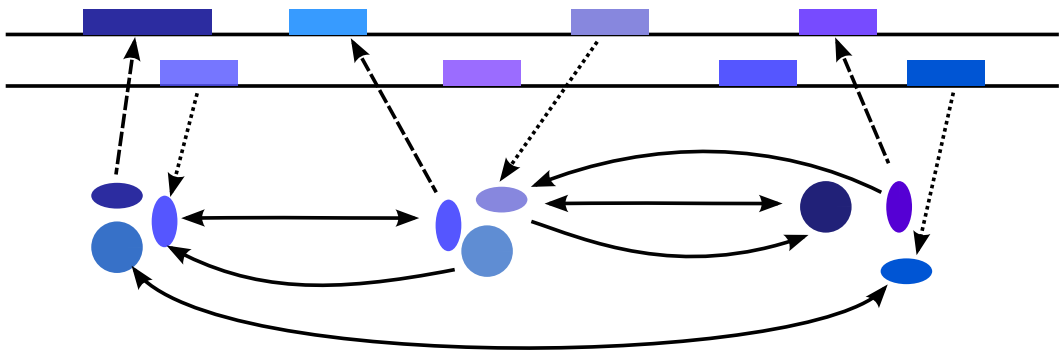
proteins network

Formal methods: rigorous and automatic analysis

- Formalizing biological hypotheses and test them *in silico*
- Infer new properties and observe them *in vivo*

Biological networks

- Understanding a biological process through interactions between its elements
- Biological networks represents metabolism, gene regulation, signal transduction, protein interactions, etc



genes network

proteins network

Formal methods: rigorous and automatic analysis

- Formalizing biological hypotheses and test them *in silico*
- Infer new properties and observe them *in vivo*

Models for biological networks

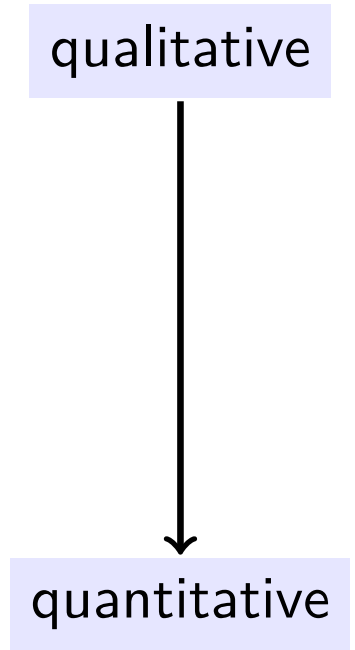
Interaction Graphs

Petri Nets

Flux based models

Thomas networks

Differential equations,
Hybrid systems



A number of formal methods exist for qualitative models but only a few apply for quantitative models

This work is concerned with this level

Models for biological networks

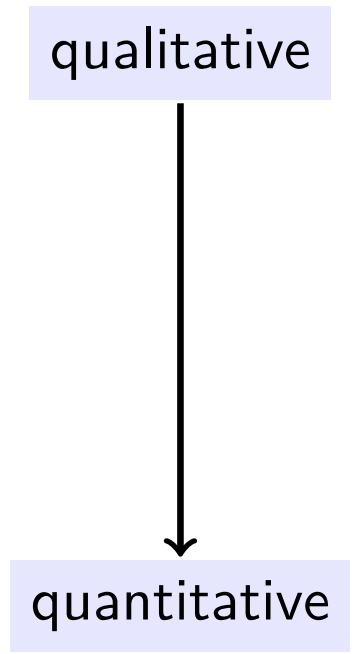
Interaction Graphs

Petri Nets

Flux based models

Thomas networks

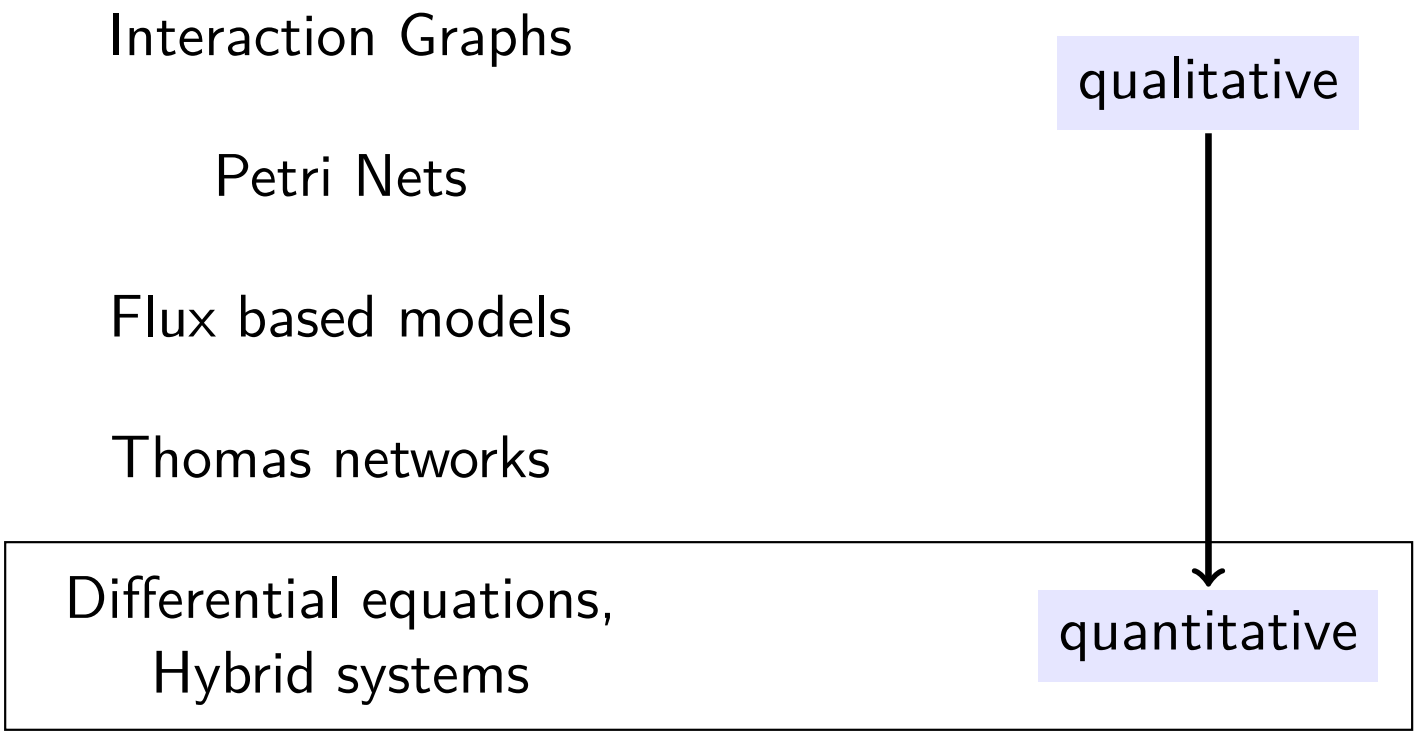
Differential equations,
Hybrid systems



A number of formal methods exist for qualitative models but only a few apply for quantitative models

This work is concerned with this level

Models for biological networks



A number of formal methods exist for qualitative models but only a few apply for quantitative models

This work is concerned with this level

Formal methods and verification

Model-Checking: about *proving* correctness.

To *prove* correctness, we need:

- a model, describing the systems behaviors
- a specification language to describe *desired* (good) and *unwanted* (bad) properties

  Coffee machine example:
  - a good property is: *If I insert a coin and push "coffee", I get coffee*
  - a bad one: *I get a free (and no change)*

- some procedure to decide whether *all* the behaviors satisfies *all* the good properties and *none* of the bad ones

Formal methods and verification

Model-Checking: about *proving* correctness.

To *prove* correctness, we need:

- a model, describing the systems behaviors
- a specification language to describe *desired* (good) and *unwanted* (bad) properties

  Coffee machine example:
  - a good property is: *If I insert a coin and push "coffee", I get coffee*
  - a bad one: *I get a tea (and no change)*

- some procedure to decide whether *all* the behaviors satisfies *all* the good properties and *none* of the bad ones

Formal methods and verification

Model-Checking: about *proving* correctness.

To *prove* correctness, we need:

- a model, describing the systems behaviors
- a specification language to describe *desired* (good) and *unwanted* (bad) properties

  Coffee machine example:
  - a good property is: *If I insert a coin and push "coffee", I get coffee*
  - a bad one is: *I get a free (and no change)*
- some procedure to decide whether *all* the behaviors satisfies *all* the good properties and *none* of the bad ones

Formal methods and verification

Model-Checking: about *proving* correctness.

To *prove* correctness, we need:

- a model, describing the systems behaviors
- a specification language to describe *desired* (good) and *unwanted* (bad) properties

  Coffee machine example:

  - a good property is: *if I insert a coin and push 'coffee', I get coffee*
  - a bad one: *I get a tea (and no change)*

- some procedure to decide whether *all* the behaviors satisfies *all* the good properties and *none* of the bad ones

Formal methods and verification

Model-Checking: about *proving* correctness.

To *prove* correctness, we need:

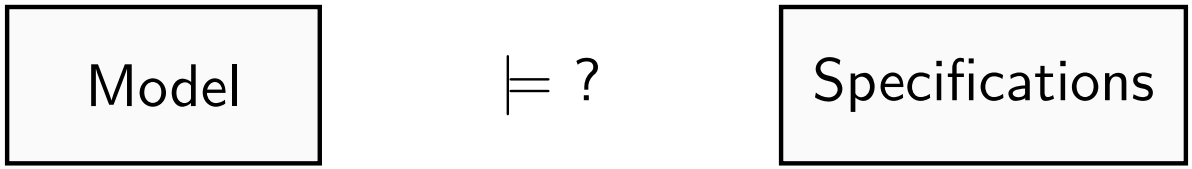
- a model, describing the systems behaviors
- a specification language to describe *desired* (good) and *unwanted* (bad) properties

  Coffee machine example:
  - a good property is: *if I insert a coin and push 'coffee', I get coffee*
  - a bad one: *I get a tea (and no change)*

- some procedure to decide whether *all* the behaviors satisfies *all* the good properties and *none* of the bad ones

Formal methods and verification

Model-Checking: about *proving* correctness.



To *prove* correctness, we need:

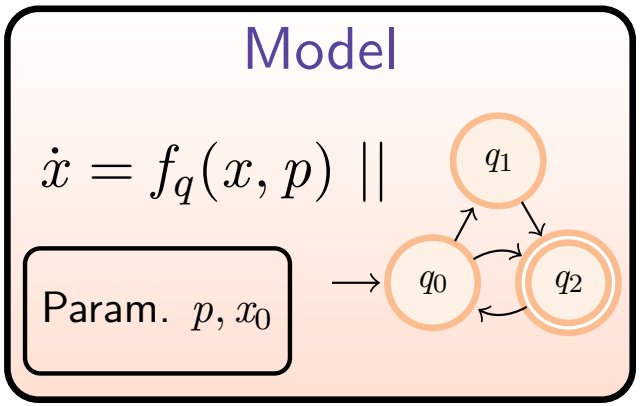
- a model, describing the systems behaviors
- a specification language to describe *desired* (good) and *unwanted* (bad) properties

  Coffee machine example:

  - a good property is: *if I insert a coin and push 'coffee', I get coffee*
  - a bad one: *I get a tea (and no change)*

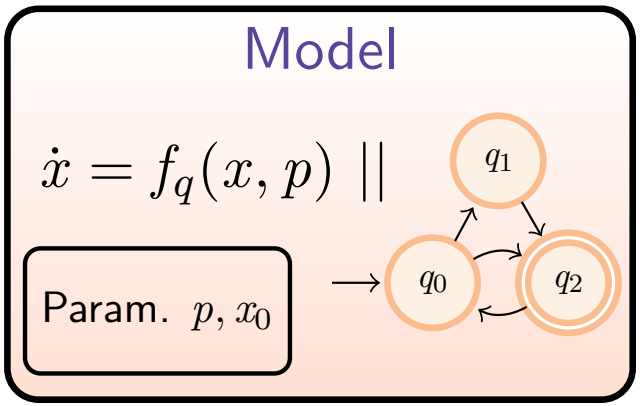
- some procedure to decide whether *all* the behaviors satisfies *all* the good properties and *none* of the bad ones

Hybrid dynamical systems



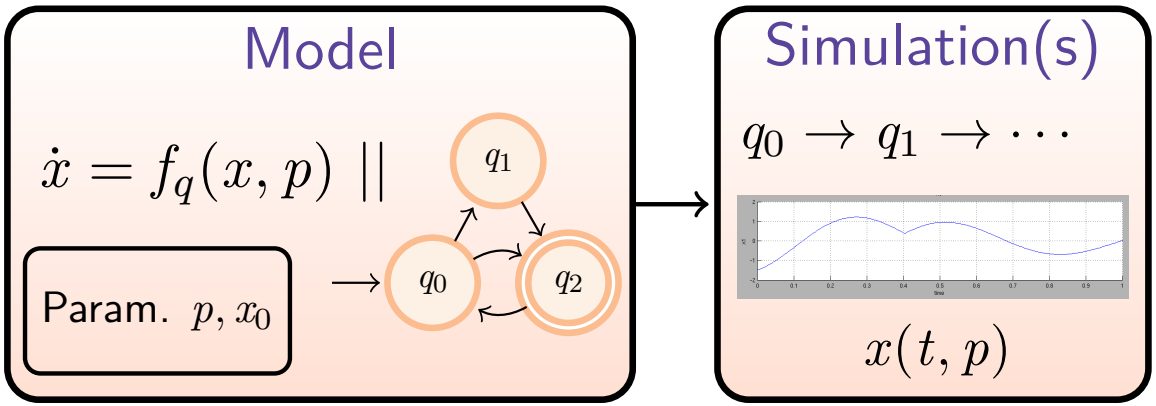
- *Proving* anything for hybrid systems is hard;
- Simulation, we can do;
- So, what can we tell from a (carefully selected) bunch of traces ?
- A lot depends on the *questions* one can ask to these simulations...

Hybrid dynamical systems



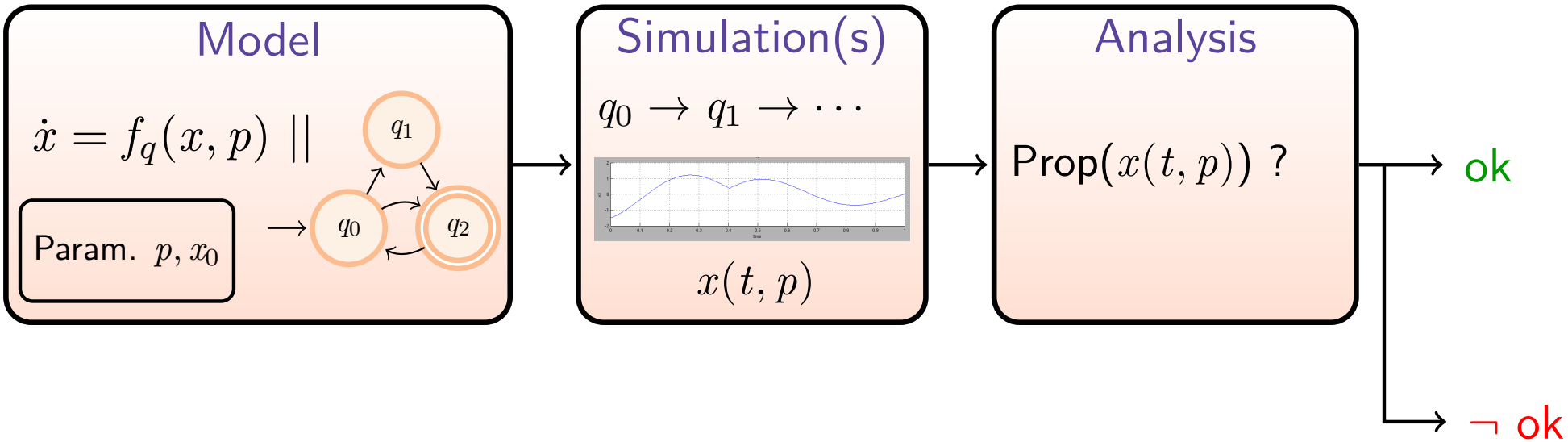
- *Proving* anything for hybrid systems is hard;
- Simulation, we can do;
- So, what can we tell from a (carefully selected) bunch of traces ?
- A lot depends on the *questions* one can ask to these simulations...

Hybrid dynamical systems



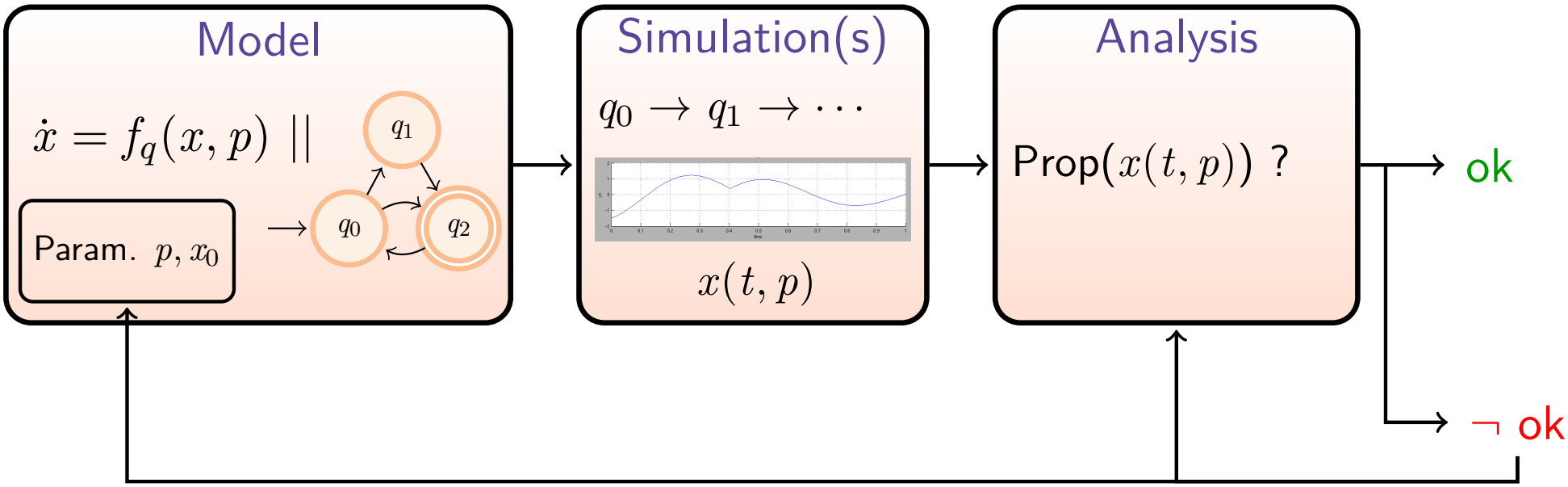
- *Proving* anything for hybrid systems is hard;
- Simulation, we can do;
- So, what can we tell from a (carefully selected) bunch of traces ?
- A lot depends on the *questions* one can ask to these simulations...

Hybrid dynamical systems



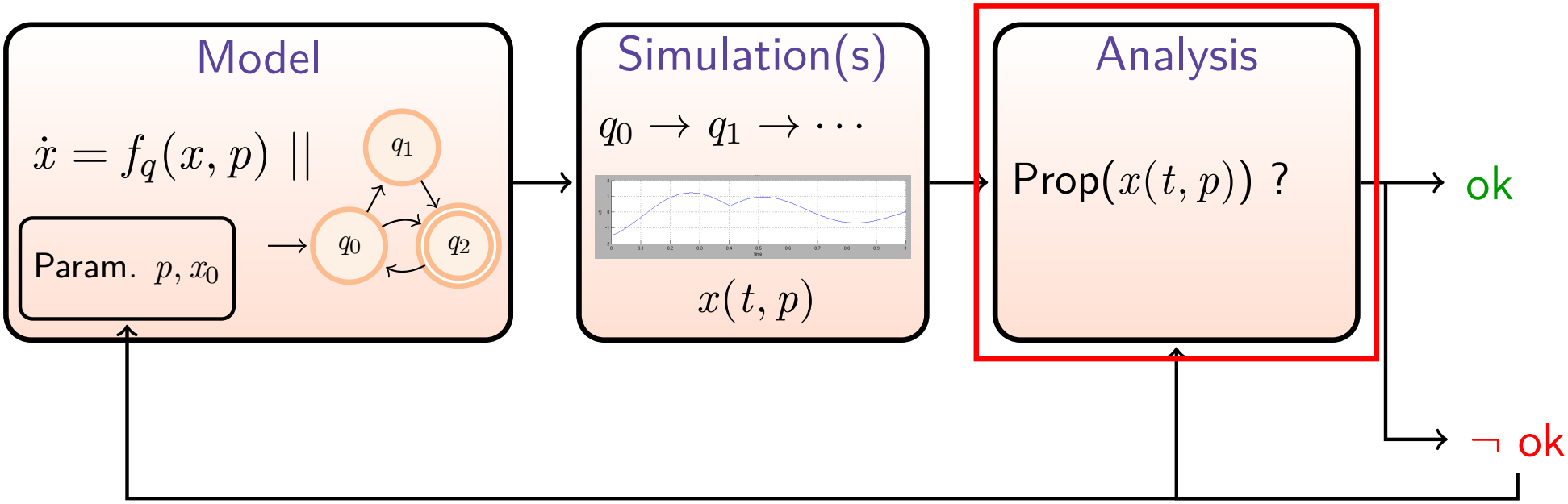
- *Proving* anything for hybrid systems is hard;
- Simulation, we can do;
- So, what can we tell from a (carefully selected) bunch of traces ?
- A lot depends on the *questions* one can ask to these simulations...

Hybrid dynamical systems



- *Proving* anything for hybrid systems is hard;
- Simulation, we can do;
- So, what can we tell from a (carefully selected) bunch of traces ?
- A lot depends on the *questions* one can ask to these simulations...

Hybrid dynamical systems

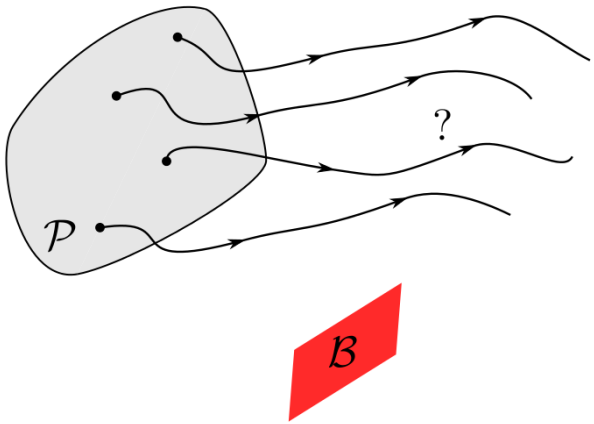


- *Proving* anything for hybrid systems is hard;
- Simulation, we can do;
- So, what can we tell from a (carefully selected) bunch of traces ?
- A lot depends on the *questions* one can ask to these simulations...

Outline

Safety verification

Define a set $\mathcal{P}$ of parameters p (init. cond. or param), each corresponding to one traj. and some forbidden region $\mathcal{B}$. How to *verify* that all traj. avoid $\mathcal{B}$?



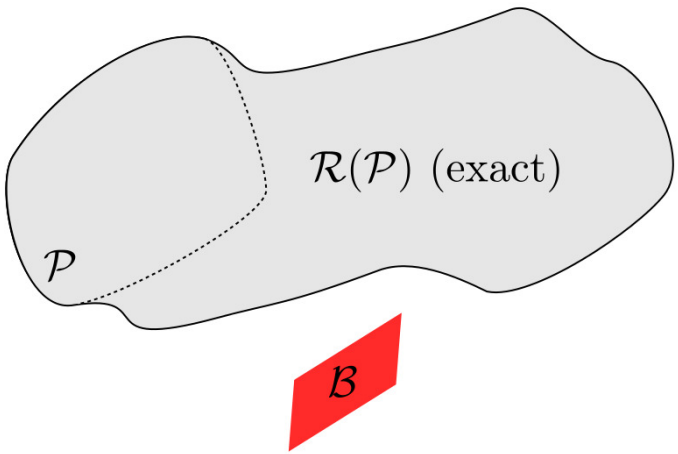
Reachability analysis

- Trying to compute the set containing *all* trajectories
- Using simple set representation
- Empty intersection with $\mathcal{B}$ proves safety

- Difficulties Spurious results in case of imprecise over-approximation + difficult for nonlinear system with more than a few continuous variables

Safety verification

Define a set $\mathcal{P}$ of parameters p (init. cond. or param), each corresponding to one traj. and some forbidden region $\mathcal{B}$. How to *verify* that all traj. avoid $\mathcal{B}$?



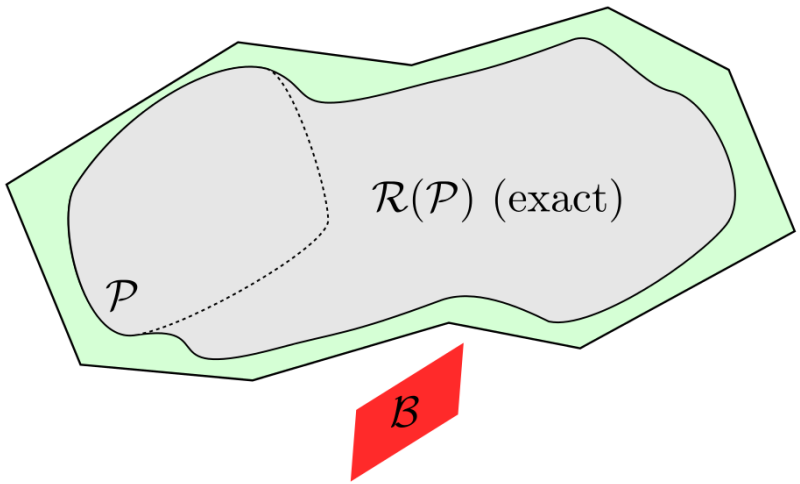
Reachability analysis

- Trying to compute the set containing *all* trajectories
- Using simple set representation
- Empty intersection with $\mathcal{B}$ proves safety

- Difficulties Spurious results in case of imprecise over-approximation + difficult for nonlinear system with more than a few continuous variables

Safety verification

Define a set $\mathcal{P}$ of parameters p (init. cond. or param), each corresponding to one traj. and some forbidden region $\mathcal{B}$. How to *verify* that all traj. avoid $\mathcal{B}$?



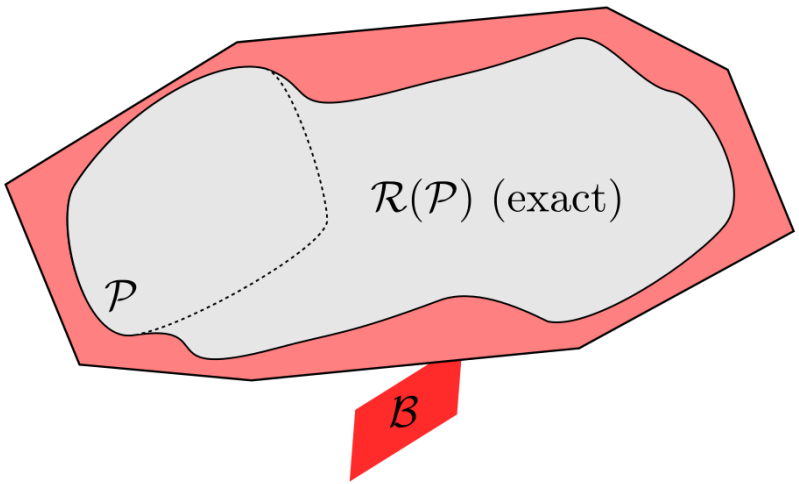
Reachability analysis

- Trying to compute the set containing *all* trajectories
- Using simple set representation
- Empty intersection with $\mathcal{B}$ proves safety

- Difficulties Spurious results in case of imprecise over-approximation + difficult for nonlinear system with more than a few continuous variables

Safety verification

Define a set $\mathcal{P}$ of parameters p (init. cond. or param), each corresponding to one traj. and some forbidden region $\mathcal{B}$. How to *verify* that all traj. avoid $\mathcal{B}$?

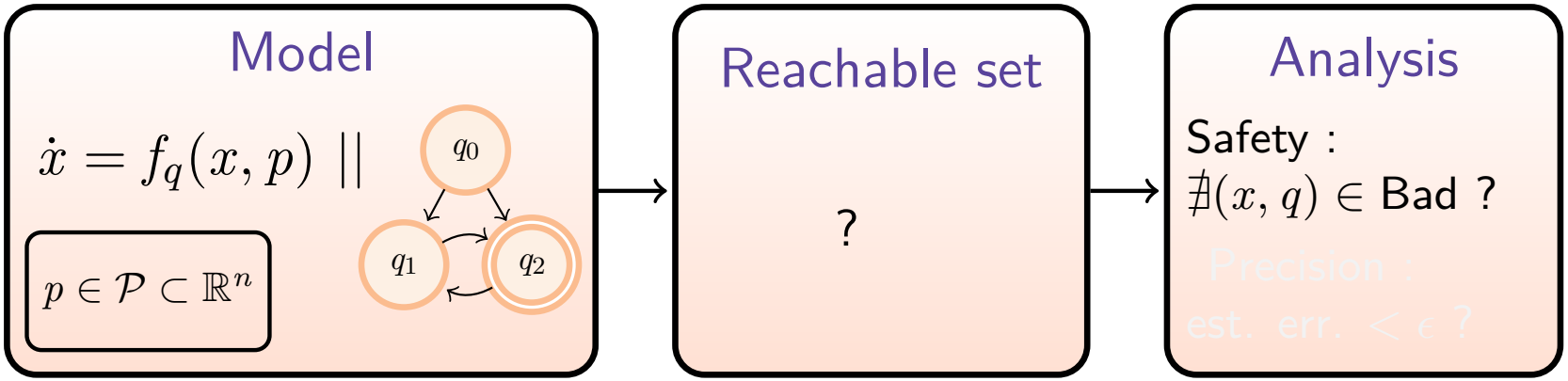


Reachability analysis

- Trying to compute the set containing *all* trajectories
- Using simple set representation
- Empty intersection with $\mathcal{B}$ proves safety

- Difficulties Spurious results in case of imprecise over-approximation + difficult for nonlinear system with more than a few continuous variables

Reachability analysis

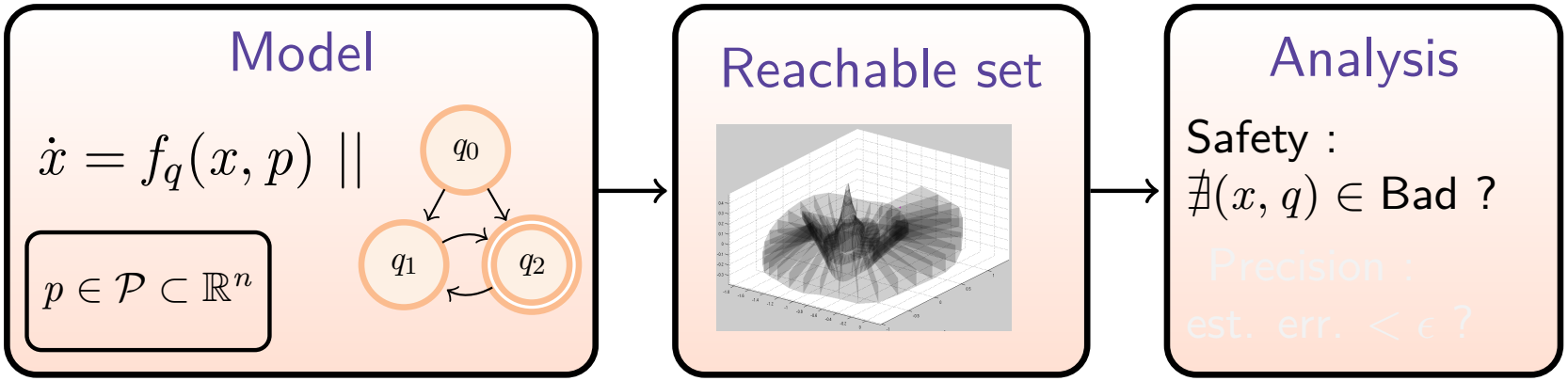


Computing $\mathcal{R}(\mathcal{P}) = \{(x, q) \mid \exists p \exists t \exists n,\ x(t, p) = x \wedge q_n = q\}$

Approach

- Approximate method based on simulation and local sensitivity analysis
- Numerical error estimate to control precision
- Hierarchical refinement of the parameter set

Reachability analysis

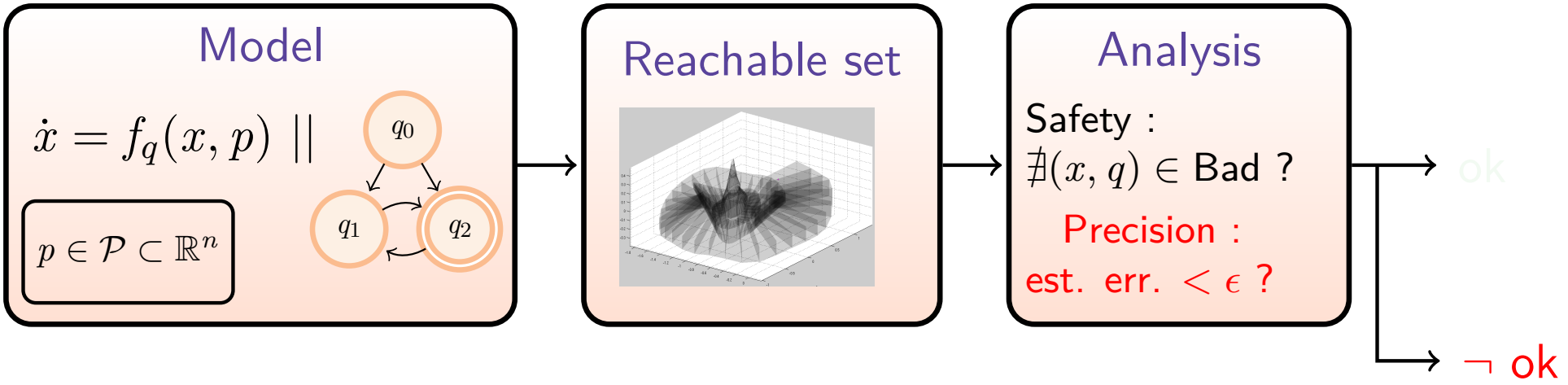


Computing $\mathcal{R}(\mathcal{P}) = \{(x, q) \mid \exists p \exists t \exists n,\ x(t, p) = x \wedge q_n = q\}$

Approach

- Approximate method based on simulation and local sensitivity analysis
- Numerical error estimate to control precision
- Hierarchical refinement of the parameter set

Reachability analysis

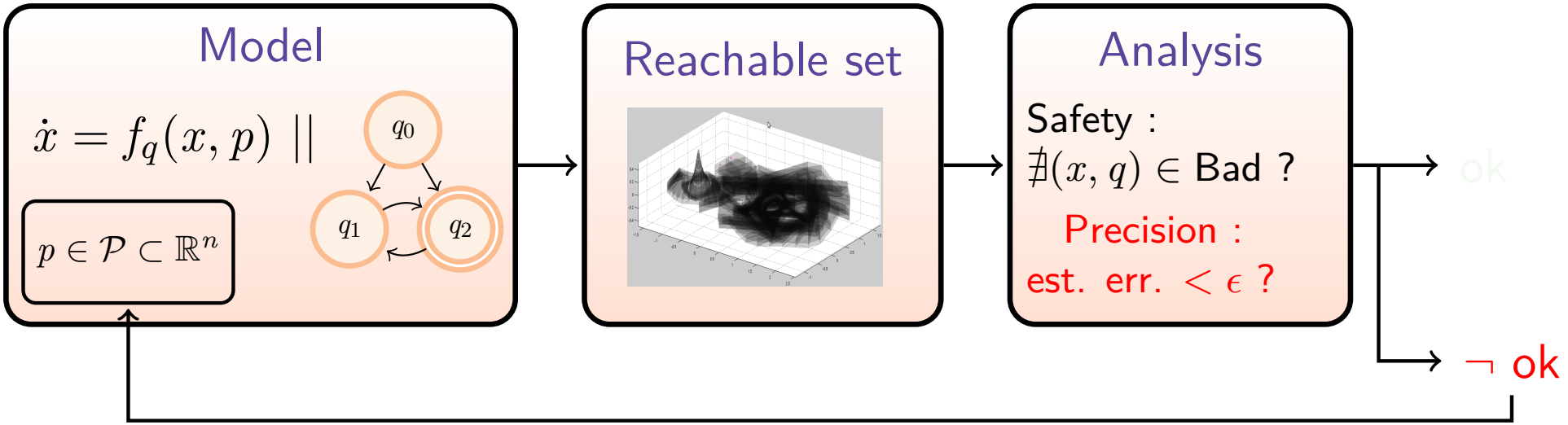


Computing $\mathcal{R}(\mathcal{P}) = \{(x, q) \mid \exists p \exists t \exists n,\ x(t, p) = x \wedge q_n = q\}$

Approach

- Approximate method based on simulation and local sensitivity analysis
- Numerical error estimate to control precision
- Hierarchical refinement of the parameter set

Reachability analysis

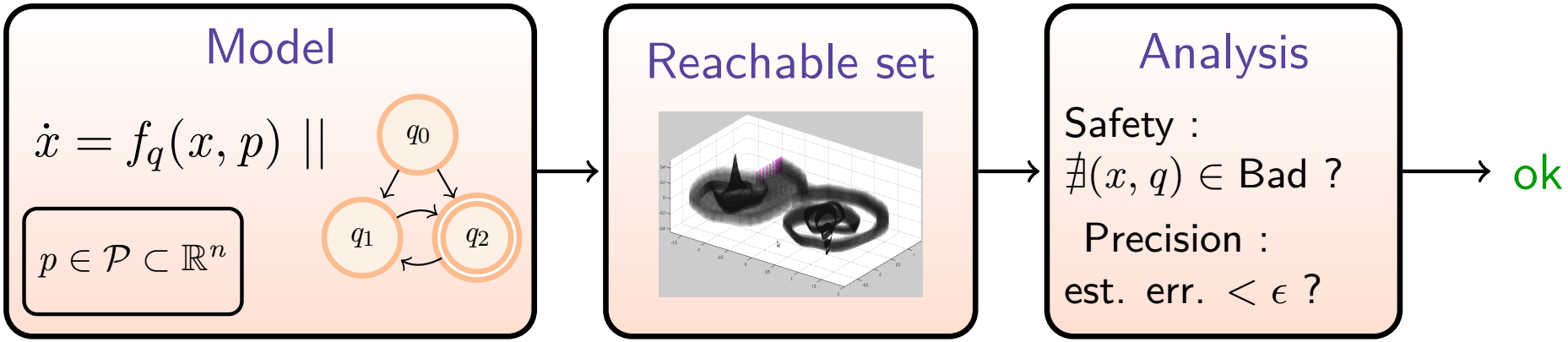


Computing $\mathcal{R}(\mathcal{P}) = \{(x, q) \mid \exists p \exists t \exists n,\ x(t, p) = x \wedge q_n = q\}$

Approach

- Approximate method based on simulation and local sensitivity analysis
- Numerical error estimate to control precision
- Hierarchical refinement of the parameter set

Reachability analysis

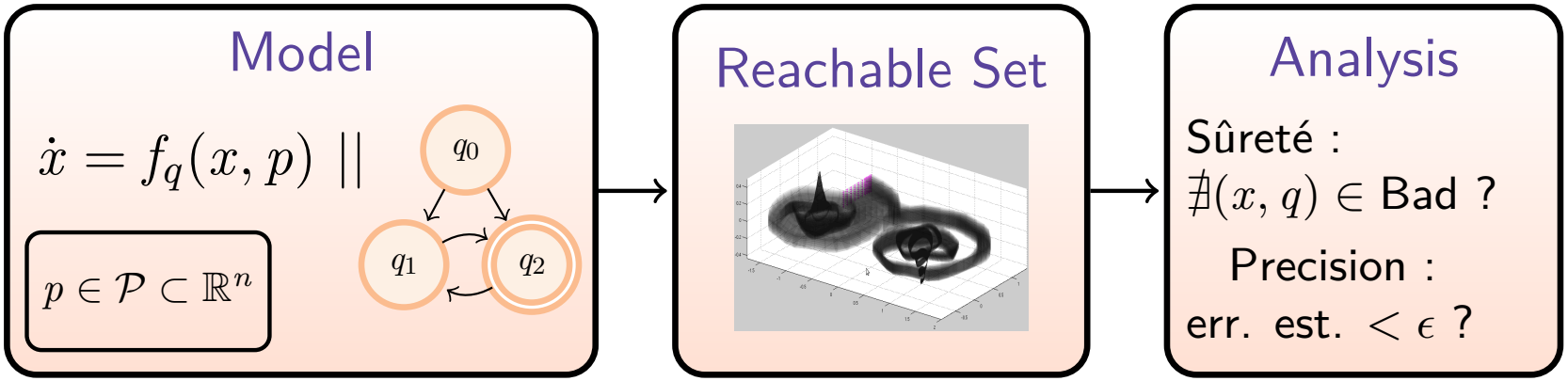


Computing $\mathcal{R}(\mathcal{P}) = \{(x, q) \mid \exists p \exists t \exists n,\ x(t, p) = x \wedge q_n = q\}$

Approach

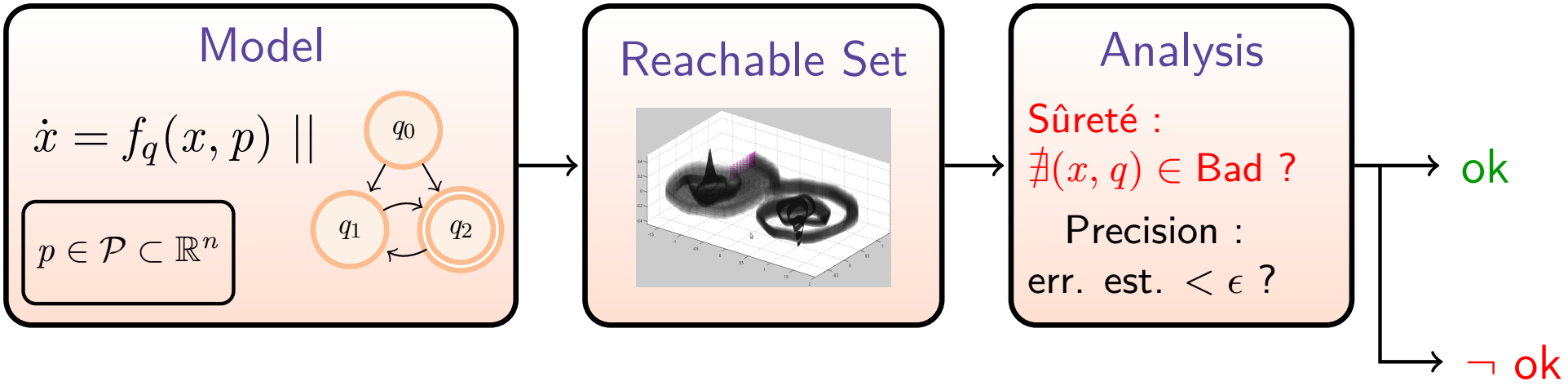
- Approximate method based on simulation and local sensitivity analysis
- Numerical error estimate to control precision
- Hierarchical refinement of the parameter set

Parameter synthesis



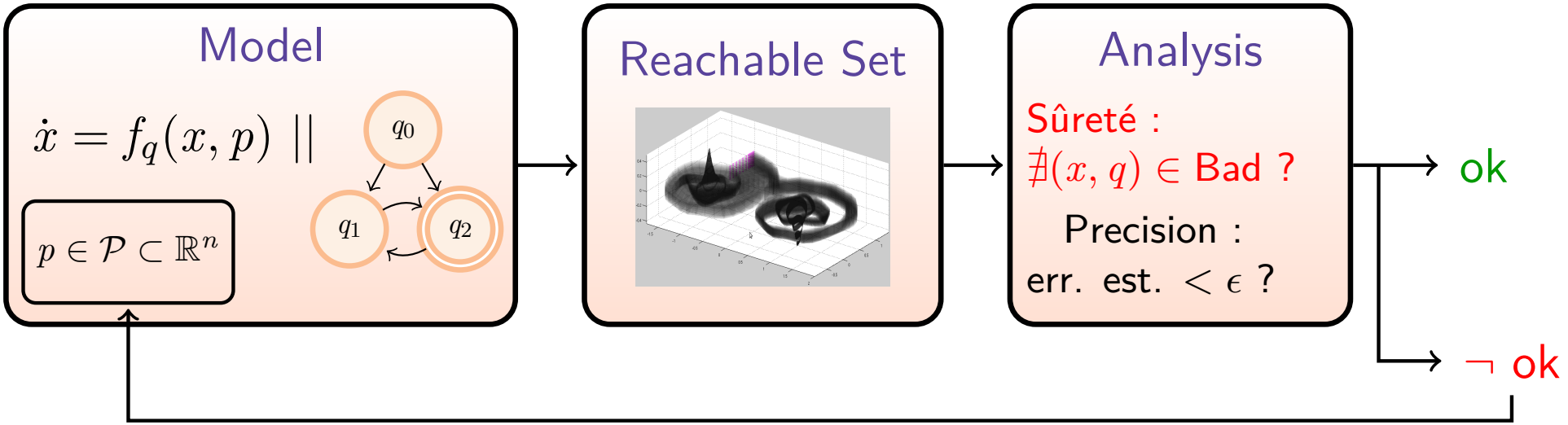
- Using local reachability analysis, sub-regions can be certified
- Iteratively repeating the process, we can find precise boundaries

Parameter synthesis



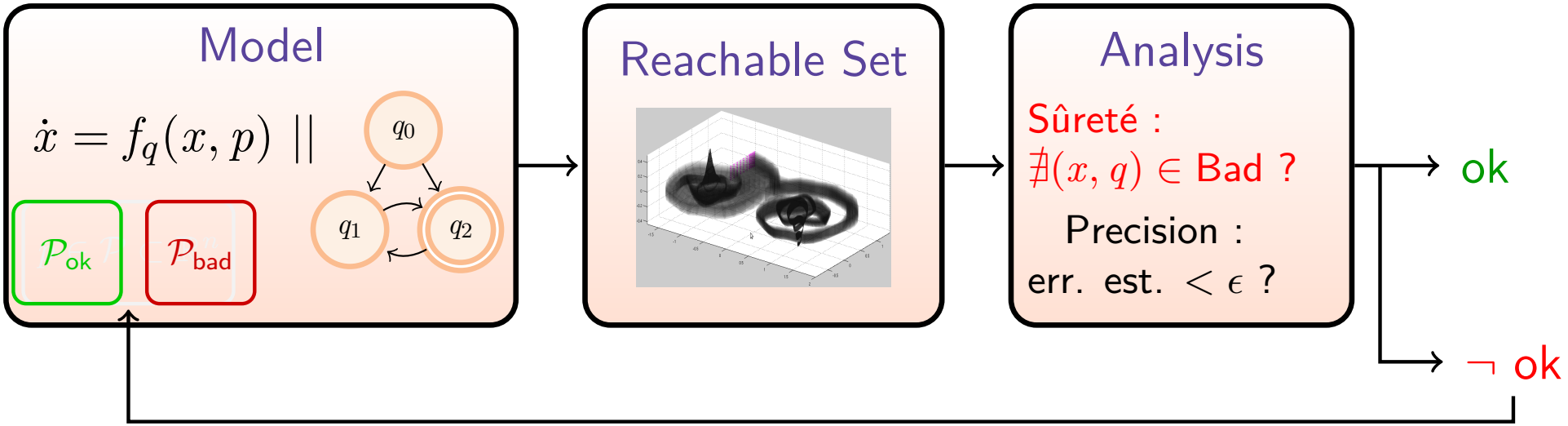
- Using local reachability analysis, sub-regions can be certified
- Iteratively repeating the process, we can find precise boundaries

Parameter synthesis



- Using local reachability analysis, sub-regions can be certified
- Iteratively repeating the process, we can find precise boundaries

Parameter synthesis



- Using local reachability analysis, sub-regions can be certified
- Iteratively repeating the process, we can find precise boundaries

First example

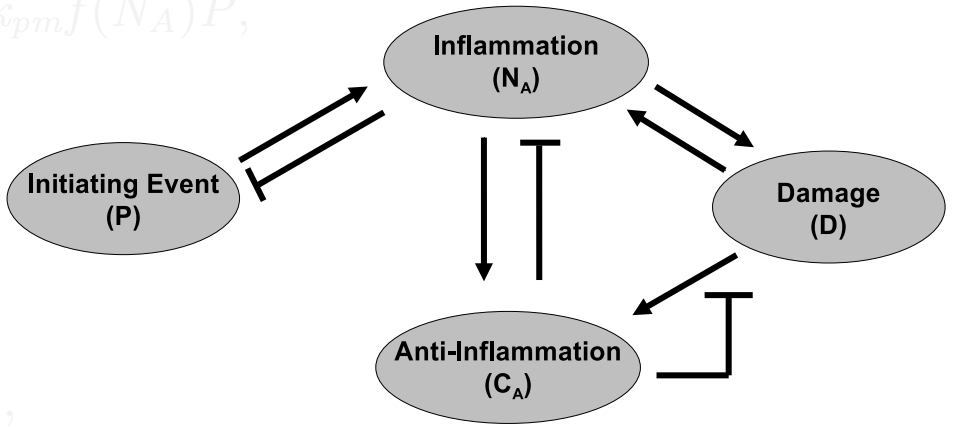
A simple model of the acute inflamatory response to a pathogen infection

$$\frac{dP}{dt} = k_{pg}P\left(1 - \frac{P}{p_\infty}\right) - \frac{k_{pm}s_m P}{\mu_m + k_{mp}P} - k_{pm}f(N_A)P,$$

$$\frac{dN_A}{dt} = \frac{s_{nr}R}{\mu_{nr} + R} - \mu_n N_A,$$

$$\frac{dD}{dt} = k_{dn}f_s(f(N_A)) - \mu_d D,$$

$$\frac{dC_A}{dt} = s_c + \frac{k_{cn}f(N_A + k_{cmd}D)}{1 + f(N_A + k_{cmd}D)} - \mu_c C_A,$$



Three possible outcomes

- Health: pathogen and damage are driven to a low steady state
- Aseptic death: pathogen is eliminated but not tissue damage
- Septic death: tissue damage and pathogen remain high

First example

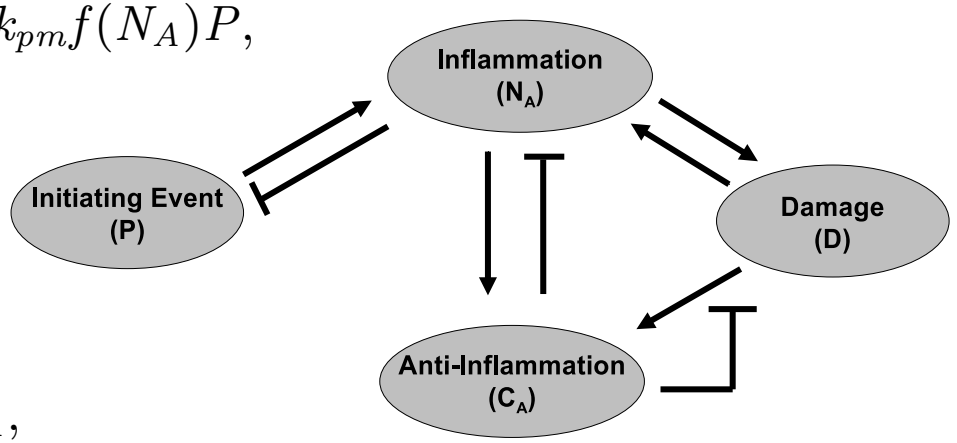
A simple model of the acute inflamatory response to a pathogen infection

$$\frac{dP}{dt} = k_{pg}P\left(1-\frac{P}{p_\infty}\right) - \frac{k_{pm}s_m P}{\mu_m + k_{mp}P} - k_{pm}f(N_A)P,$$

$$\frac{dN_A}{dt} = \frac{s_{nr}R}{\mu_{nr}+R} - \mu_n N_A,$$

$$\frac{dD}{dt} = k_{dn}f_s(f(N_A)) - \mu_d D,$$

$$\frac{dC_A}{dt} = s_c + \frac{k_{cn}f(N_A + k_{cmd}D)}{1+f(N_A+k_{cmd}D)} - \mu_c C_A,$$



Three possible outcomes

- Health: pathogen and damage are driven to a low steady state
- Aseptic death: pathogen is eliminated but not tissue damage
- Septic death: tissue damage and pathogen remain high

First example

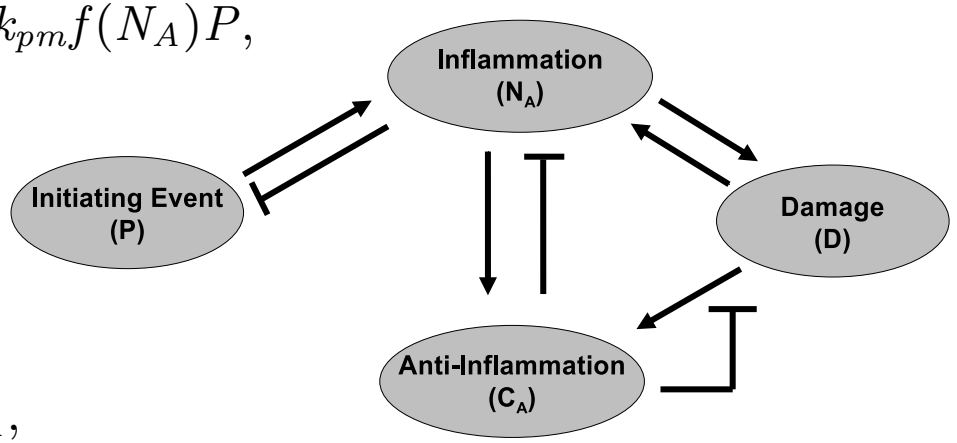
A simple model of the acute inflamatory response to a pathogen infection

$$\frac{dP}{dt} = k_{pg}P\left(1-\frac{P}{p_\infty}\right) - \frac{k_{pm}s_m P}{\mu_m + k_{mp}P} - k_{pm}f(N_A)P,$$

$$\frac{dN_A}{dt} = \frac{s_{nr}R}{\mu_{nr}+R} - \mu_n N_A,$$

$$\frac{dD}{dt} = k_{dn}f_s(f(N_A)) - \mu_d D,$$

$$\frac{dC_A}{dt} = s_c + \frac{k_{cn}f(N_A + k_{cmd}D)}{1+f(N_A+k_{cmd}D)} - \mu_c C_A,$$



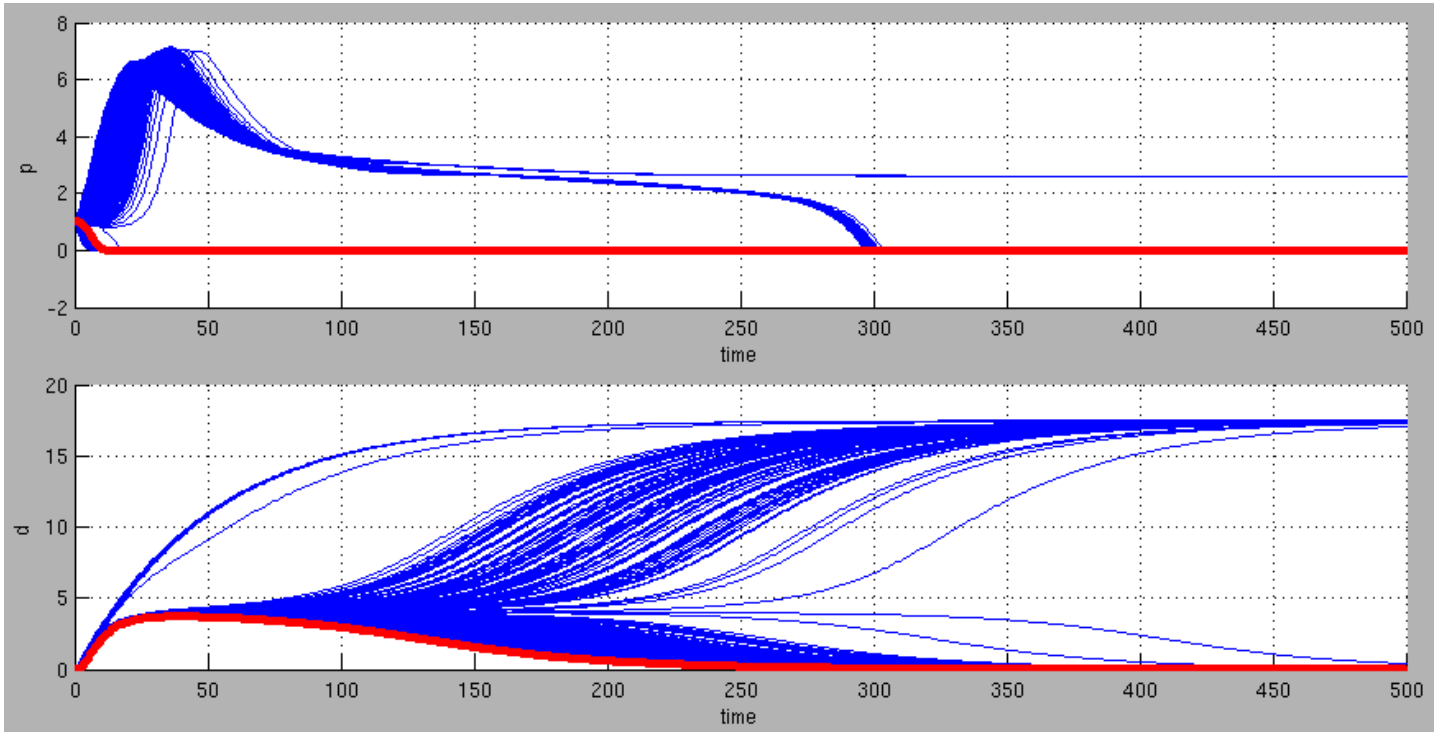
Three possible outcomes

- Health: pathogen and damage are driven to a low steady state
- Aseptic death: pathogen is eliminated but not tissue damage
- Septic death: tissue damage and pathogen remain high

Health outcome

Pathogen

Damage



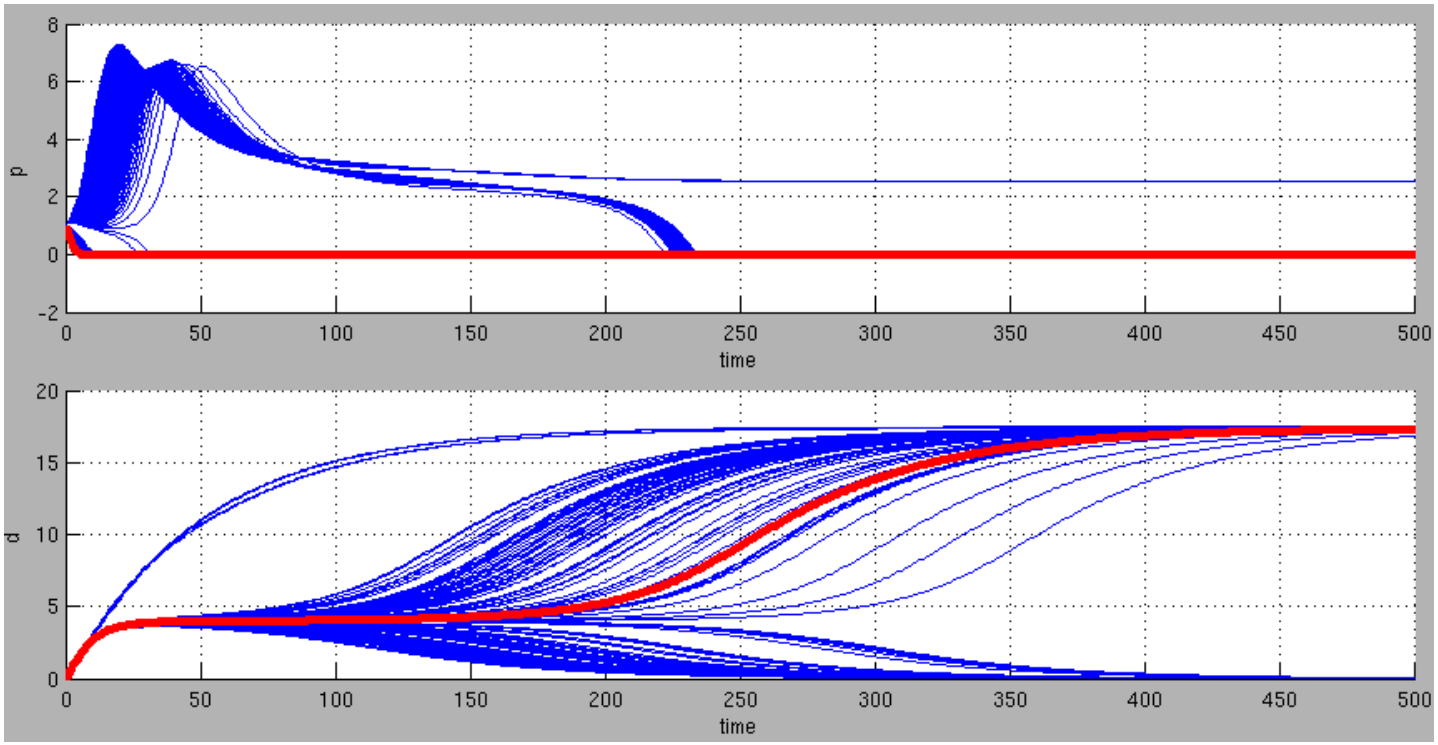
Question

Identify ranges for initial conditions and parameters in the model that lead to predictable outcomes

Aseptic death outcome

Pathogen

Damage



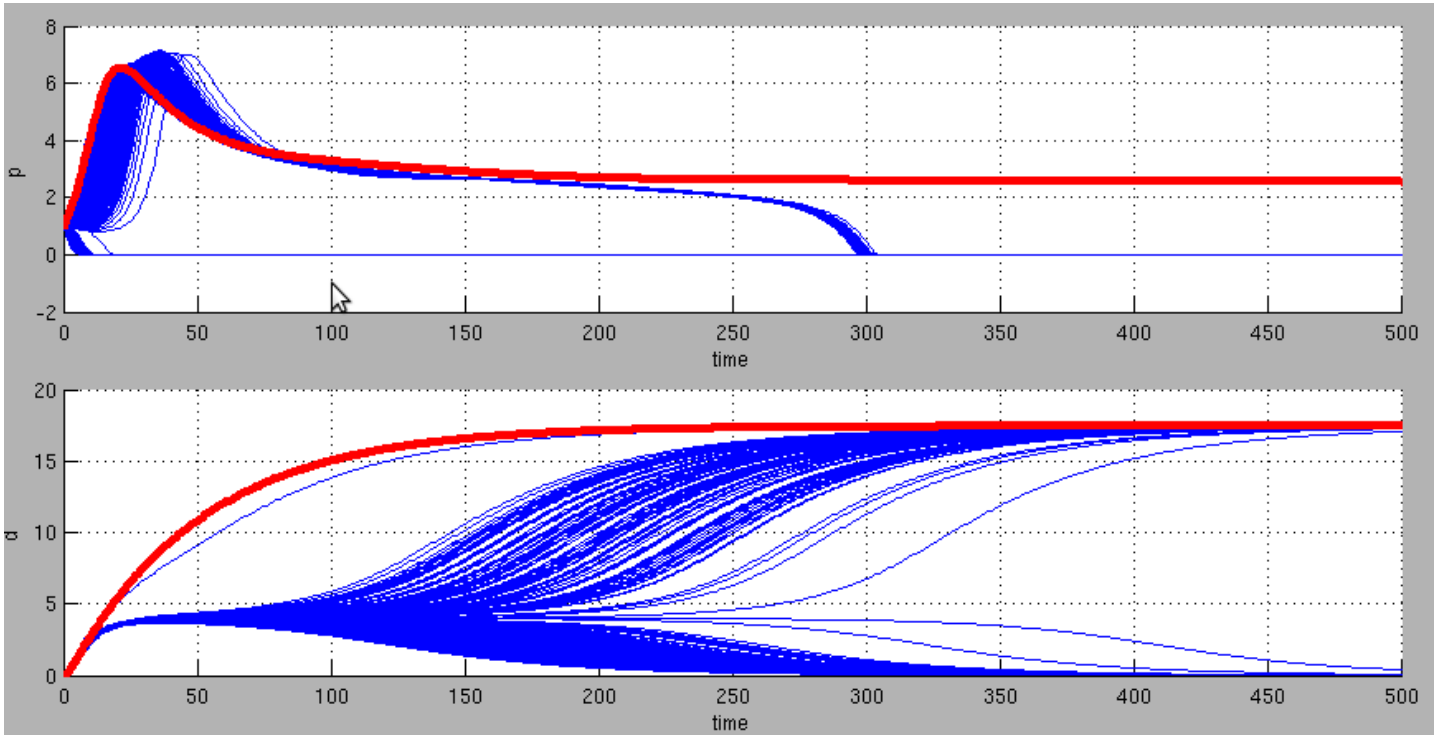
Question

Identify ranges for initial conditions and parameters in the model that lead to predictable outcomes

Septic death outcome

Pathogen

Damage



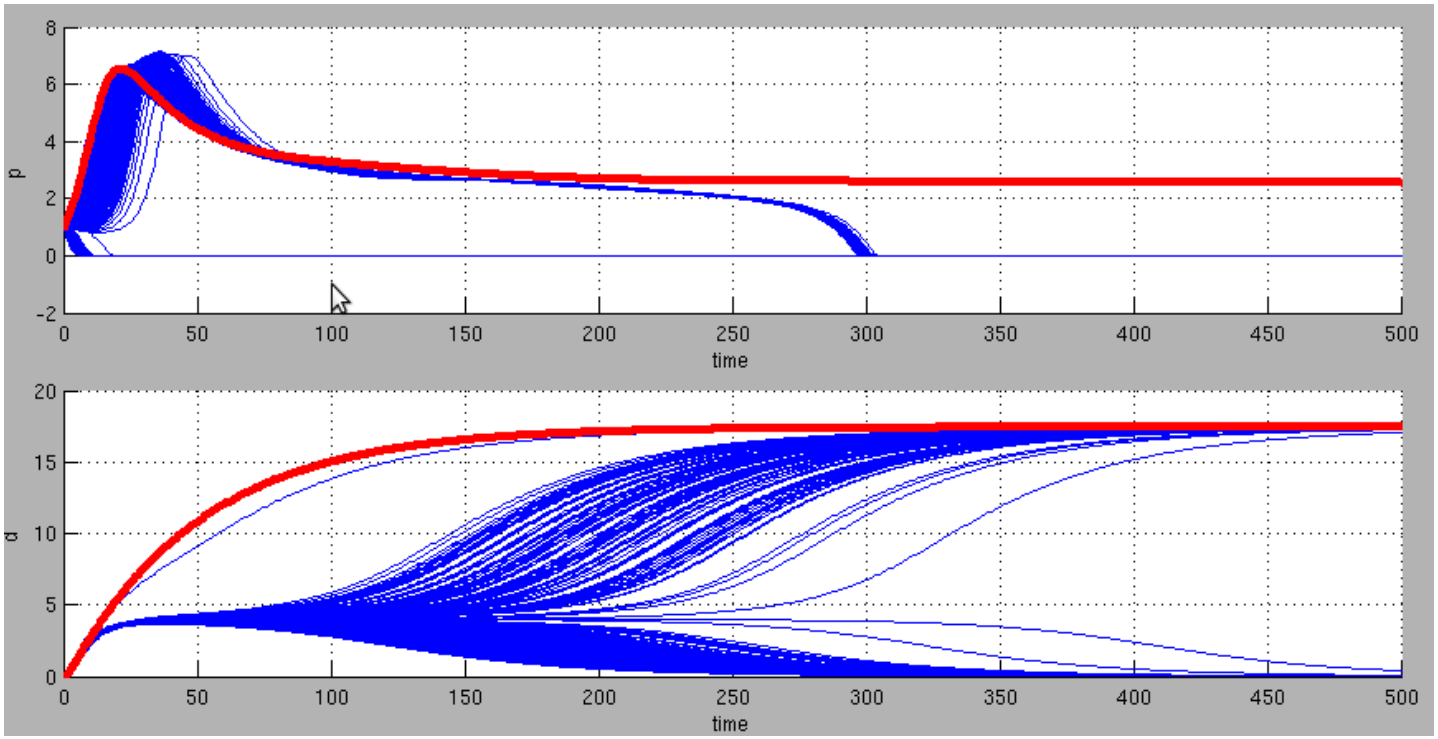
Question

Identify ranges for initial conditions and parameters in the model that lead to predictable outcomes

Septic death outcome

Pathogen

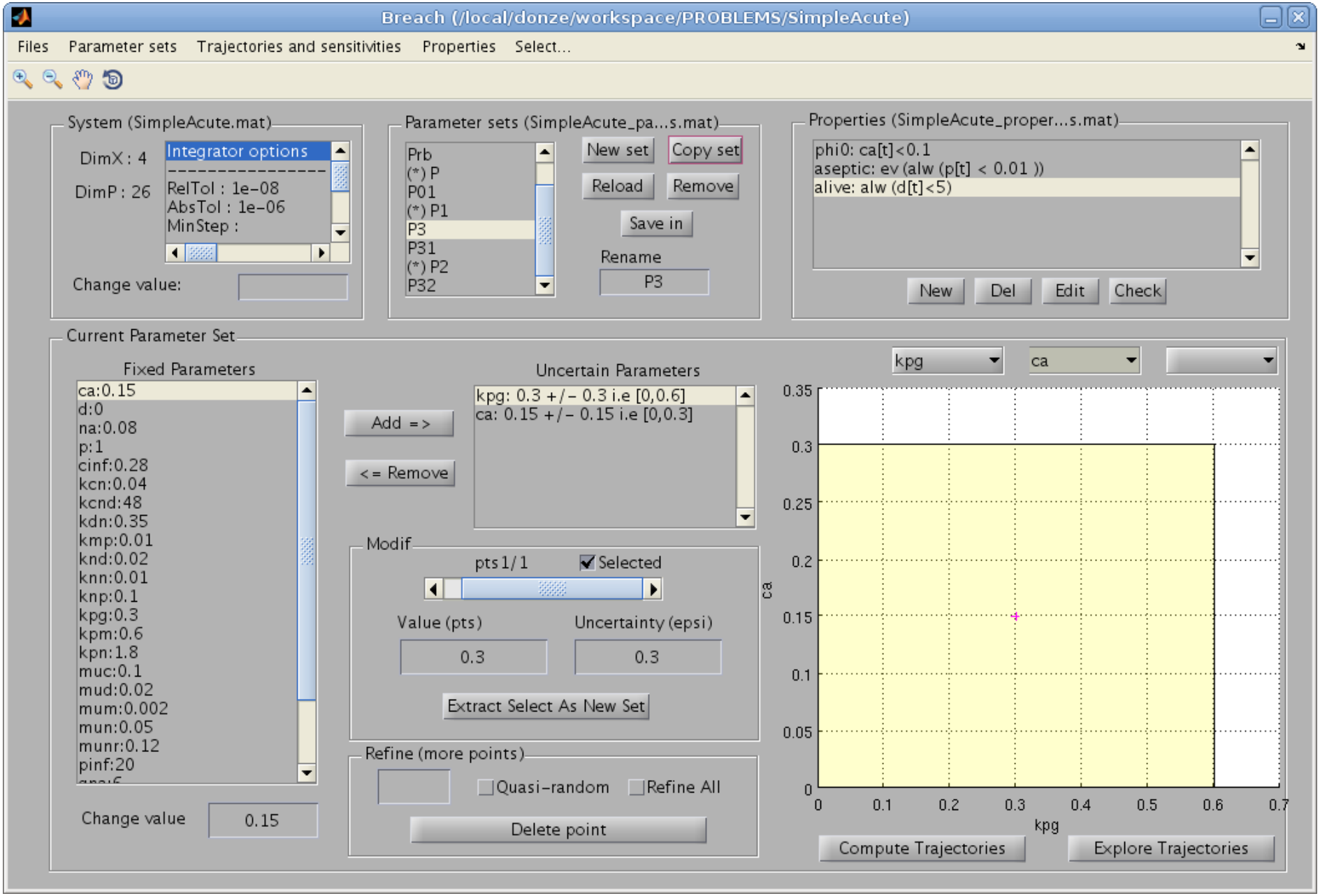
Damage



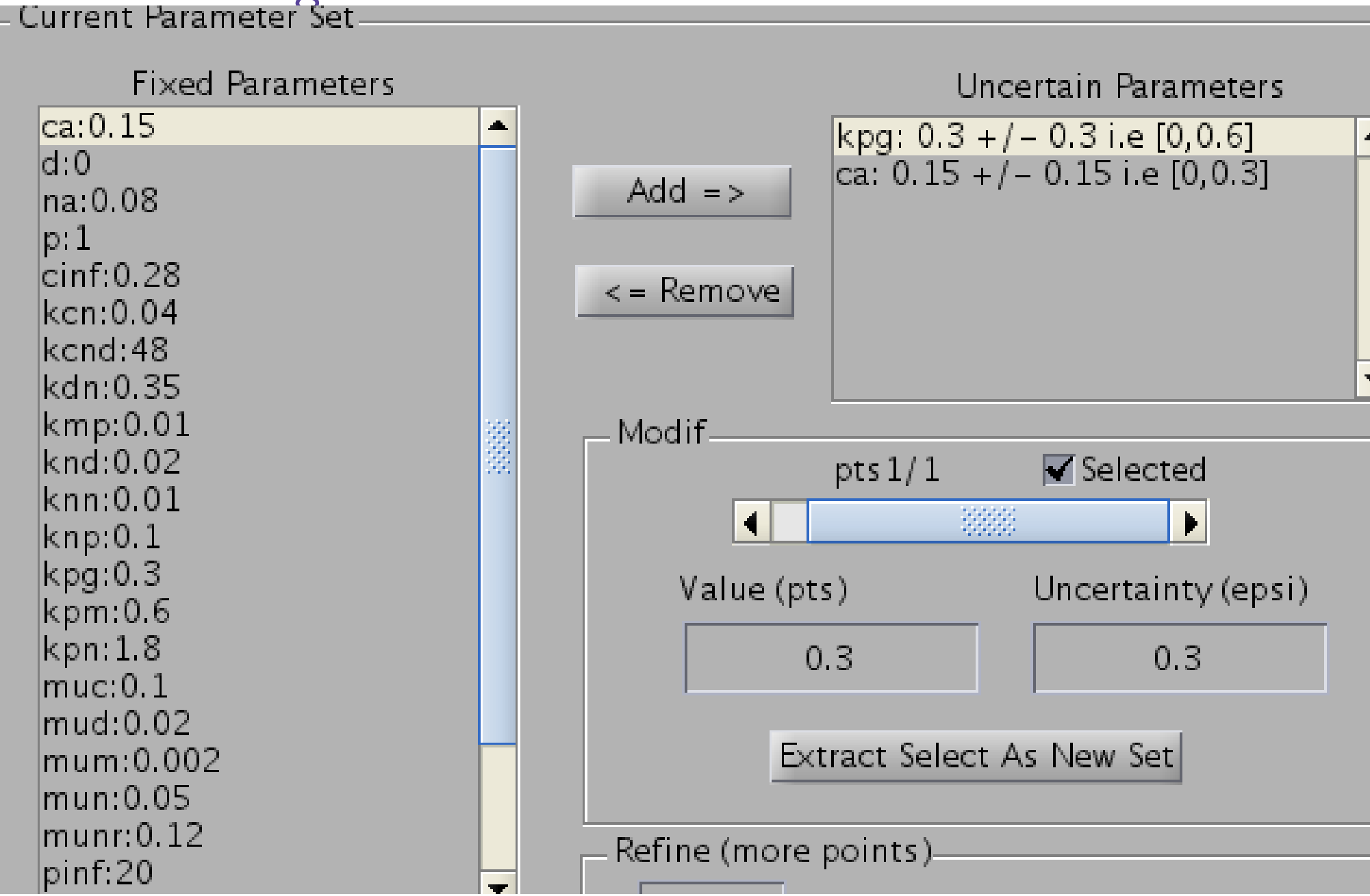
Question

Identify ranges for initial conditions and parameters in the model that lead to predictable outcomes

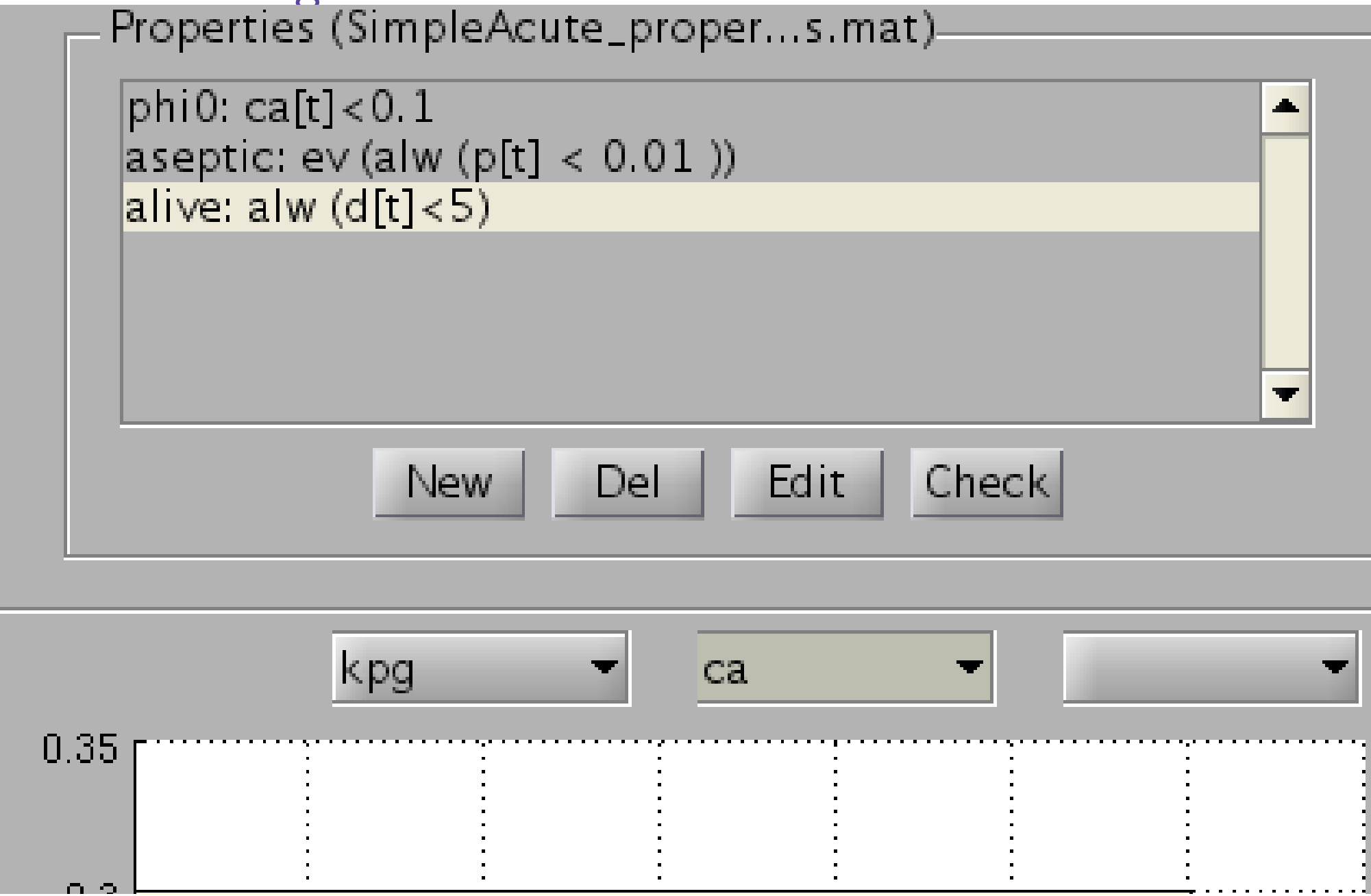
Results using Breach



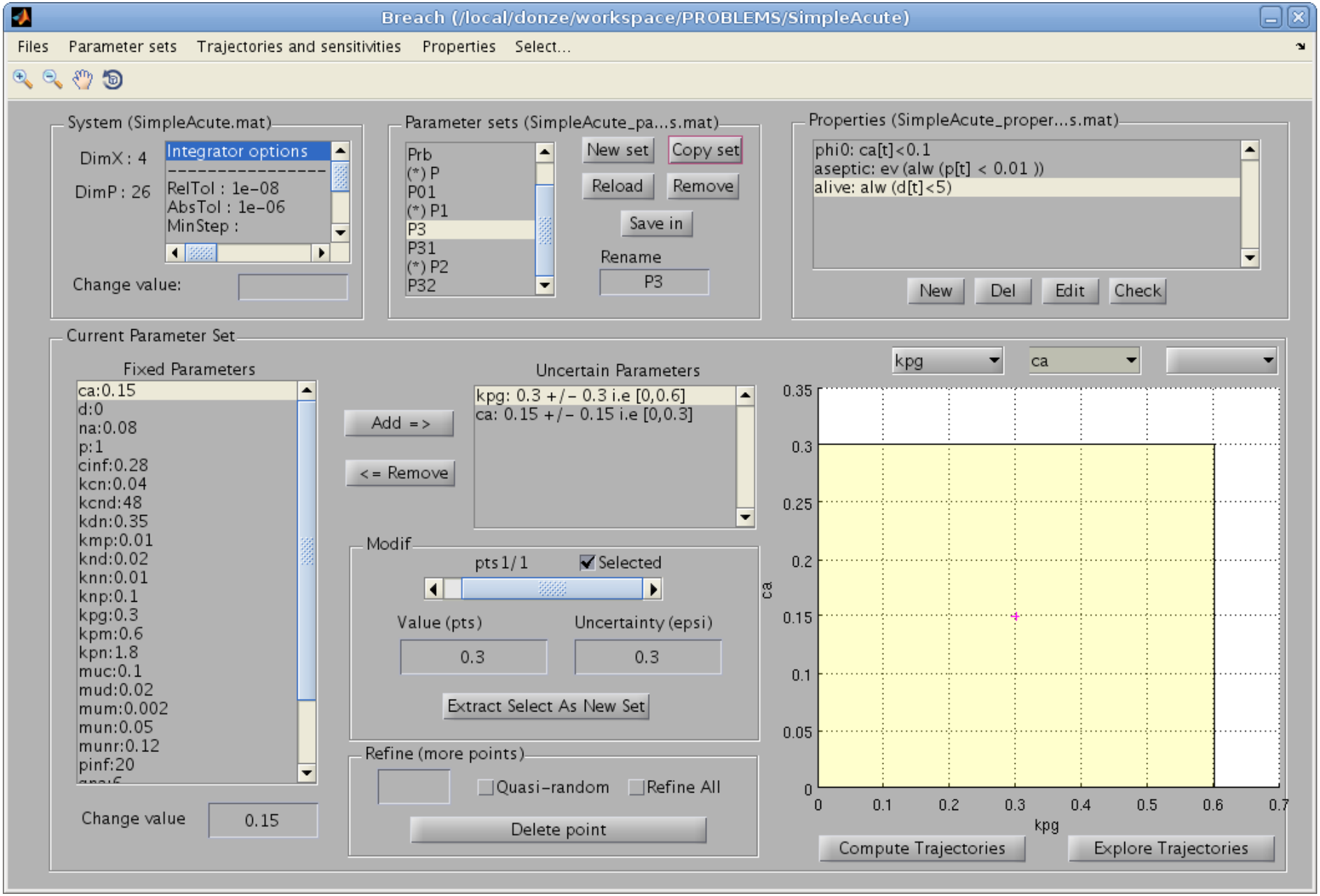
Results using Breach



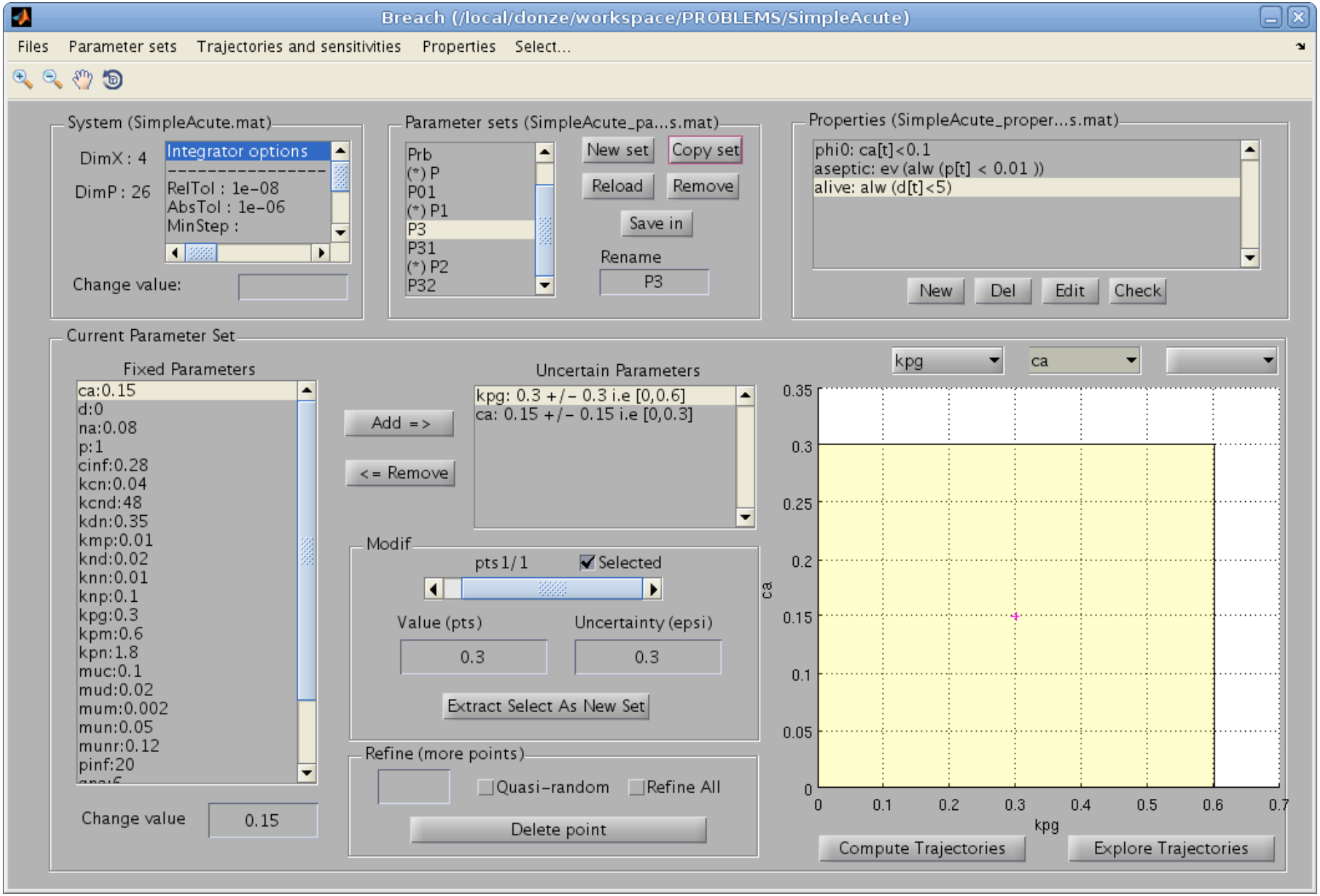
Results using Breach



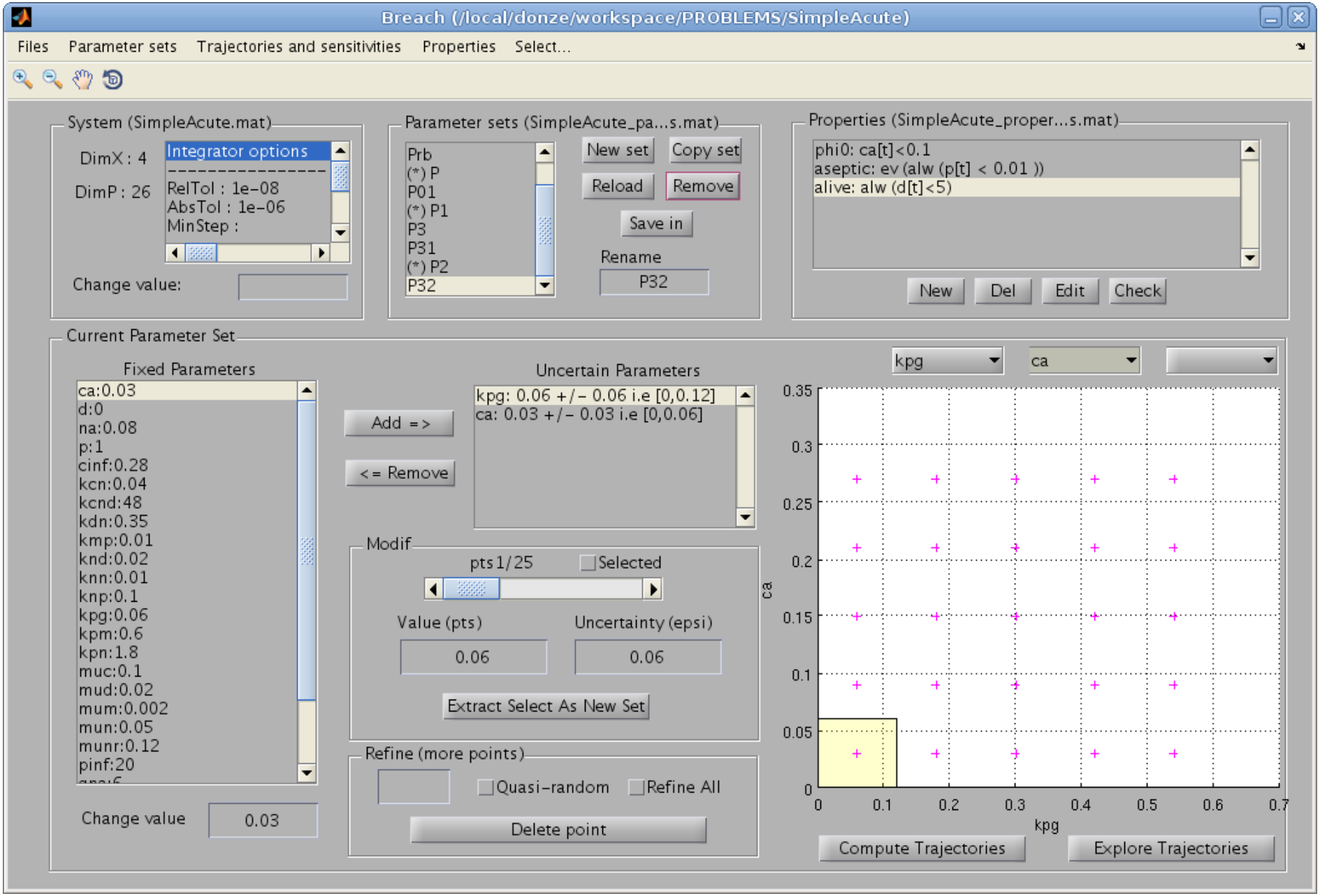
Results using Breach



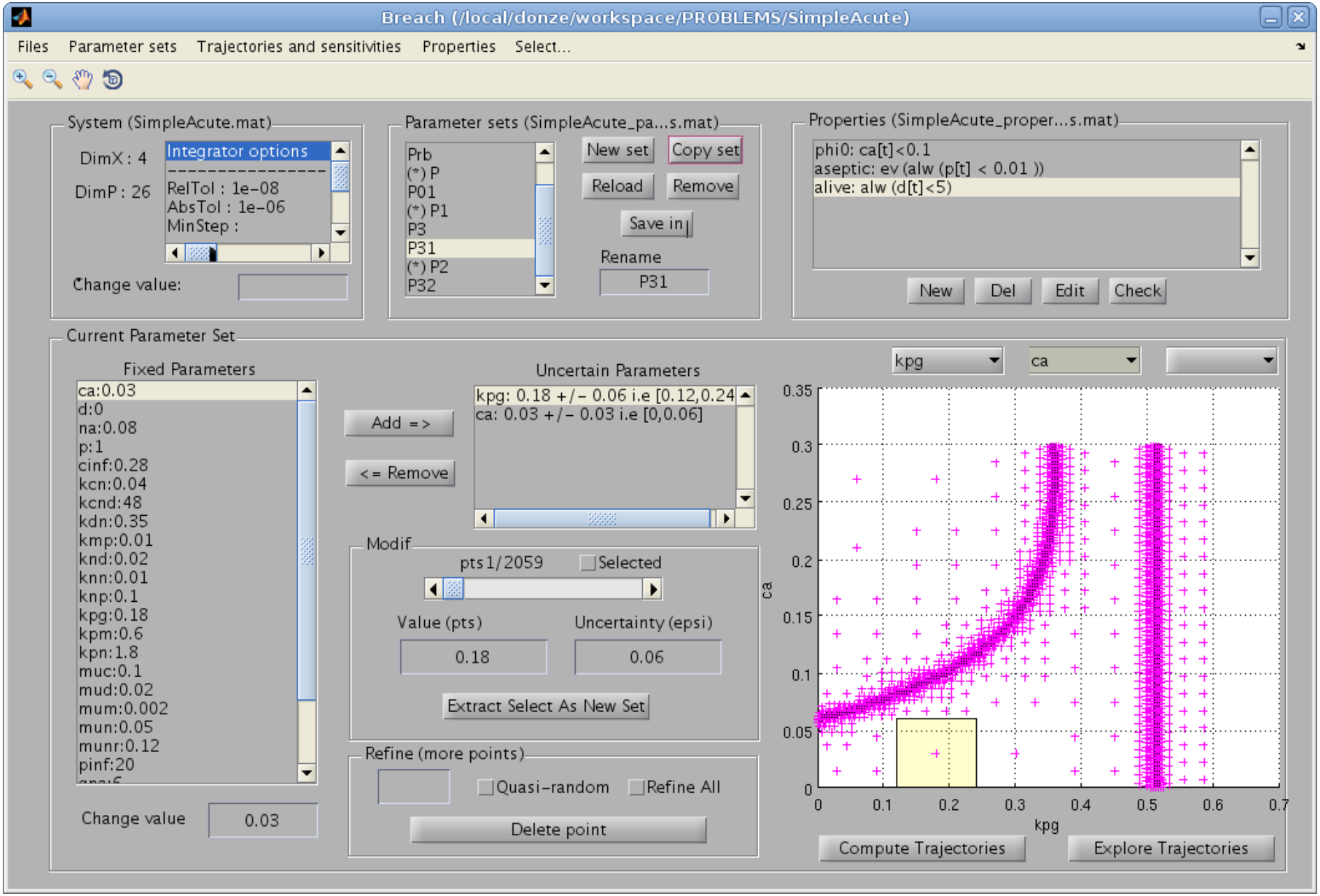
Results using Breach



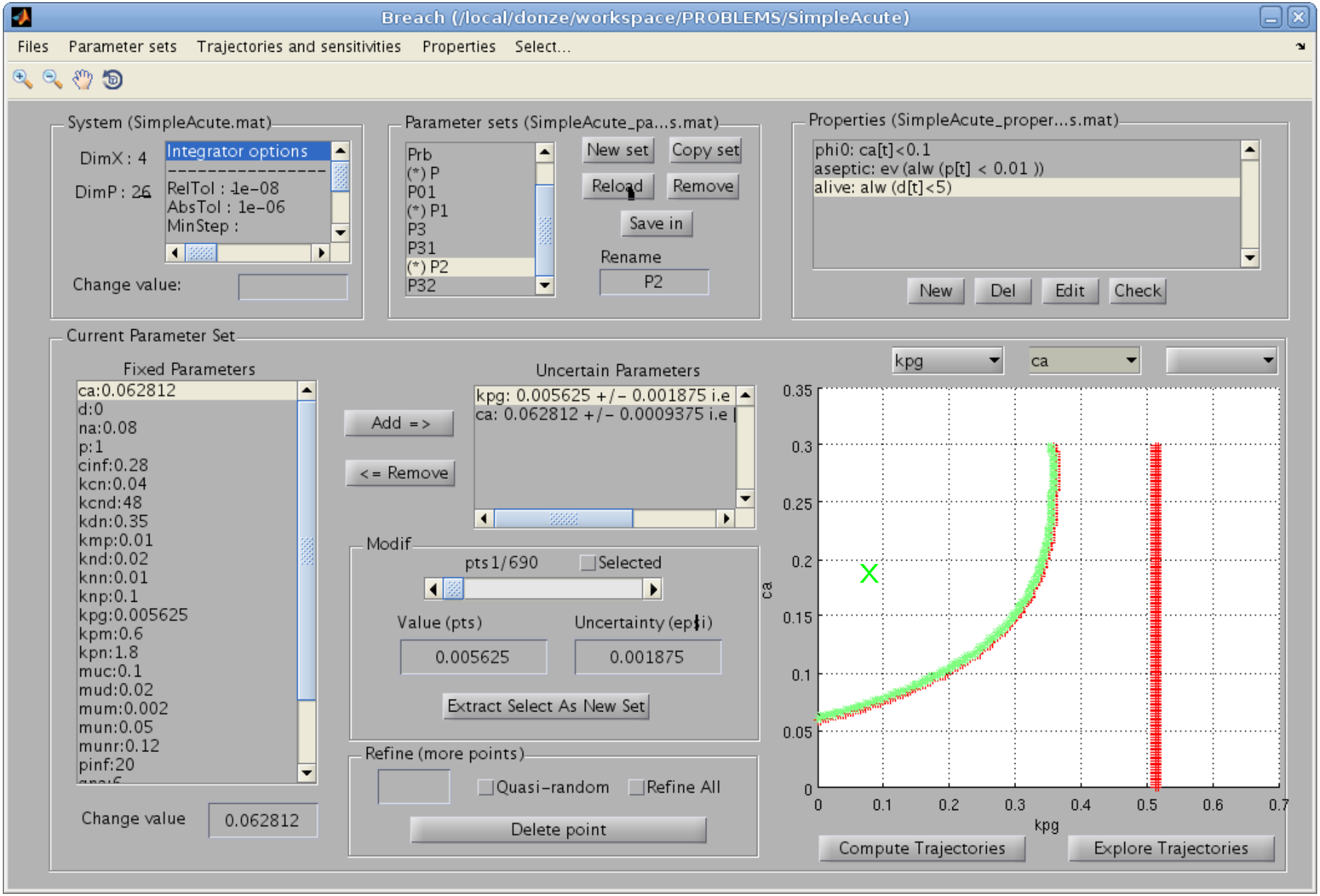
Results using Breach



Results using Breach



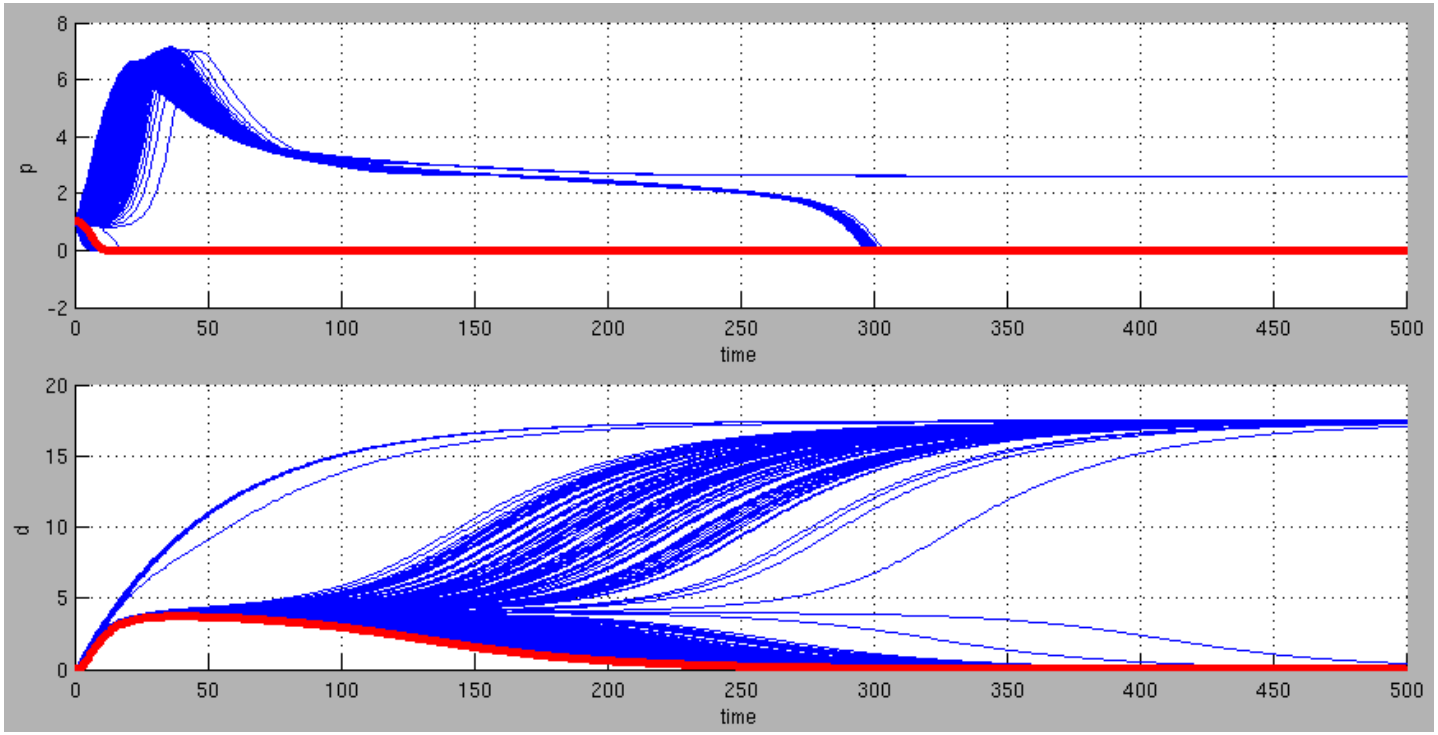
Results using Breach



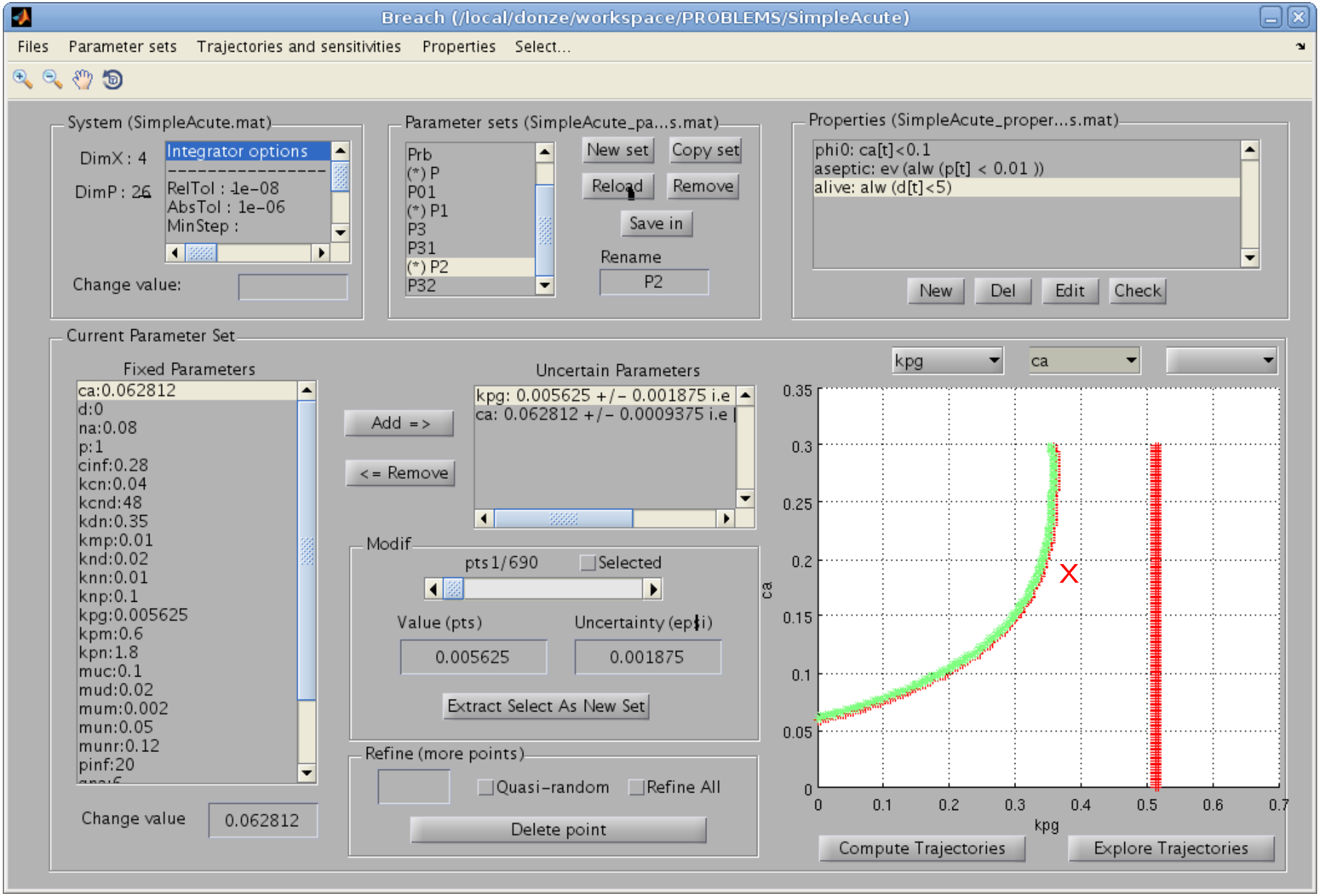
Results using Breach

Pathogen

Damage



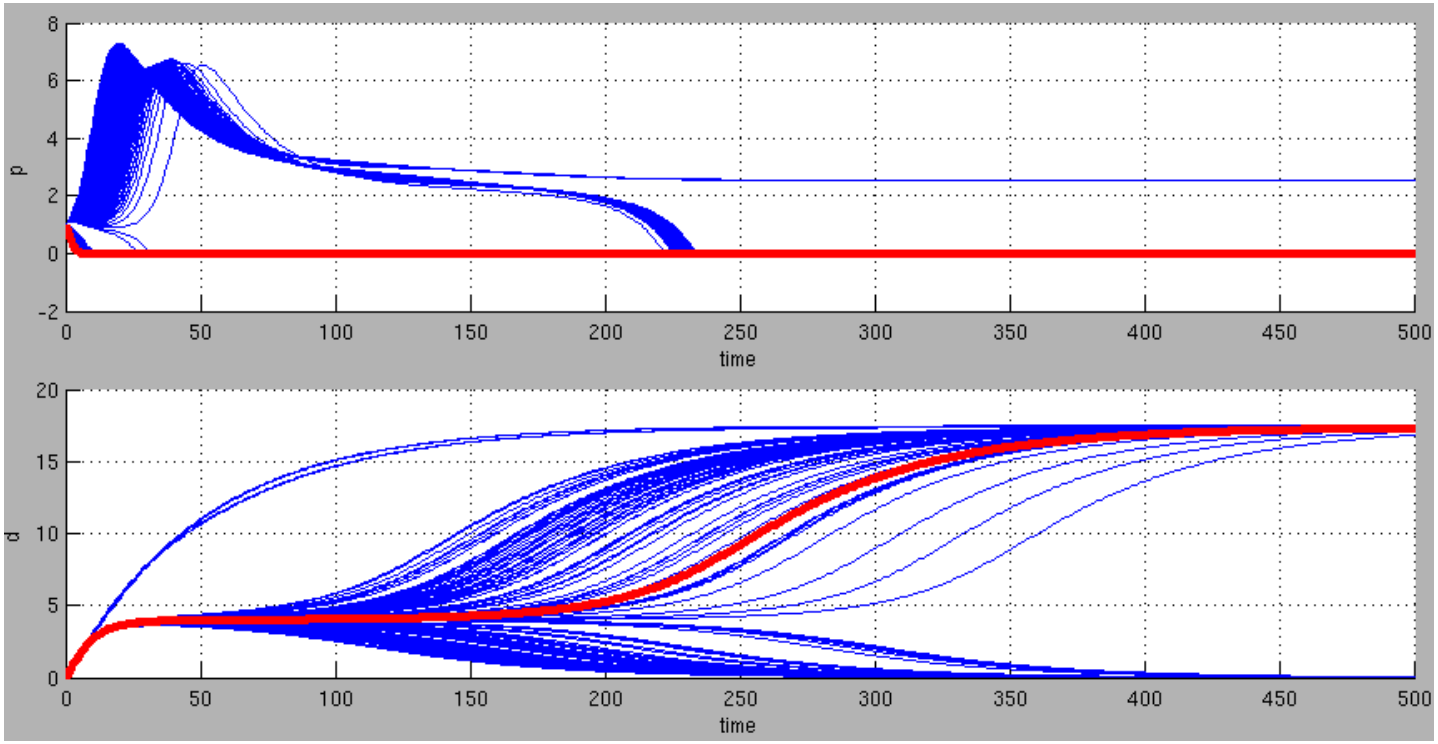
Results using Breach



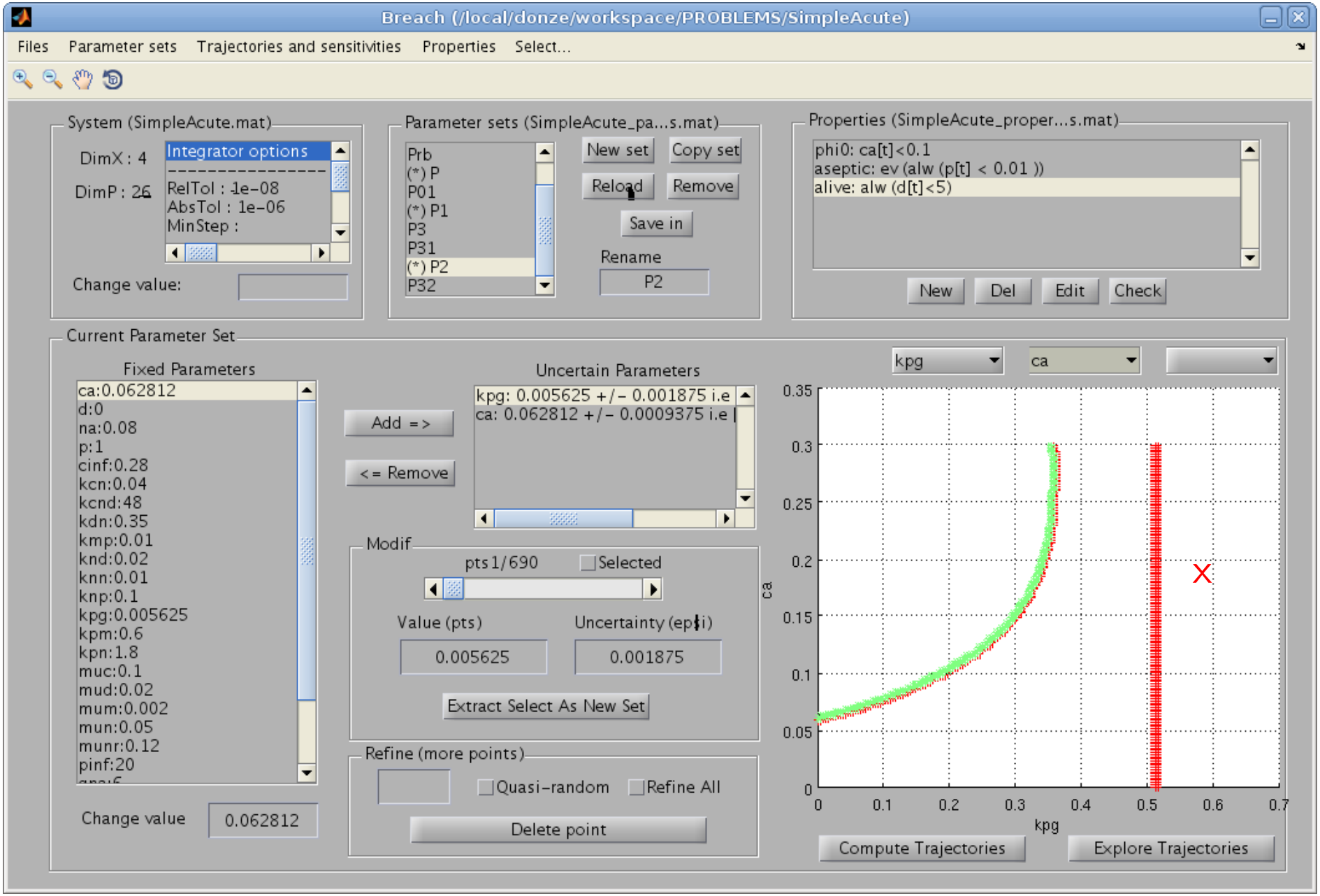
Results using Breach

Pathogen

Damage



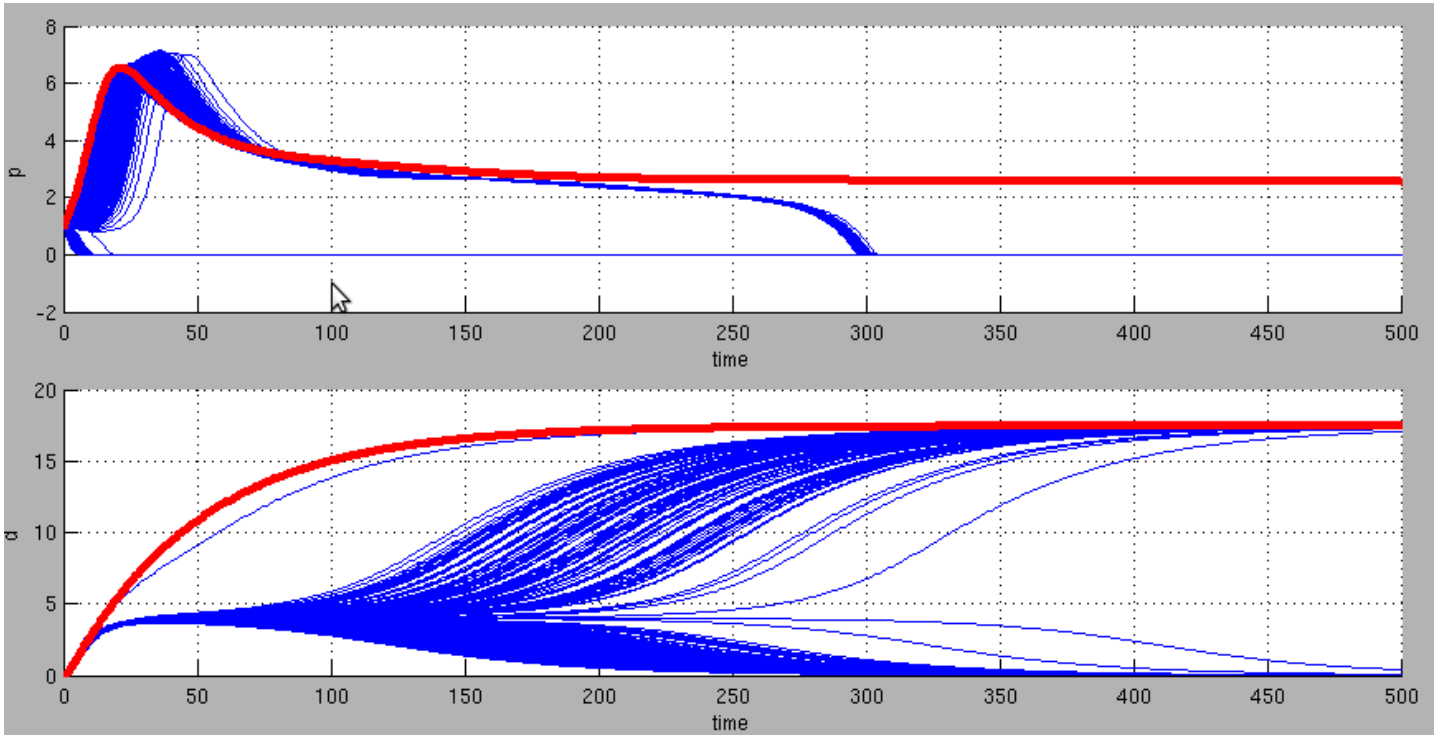
Results using Breach



Results using Breach

Pathogen

Damage



Outline

Motivations

The technique presented so far deals with *safety* properties

Theory shows that every temporal property on a bounded timed horizon can be expressed as a safety property

Since life has a bounded time horizon, this should be enough...

However, translating a property of interest into a safety property is not always trivial nor intuitive, and error prone

Motivations

The technique presented so far deals with *safety* properties

Theory shows that every temporal property on a bounded timed horizon can be expressed as a safety property

Since life has a bounded time horizon, this should be enough...

However, translating a property of interest into a safety property is not always trivial nor intuitive, and error prone

Motivations

The technique presented so far deals with *safety* properties

Theory shows that every temporal property on a bounded timed horizon can be expressed as a safety property

Since life has a bounded time horizon, this should be enough...

However, translating a property of interest into a safety property is not always trivial nor intuitive, and error prone

Temporal Logics

A key issue is the appropriate choice of language to describe properties:

- Enough expressivity
- Ease of writing specification

Temporal logics popularized in 1978 by Amir Pnueli when programs shifted from simple input-output relations to reactive programs.

A typical reactive program is an operating system:

- a good property is *always when the mouse is moved, the cursors moves*
- a bad one: *always eventually a blue screen appears and nothing happens*

A good property such as the one above is a *liveness property*.

Temporal Logics

A key issue is the appropriate choice of language to describe properties:

- Enough expressivity
- Ease of writing specification

Temporal logics popularized in 1978 by Amir Pnueli when programs shifted from simple input-output relations to reactive programs.

A typical reactive program is an operating system:

- a good property is *always when the mouse is moved, the cursors moves*
- a bad one: *always eventually a blue screen appears and nothing happens*

A good property such as the one above is a *liveness property*.

Temporal Logics

A key issue is the appropriate choice of language to describe properties:

- Enough expressivity
- Ease of writing specification

Temporal logics popularized in 1978 by Amir Pnueli when programs shifted from simple input-output relations to reactive programs.

A typical reactive program is an operating system:

- a good property is *always when the mouse is moved, the cursors moves*
- a bad one: *always eventually a blue screen appears and nothing happens*

A good property such as the one above is a *liveness property*.

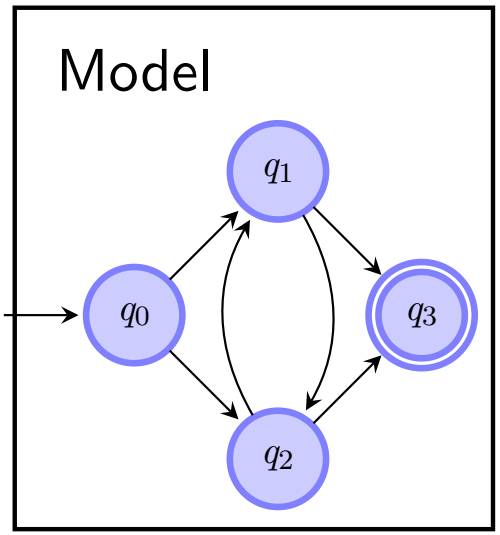
Temporal logics in a nutshell

Temporal logics allow to specify patterns that timed behaviors of systems may or may not satisfy. They come in many flavors

The most intuitive is the Linear Temporal Logic (LTL), defined over discrete sequences of states

It is based on logic operators ($\neg$, $\wedge$, $\vee$) and temporal operators : "next", "always" (alw), "eventually" (ev) and "until" ($\mathcal{U}$)
Examples:

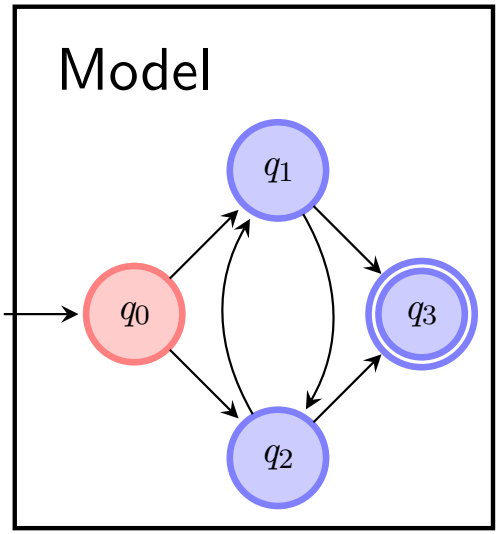
- $\varphi\ \varphi\ \varphi\ \varphi\ \cdots$ satisfies alw φ
- $\psi\ \psi\ \psi\ \varphi\ \psi\ \cdots$ satisfies ev φ
- $\varphi\ \varphi\ \varphi\ \varphi\ \psi\ \cdots$ satisfies $\varphi\ \mathcal{U}\ \psi$



$\models$?

Specifications

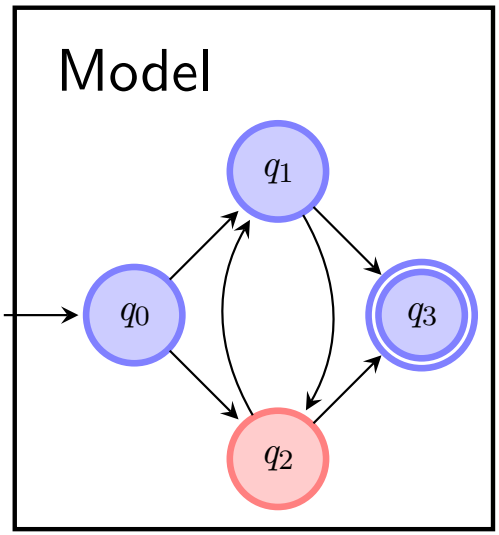
$\varphi := \mathsf{alw}(q_0 \Rightarrow \mathsf{ev}\, q_3)$



$\not\models$

Specifications

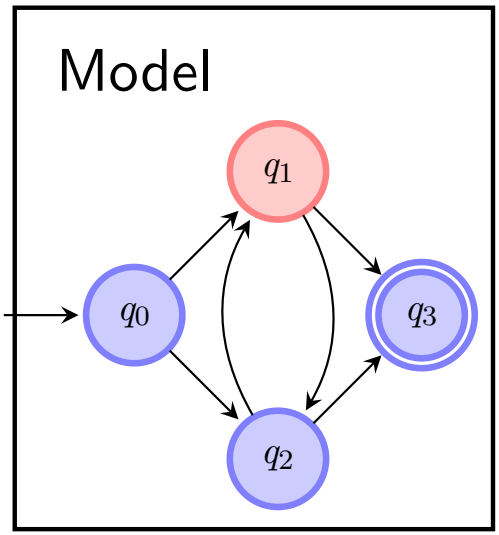
$\varphi := \mathsf{alw}(q_0 \Rightarrow \mathsf{ev}\, q_3)$



$\not\models$

Specifications

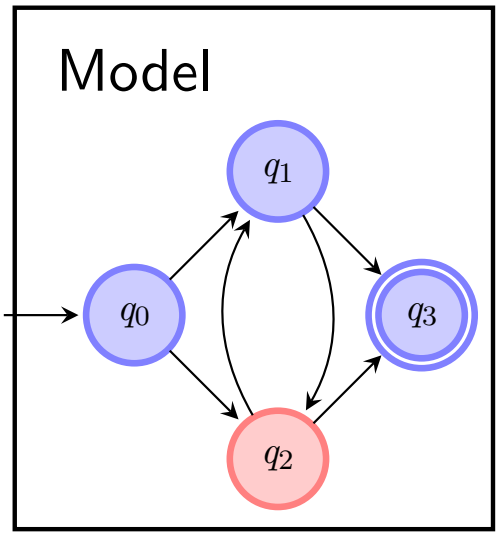
$\varphi := \mathsf{alw}(q_0 \Rightarrow \mathsf{ev}\, q_3)$



$\not\models$

Specifications

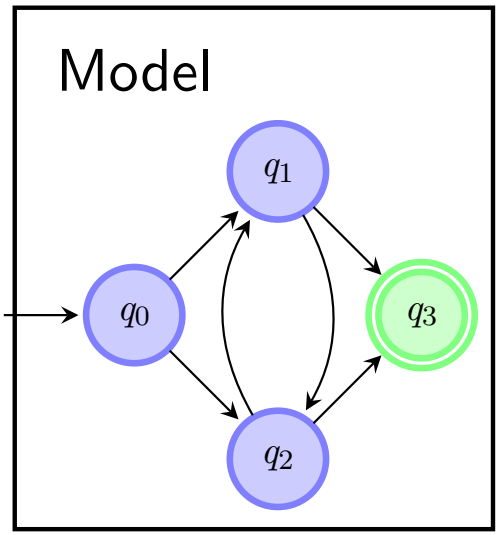
$\varphi := \mathsf{alw}(q_0 \Rightarrow \mathsf{ev}\, q_3)$



$\not\models$

Specifications

$\varphi := \mathsf{alw}(q_0 \Rightarrow \mathsf{ev}\, q_3)$



$\models$

Specifications

$\varphi := \mathsf{alw}(q_0 \Rightarrow \mathsf{ev}\, q_3)$

From discrete to continuous

Temporal logics mostly developped for discrete systems, a natural way to go is to discretize time and space

However this means that formulas apply to an abstraction of the system, thus introducing a distance between specification and the "real" system

Temporal logics adapted to continuous time and space

- spatial constraints are specified on the real-valued quantities
- temporal constraints involve dense-time intervals rather than e.g. fixed time steps

From discrete to continuous

Temporal logics mostly developped for discrete systems, a natural way to go is to discretize time and space

However this means that formulas apply to an abstraction of the system, thus introducing a distance between specification and the "real" system

Temporal logics adapted to continuous time and space

- spatial constraints are specified on the real-valued quantities
- temporal constraints involve dense-time intervals rather than e.g. fixed time steps

Temporal logic formulas: atomic predicates

A predicate is a general inequality constraints on the variables (say A, B, C etc) and parameters at time t

```
% distance to (A0,B0) is more than 1.
 sqrt((A[t]-A0)^2 + (B[t]-B0)^2) > 1.

% the system reached quasi stationnary steady state
abs(ddt{A}[t])+abs(ddt{A}[t])) < 1e-10

% A is sensitive to parameter p
abs(d{A}{p}[t]) > 10*A[t]/p
```

The canonical form of a predicate μ is:

$$\mu \equiv \mu(\xi_{\mathbf{p}}, t) \geq 0$$

Temporal logic operators

Metric Interval Temporal Logic (MITL) syntax:

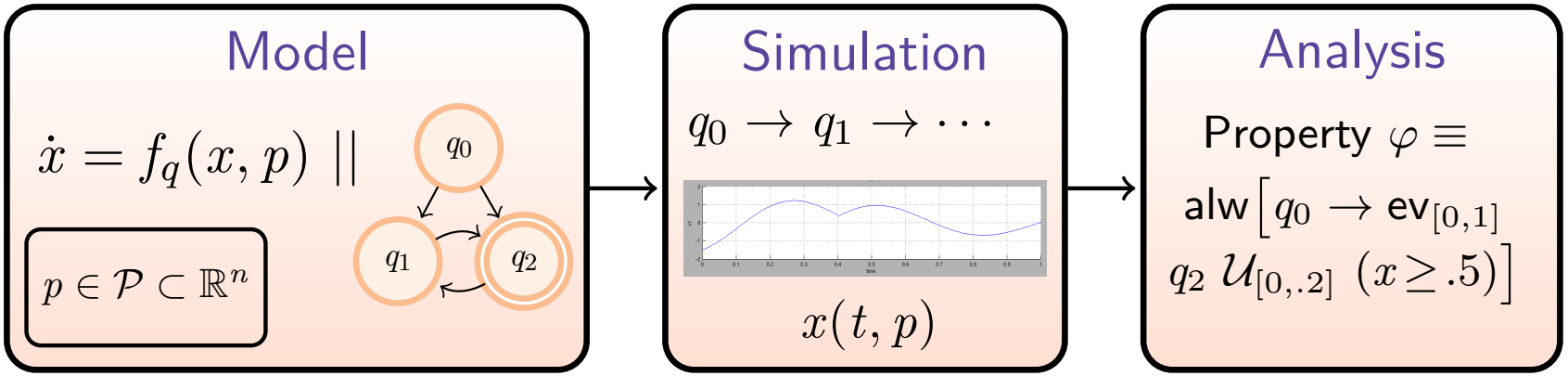
$$\varphi := \mu \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi\, \mathcal{U}_{[a,b)}\ \varphi | \mathsf{ev}_{[a,b)}\ \varphi | \mathsf{alw}_{[a,b)}\ \varphi$$

```
% The concentration of A becomes more than 1e-6 within 2 s
ev_[0,2] (A[t]> 1e-6)

% A remains low until B is quasi stationary before 10 seconds
(A[t] < 1e-8) until_[0, 10] always ((abs(ddt{B}[t]) < 1e-9))
```

The result is a query language which is close enough to English formulation.

Formalizing continuous and hybrid dynamic behaviors



Qualitative

"$\exists$ a stable steady state", "converges to a limit cycle"

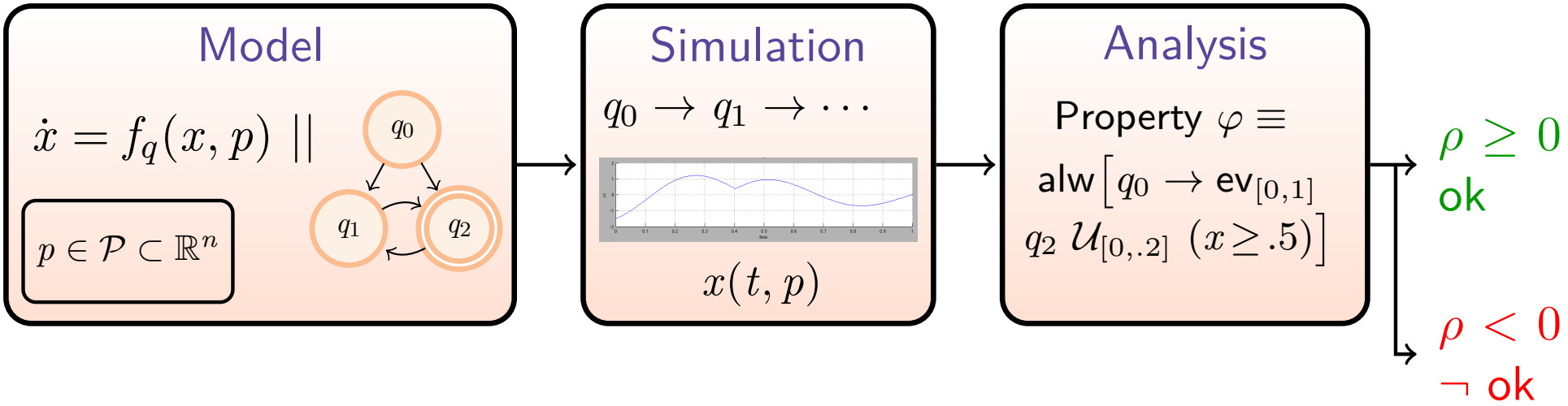
Quantitative/transient

"$\exists$ an interval of 20 s. when x is above 0.5",
"x is periodic with period ≤ 2s and amplitude ≥ 0.1"

Robust satisfaction

$\rho \equiv$ margin of satisfaction or violation for spatial and temporal constraints

Formalizing continuous and hybrid dynamic behaviors



Qualitative

"$\exists$ a stable steady state", "converges to a limit cycle"

Quantitative/transient

"$\exists$ an interval of 20 s. when x is above 0.5",
"x is periodic with period ≤ 2s and amplitude ≥ 0.1"

Robust satisfaction

$\rho \equiv$ margin of satisfaction or violation for spatial and temporal constraints

Computing the satisfaction function

For a predicate $\mu(\xi_p, \tau) \geq 0$, we have simply $\rho(\mu, \xi_p, \tau) = \mu(\xi_p, \tau)$

For operators: extension of the known correspondance between $\min - \max$ operators and boolean operators:

$$
\begin{array}{lcl}
\rho(\neg\varphi, \xi_p, \tau) & = & -\rho(\varphi, \xi_p, \tau) \\
\rho(\varphi_1 \wedge \varphi_2, \xi_p, \tau) & = & \min(\rho(\varphi_1, \xi_p, \tau), \rho(\varphi_2, \xi_p, \tau)) \\
\rho(\mathsf{ev}_{[a,b]}\ \varphi) & = & \max\limits_{\tau' \in [\tau+a,\ \tau+b]} \rho(\varphi, \xi_p, \tau') \\
\rho(\varphi_1 \mathcal{U}_{[a,b]}\ \varphi_2, \xi_p, t) & = & \max\limits_{r \in \tau+[a,b]} (\min(\rho(\varphi_2, \xi_p, r), \min\limits_{s \in [\tau,r]} \rho(\varphi_1, \xi_p, s))
\end{array}
$$

Computing ρ is somehow tricky but the cost can be roughly linear in the size of the formula and the length of the simulation (small computional overhead)

Computing the satisfaction function

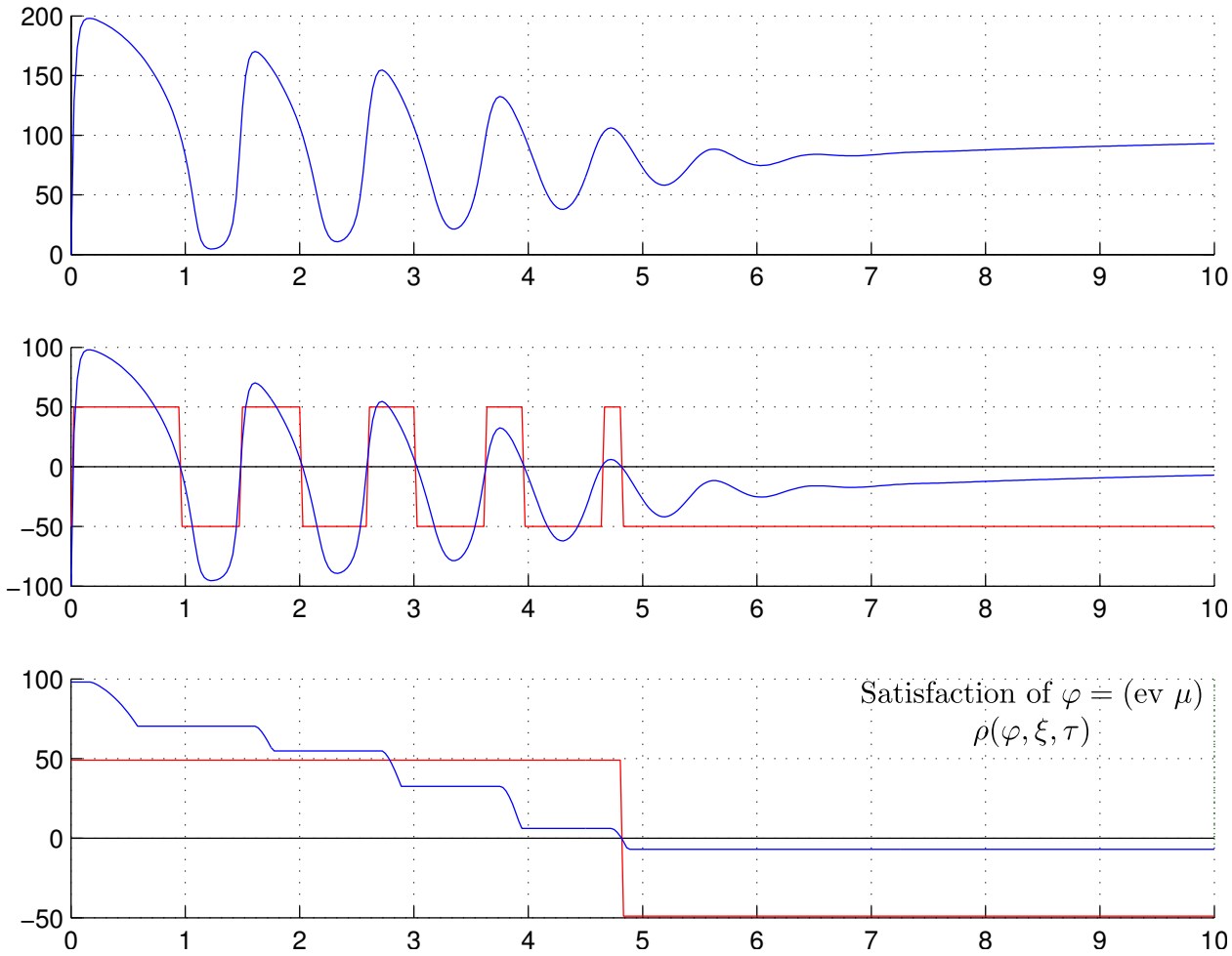
For a predicate $\mu(\xi_p, \tau) \geq 0$, we have simply $\rho(\mu, \xi_p, \tau) = \mu(\xi_p, \tau)$

For operators: extension of the known correspondance between $\min - \max$ operators and boolean operators:

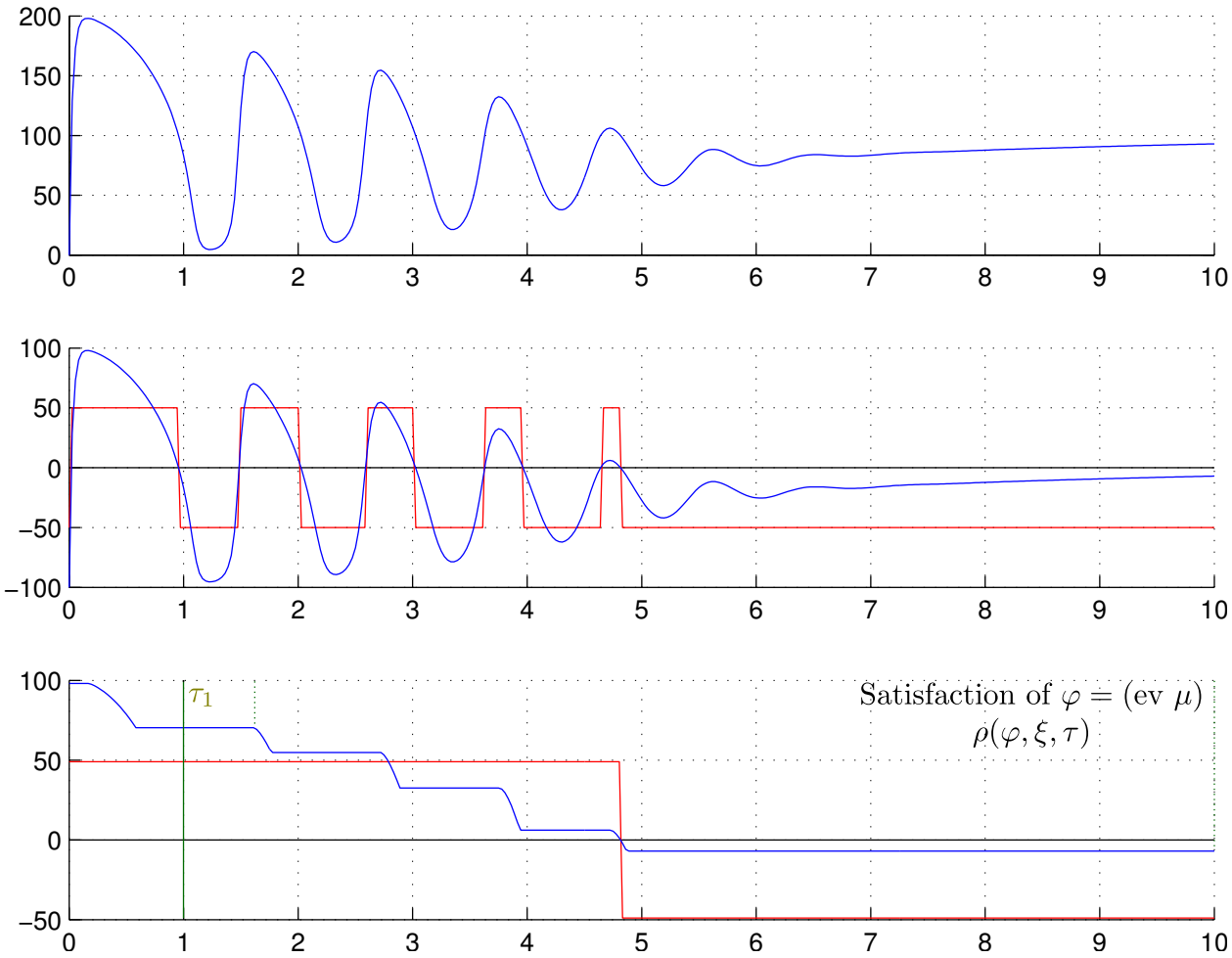
$$
\begin{array}{lcl}
\rho(\neg\varphi, \xi_p, \tau) & = & -\rho(\varphi, \xi_p, \tau) \\
\rho(\varphi_1 \wedge \varphi_2, \xi_p, \tau) & = & \min(\rho(\varphi_1, \xi_p, \tau), \rho(\varphi_2, \xi_p, \tau)) \\
\rho(\mathsf{ev}_{[a,b]}\ \varphi) & = & \max\limits_{\tau' \in [\tau+a,\ \tau+b]} \rho(\varphi, \xi_p, \tau') \\
\rho(\varphi_1 \mathcal{U}_{[a,b]}\ \varphi_2, \xi_p, t) & = & \max\limits_{r \in \tau+[a,b]} (\min(\rho(\varphi_2, \xi_p, r), \min\limits_{s \in [\tau,r]} \rho(\varphi_1, \xi_p, s))
\end{array}
$$

Computing ρ is somehow tricky but the cost can be roughly linear in the size of the formula and the length of the simulation (small computional overhead)

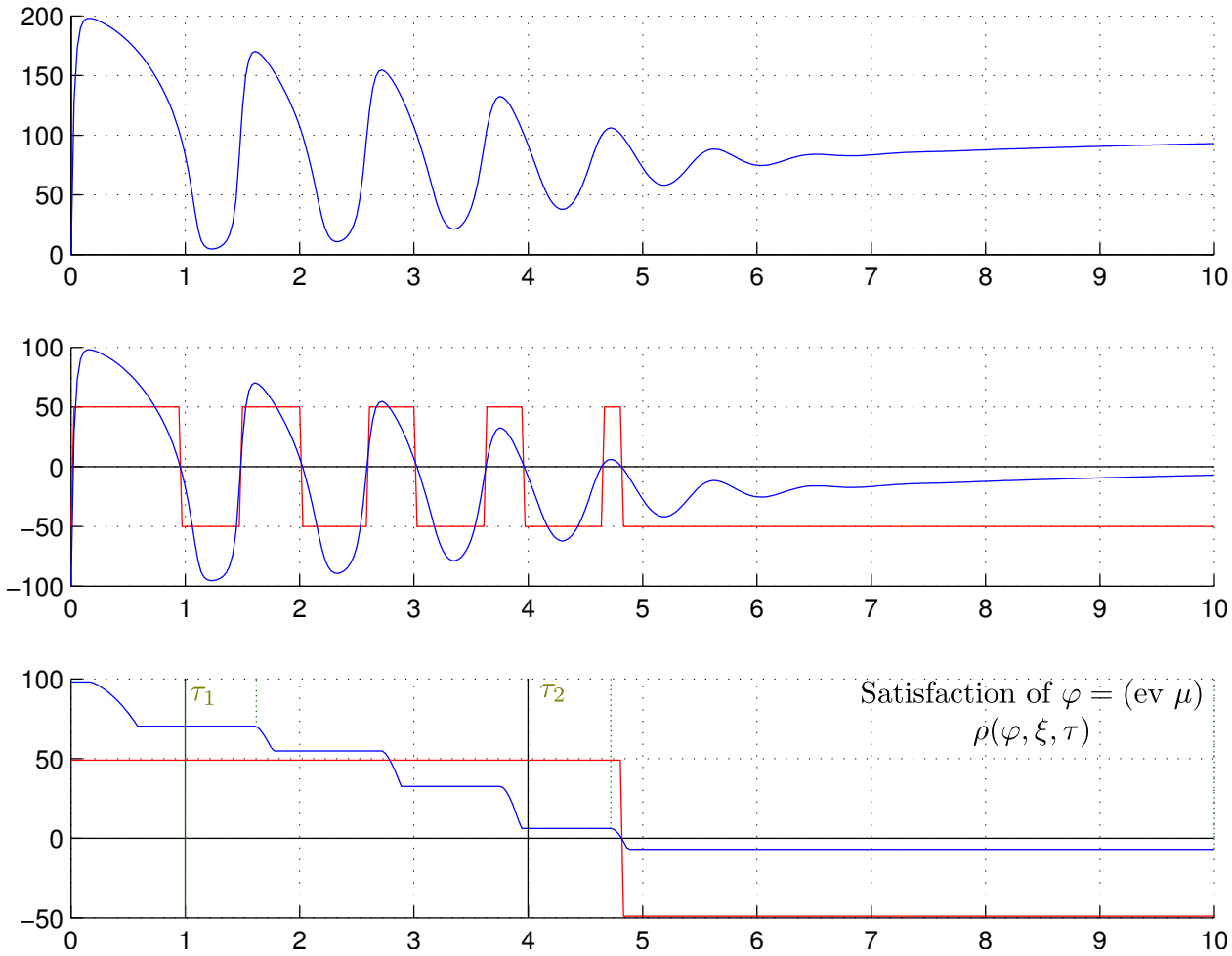
A Simple Formula



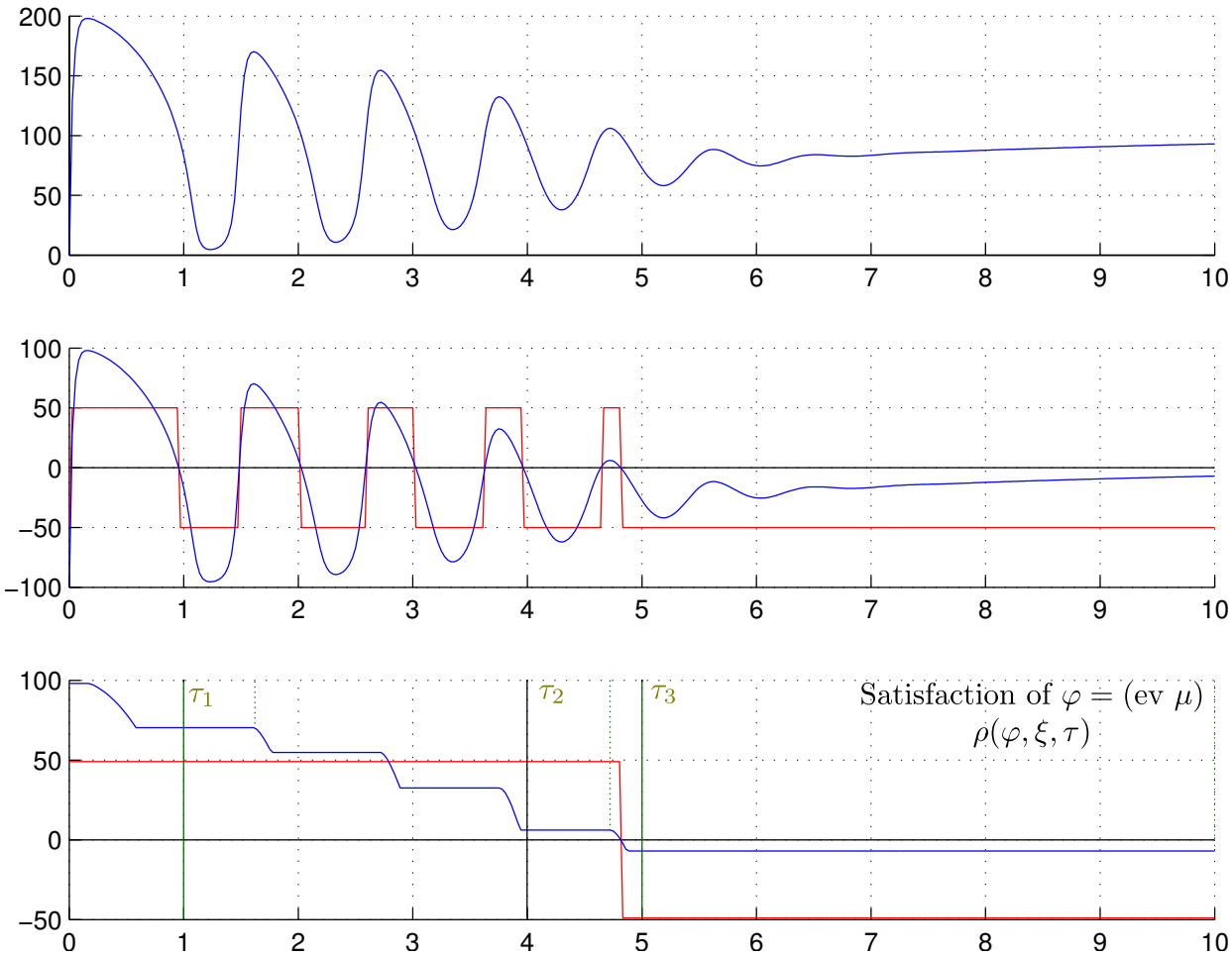
A Simple Formula



A Simple Formula



A Simple Formula



Robust satisfaction and parameters

Example: $\varphi \equiv \mathsf{alw}\ \left[(x(p_1) \geq 2)\ \Rightarrow\ \mathsf{ev}_{[0,p_2]}\ (y \leq 0.1)\right]$

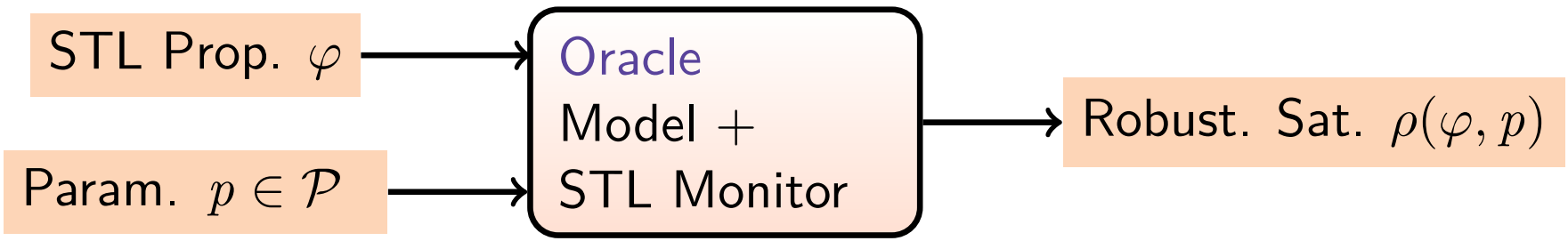
We have the following oracle:

- Satisfiability: $\exists p,\ \rho(\varphi, p) > 0$?
- Max robustness: solution of $\max\ \{\rho(\varphi, p) \mid p \in \mathcal{P}\}$
- Global robustness volume of $\{p \in \mathcal{P} \mid \rho(\varphi, p) > 0\})$?
  If n_p is large, Quasi-Monte Carlo and global sensitivity analysis (eg.: Sobol indices)

Robust satisfaction and parameters

$$\text{Example: } \varphi \equiv \text{alw } \left[(x(p_1) \geq 2) \Rightarrow \text{ev}_{[0,p_2]} (y \leq 0.1)\right]$$

We have the following oracle:

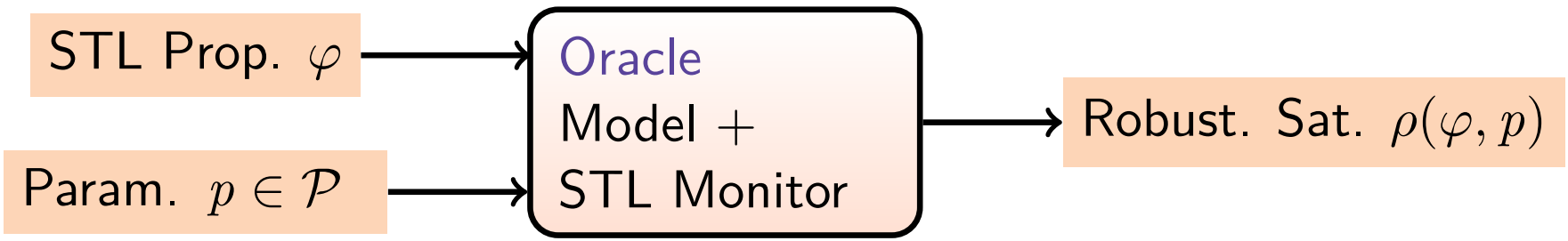


- Satisfiability: $\exists p,\ \rho(\varphi, p) > 0$?
- Max robustness: solution of $\max \{\rho(\varphi, p) \mid p \in \mathcal{P}\}$
- Global robustness volume of $\{p \in \mathcal{P} \mid \rho(\varphi, p) > 0\})$?
  If n_p is large, Quasi-Monte Carlo and global sensitivity analysis (eg.: Sobol indices)

Robust satisfaction and parameters

$$\text{Example: } \varphi \equiv \text{alw } \left[(x(p_1) \geq 2) \Rightarrow \text{ev}_{[0,p_2]} (y \leq 0.1)\right]$$

We have the following oracle:

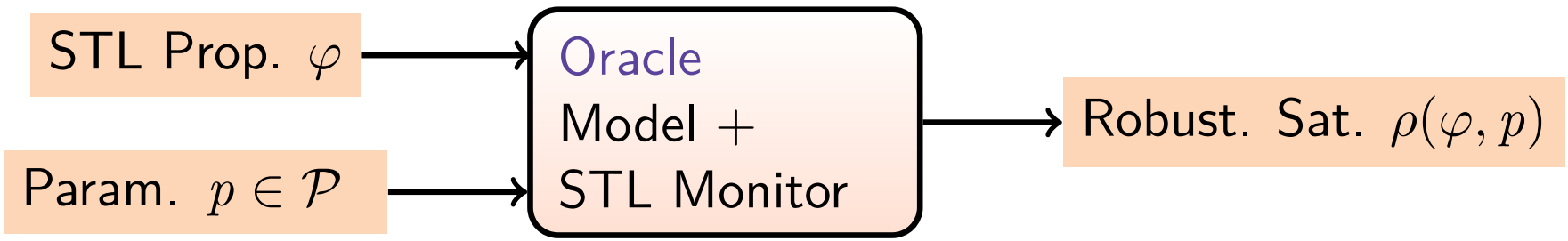


- Satisfiability: $\exists p,\ \rho(\varphi, p) > 0$?
- Max robustness: solution of $\max \{\rho(\varphi, p) \mid p \in \mathcal{P}\}$
- Global robustness volume of $\{p \in \mathcal{P} \mid \rho(\varphi, p) > 0\})$?
  If n_p is large, Quasi-Monte Carlo and global sensitivity analysis (eg.: Sobol indices)

Robust satisfaction and parameters

Example: $\varphi \equiv \mathsf{alw}\ \left[(x(p_1) \geq 2)\ \Rightarrow\ \mathsf{ev}_{[0,p_2]}\ (y \leq 0.1)\right]$

We have the following oracle:

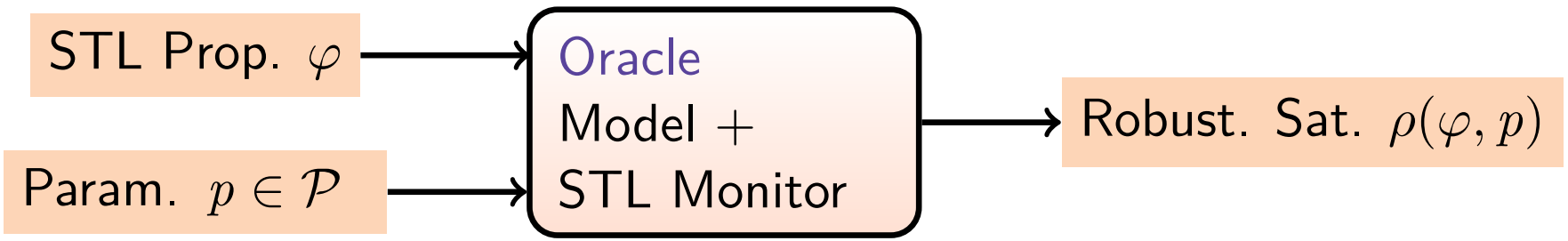


- Satisfiability: $\exists p,\ \rho(\varphi, p) > 0$?
- Max robustness: solution of $\max\ \{\rho(\varphi, p) \mid p \in \mathcal{P}\}$
- Global robustness volume of $\{p \in \mathcal{P} \mid \rho(\varphi, p) > 0\})$?
  If n_p is large, Quasi-Monte Carlo and global sensitivity analysis (eg.: Sobol indices)

Robust satisfaction and parameters

Example: $\varphi \equiv \text{alw}\ \left[(x(p_1) \geq 2)\ \Rightarrow\ \text{ev}_{[0,p_2]}\ (y \leq 0.1)\right]$

We have the following oracle:



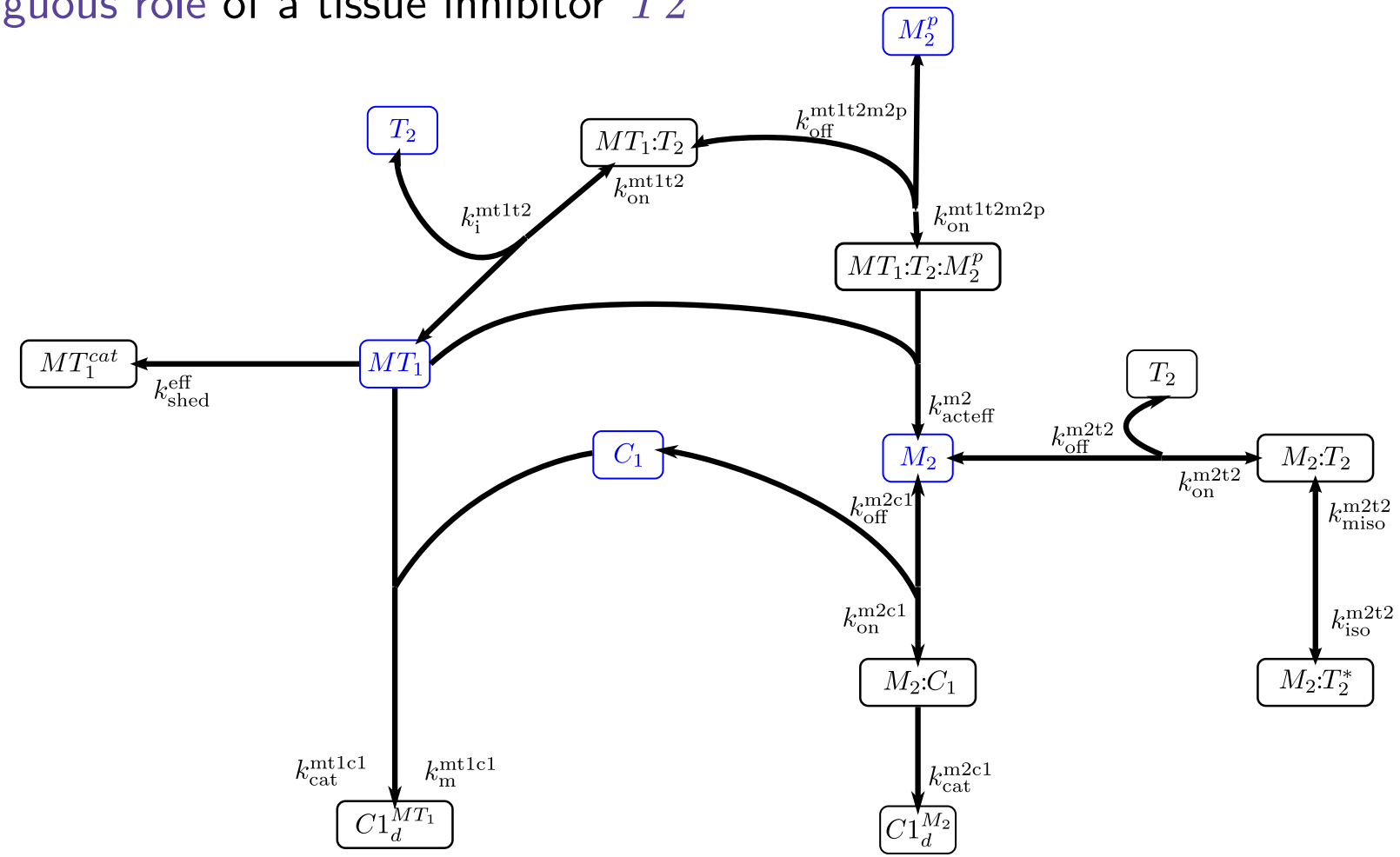
- Satisfiability: $\exists p,\ \rho(\varphi, p) > 0$?
- Max robustness: solution of $\max\ \{\rho(\varphi, p) \mid p \in \mathcal{P}\}$
- Global robustness volume of $\{p \in \mathcal{P} \mid \rho(\varphi, p) > 0\})$?
  If n_p is large, Quasi-Monte Carlo and global sensitivity analysis (eg.: Sobol indices)

Outline

An enzymatic network involved in angiogenesis

Collagen (C_1) degradation by matrix metalloproteinase (M_2^P) and membrane type 1 metalloproteinase (MT_1) [KP04]

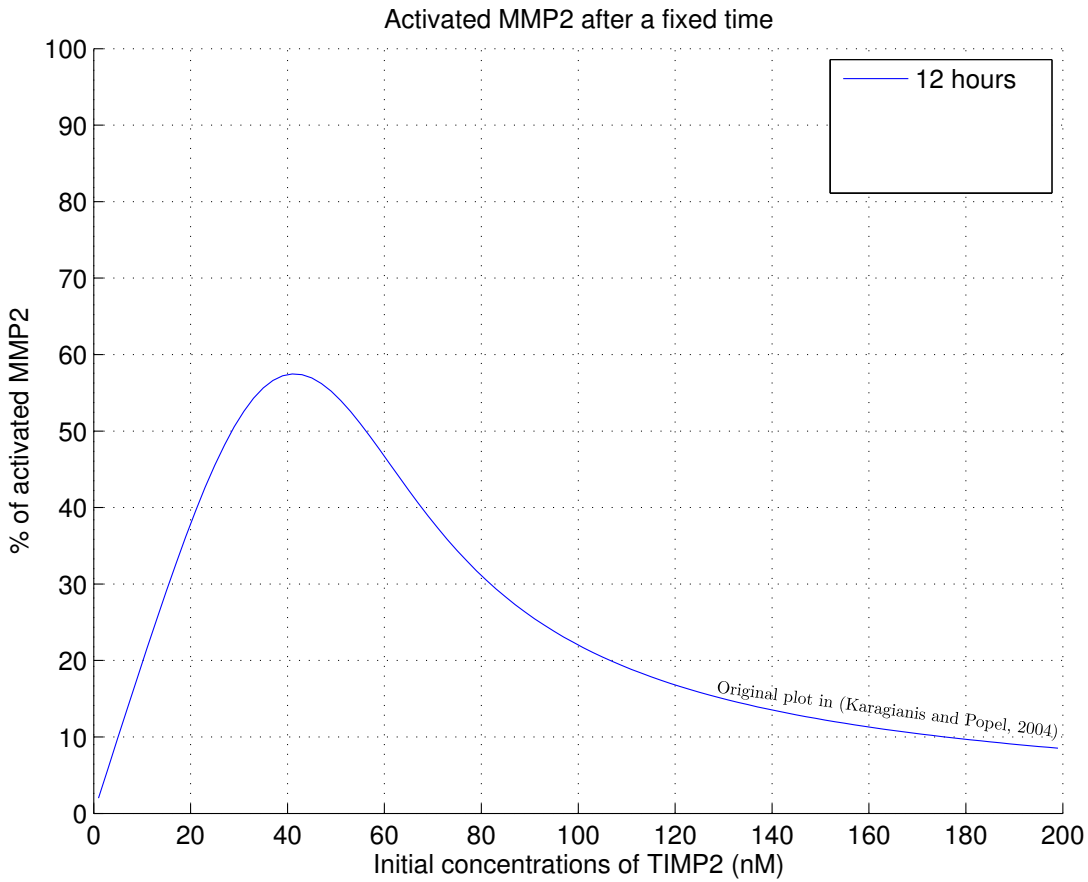
Ambiguous role of a tissue inhibitor $T2$



Rigorous steady state analysis

In [KP04], activation of M_2^P after 12h "Nearly steady state" for $T_2(0)$ between 0 and 200 nM. It turned out that steady state was not reached for $T_2(0) > 20$ nM !

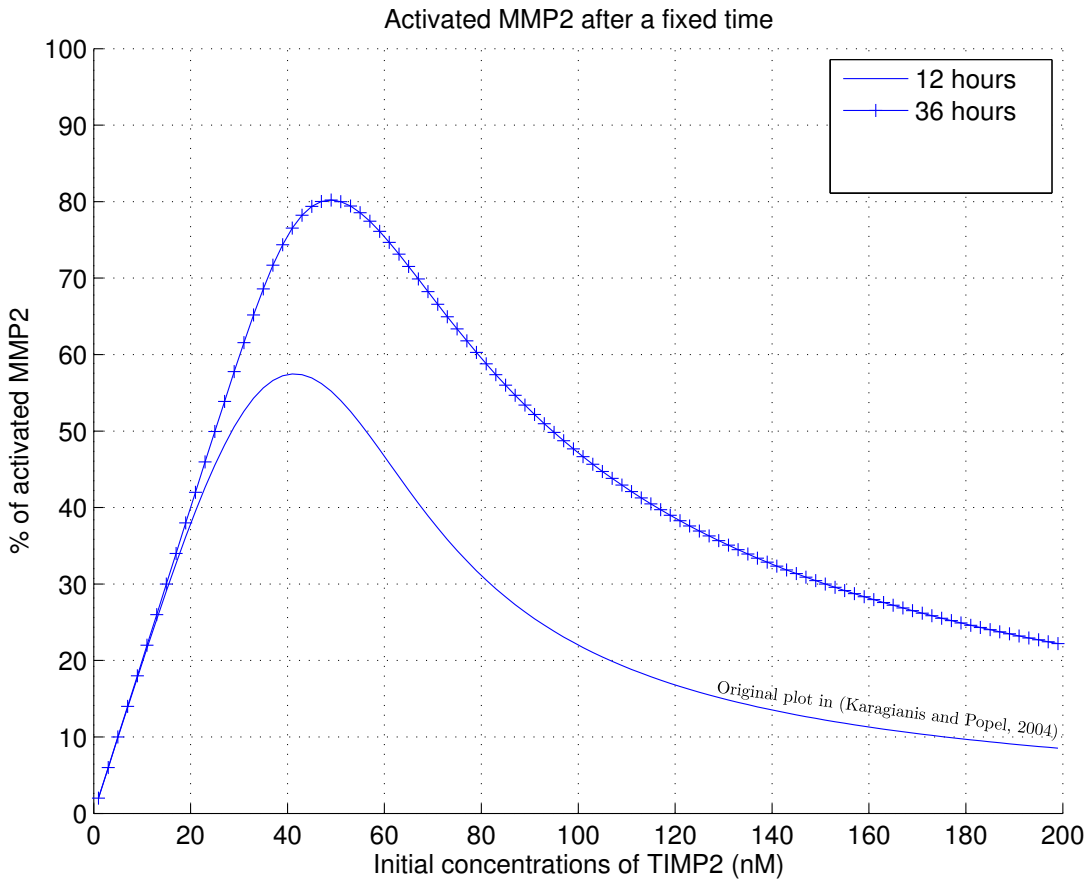
Using $\varphi \equiv$ ev alw $(|\dot{M}_2(t)| < \epsilon \times M_2^P(0))$ we could guarantee the correct plot.



Rigorous steady state analysis

In [KP04], activation of M_2^P after 12h "Nearly steady state" for $T_2(0)$ between 0 and 200 nM. It turned out that steady state was not reached for $T_2(0) > 20$ nM !

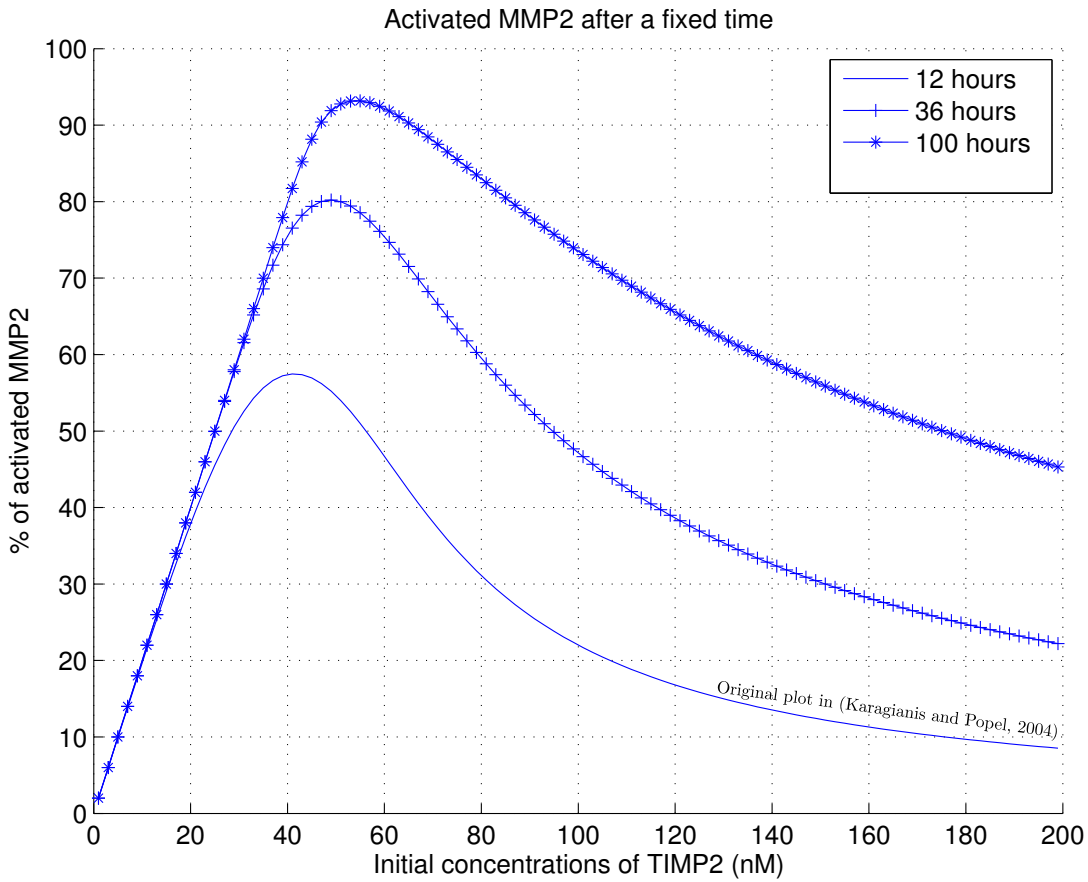
Using $\varphi \equiv$ ev alw $(|\dot{M}_2(t)| < \epsilon \times M_2^P(0))$ we could guarantee the correct plot.



Rigorous steady state analysis

In [KP04], activation of M_2^P after 12h "Nearly steady state" for $T_2(0)$ between 0 and 200 nM. It turned out that steady state was not reached for $T_2(0) > 20$ nM !

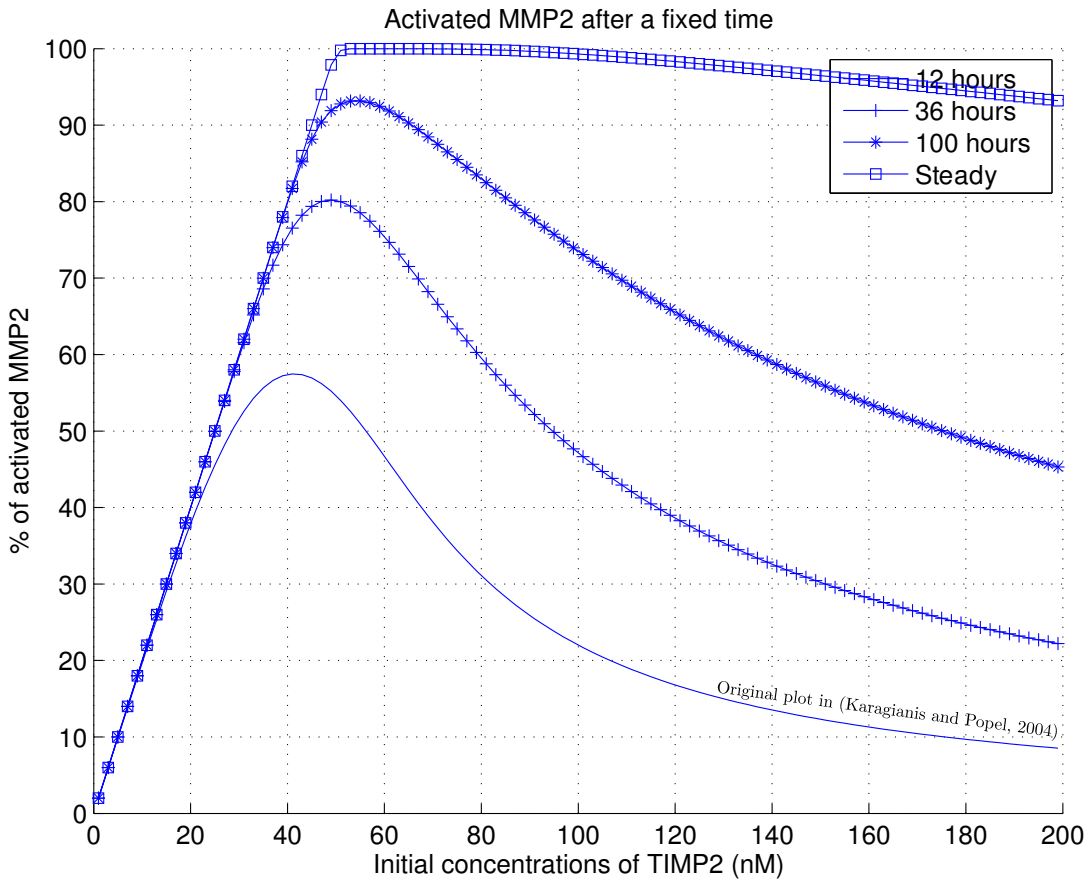
Using $\varphi \equiv$ ev alw $(|\dot{M}_2(t)| < \epsilon \times M_2^P(0))$ we could guarantee the correct plot.



Rigorous steady state analysis

In [KP04], activation of M_2^P after 12h "Nearly steady state" for $T_2(0)$ between 0 and 200 nM. It turned out that steady state was not reached for $T_2(0) > 20$ nM !

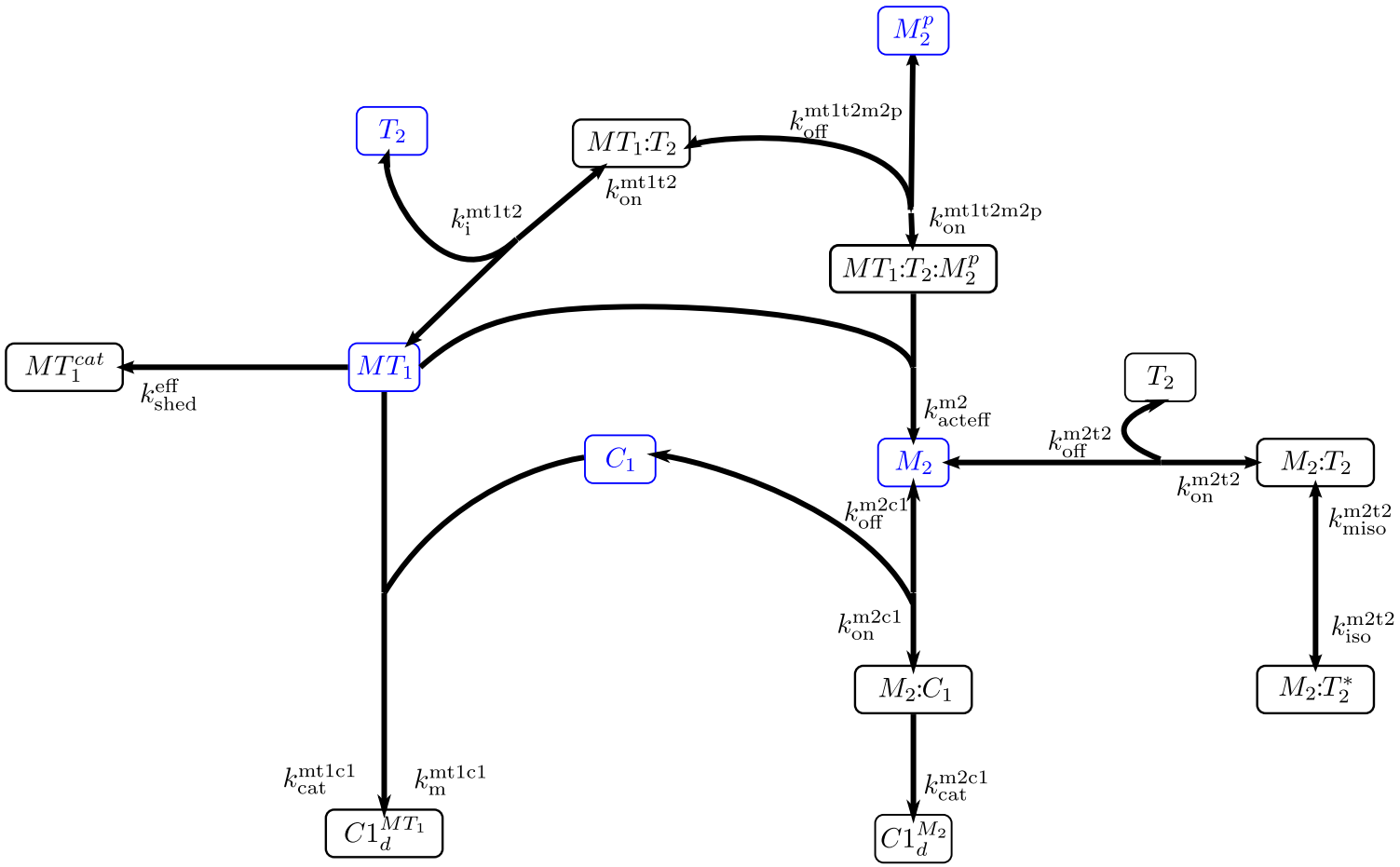
Using $\varphi \equiv$ ev alw $(|\dot{M}_2(t)| < \epsilon \times M_2^P(0))$ we could guarantee the correct plot.



Formalizing synergism

Collagen can be degraded either by MT_1 or by M_2. We defined a notion of synergism by :

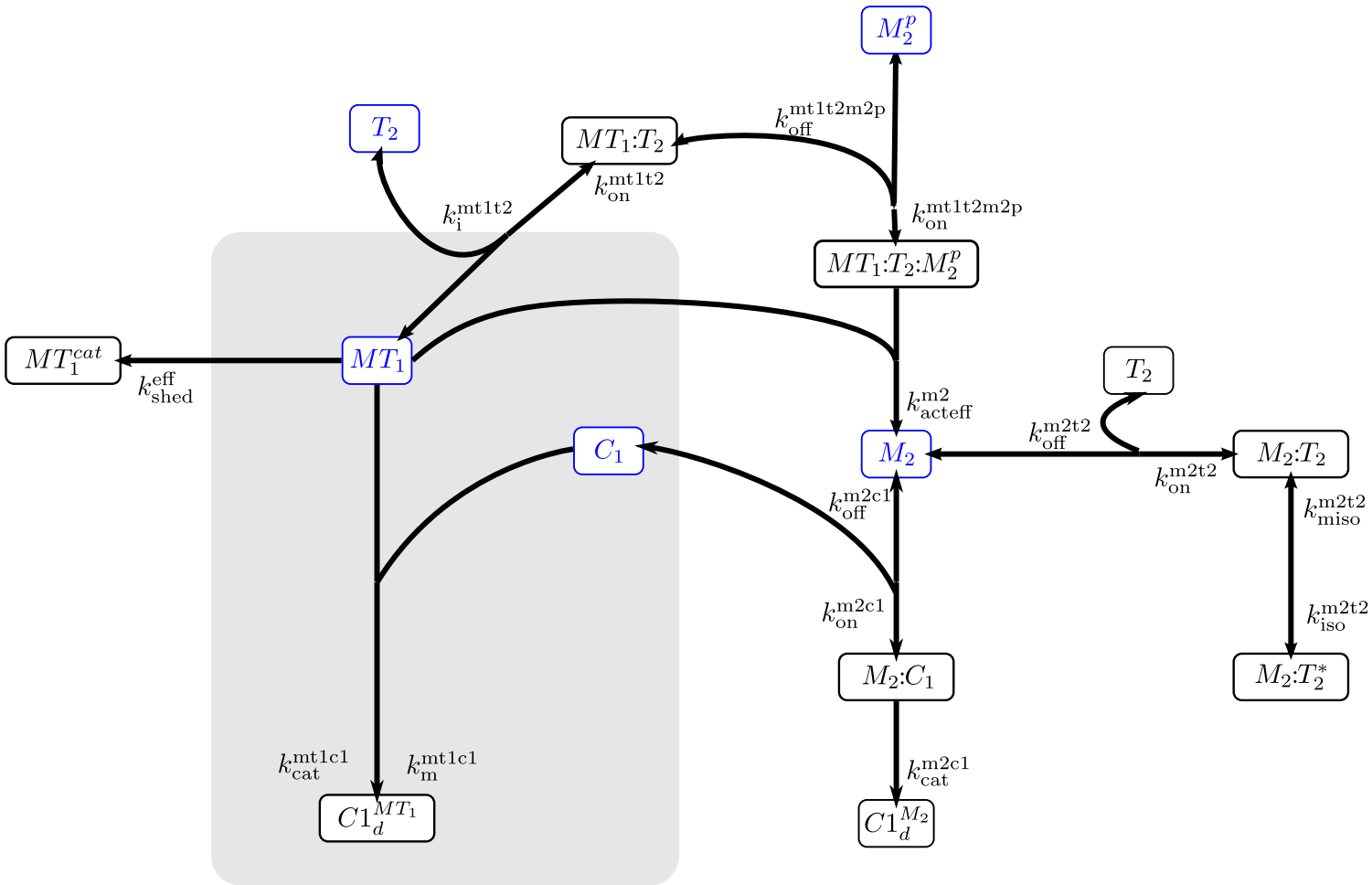
"Before 12h, 90 % of initial collagen is degraded: $\mathsf{ev}_{[0,12h]}(C_1(\tau)/C_1(0) < 0.1)$ and at least 50 % by $M2$: $\mathsf{ev}_{[0,12h]}(C1_d^{M_2}(\tau) > C1_d^{MT_1}(\tau))$ "



Formalizing synergism

Collagen can be degraded either by MT_1 or by M_2. We defined a notion of synergism by :

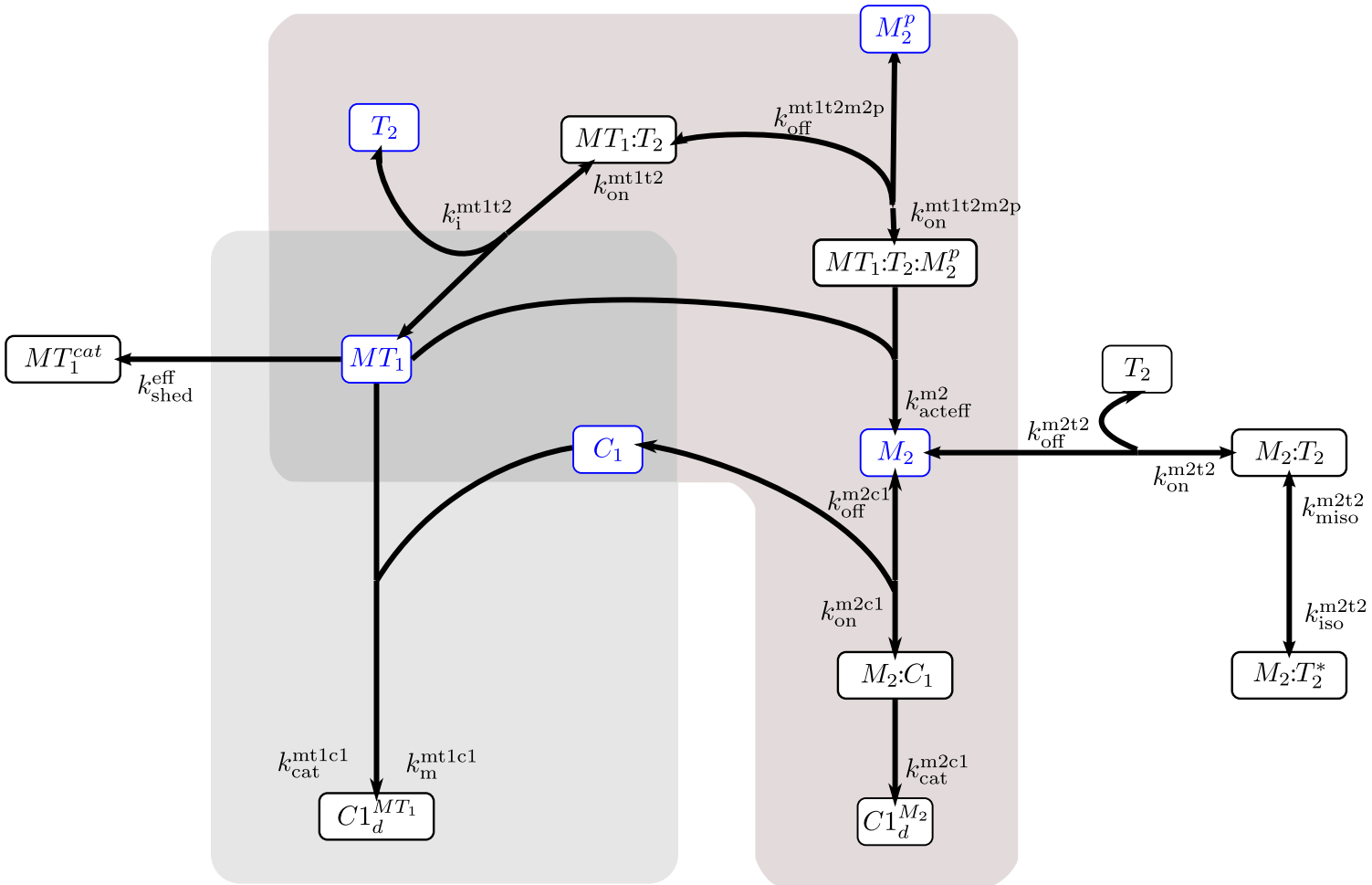
"Before 12h, 90 % of initial collagen is degraded: $\mathrm{ev}_{[0,12h]}(C_1(\tau)/C_1(0) < 0.1)$ and at least 50 % by $M2$: $\mathrm{ev}_{[0,12h]}(C1_d^{M_2}(\tau) > C1_d^{MT_1}(\tau))$ "



Formalizing synergism

Collagen can be degraded either by MT_1 or by M_2. We defined a notion of synergism by :

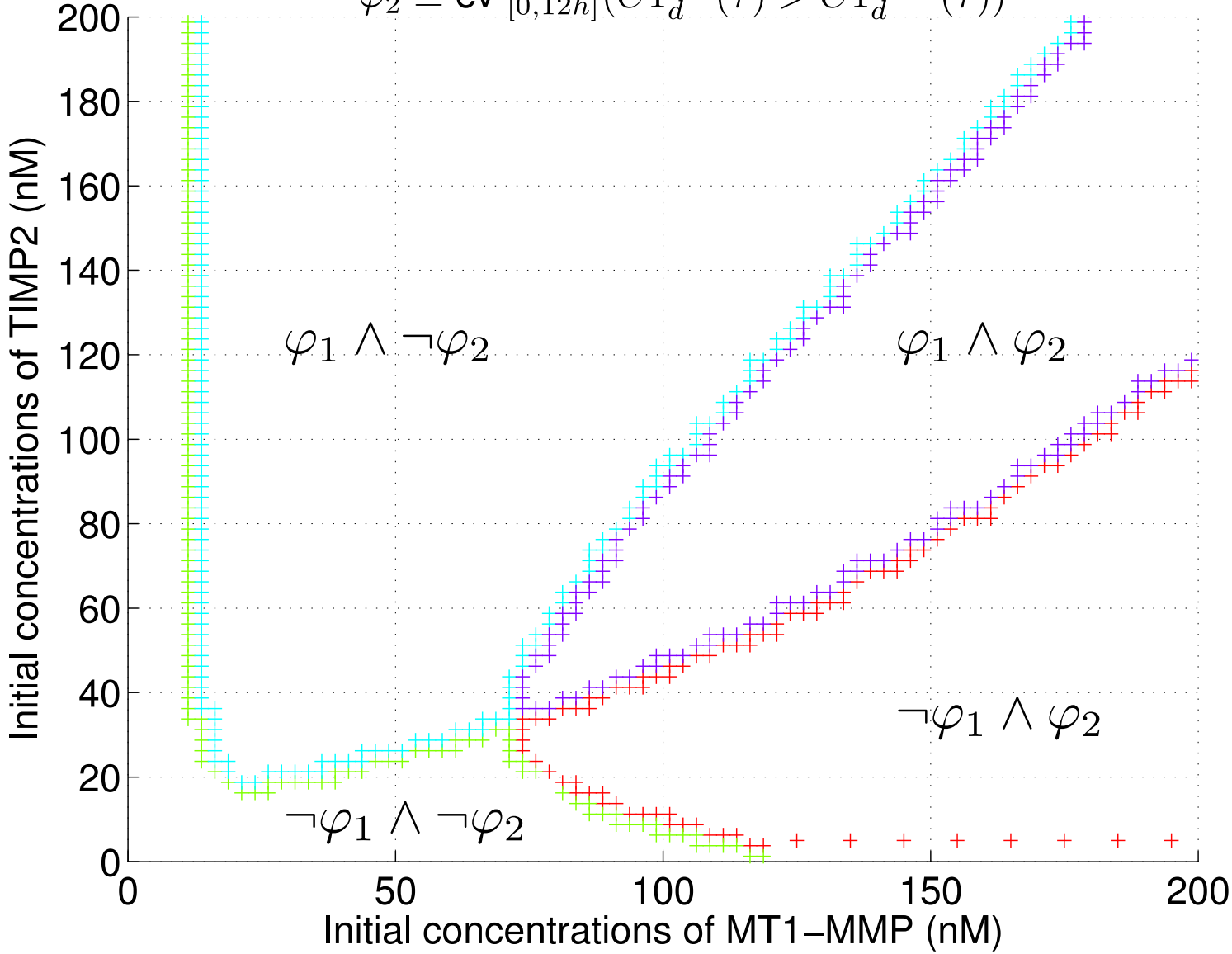
"Before 12h, 90 % of initial collagen is degraded: $\mathsf{ev}_{[0,12h]}(C_1(\tau)/C_1(0) < 0.1)$ and at least 50 % by $M2$: $\mathsf{ev}_{[0,12h]}(C1_d^{M_2}(\tau) > C1_d^{MT_1}(\tau))$ "



Synergism, result

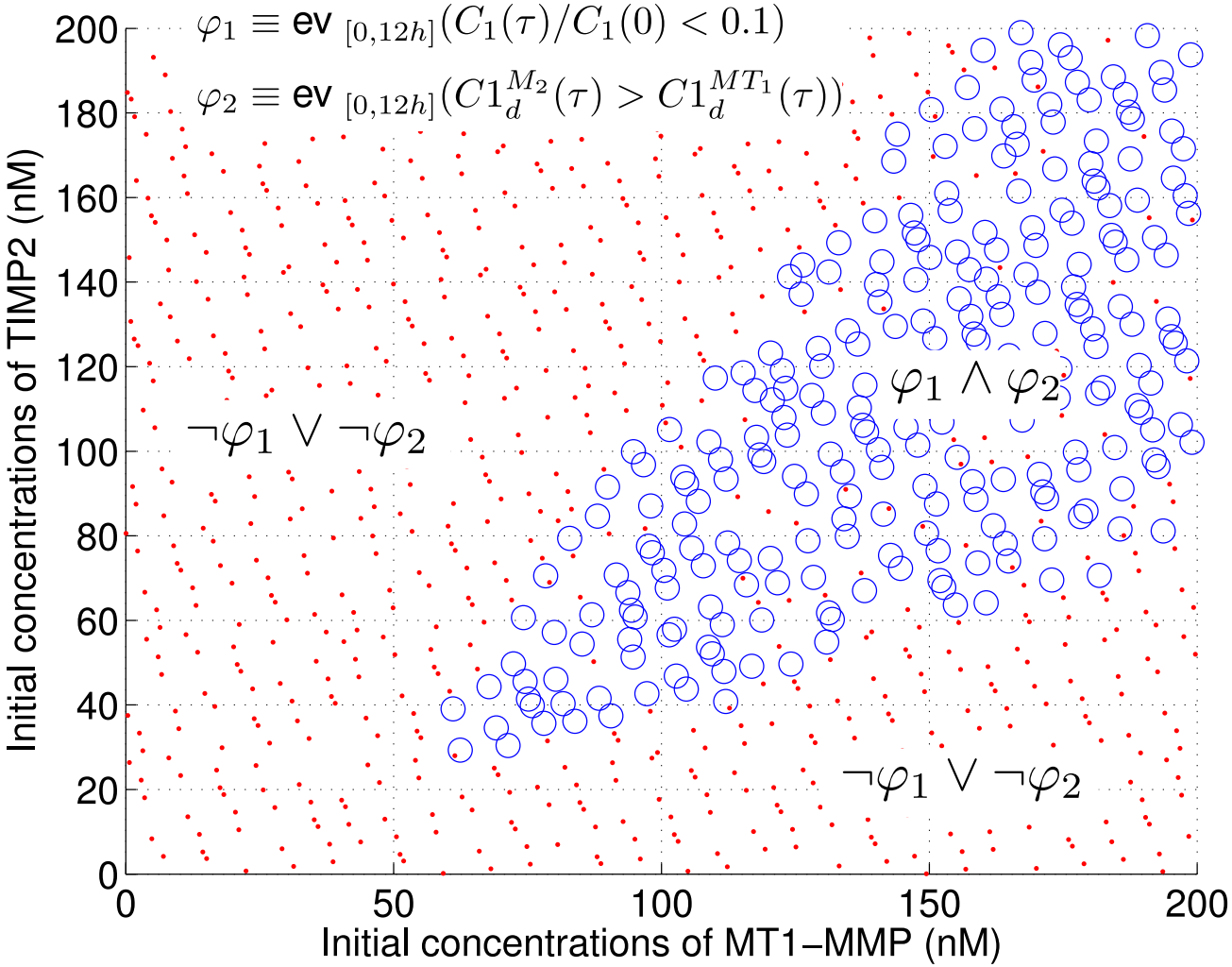
$$\varphi_1 \equiv \text{ev}_{[0,12h]}(C_1(\tau)/C_1(0) < 0.1)$$

$$\varphi_2 \equiv \text{ev}_{[0,12h]}(C1_d^{M_2}(\tau) > C1_d^{MT_1}(\tau))$$



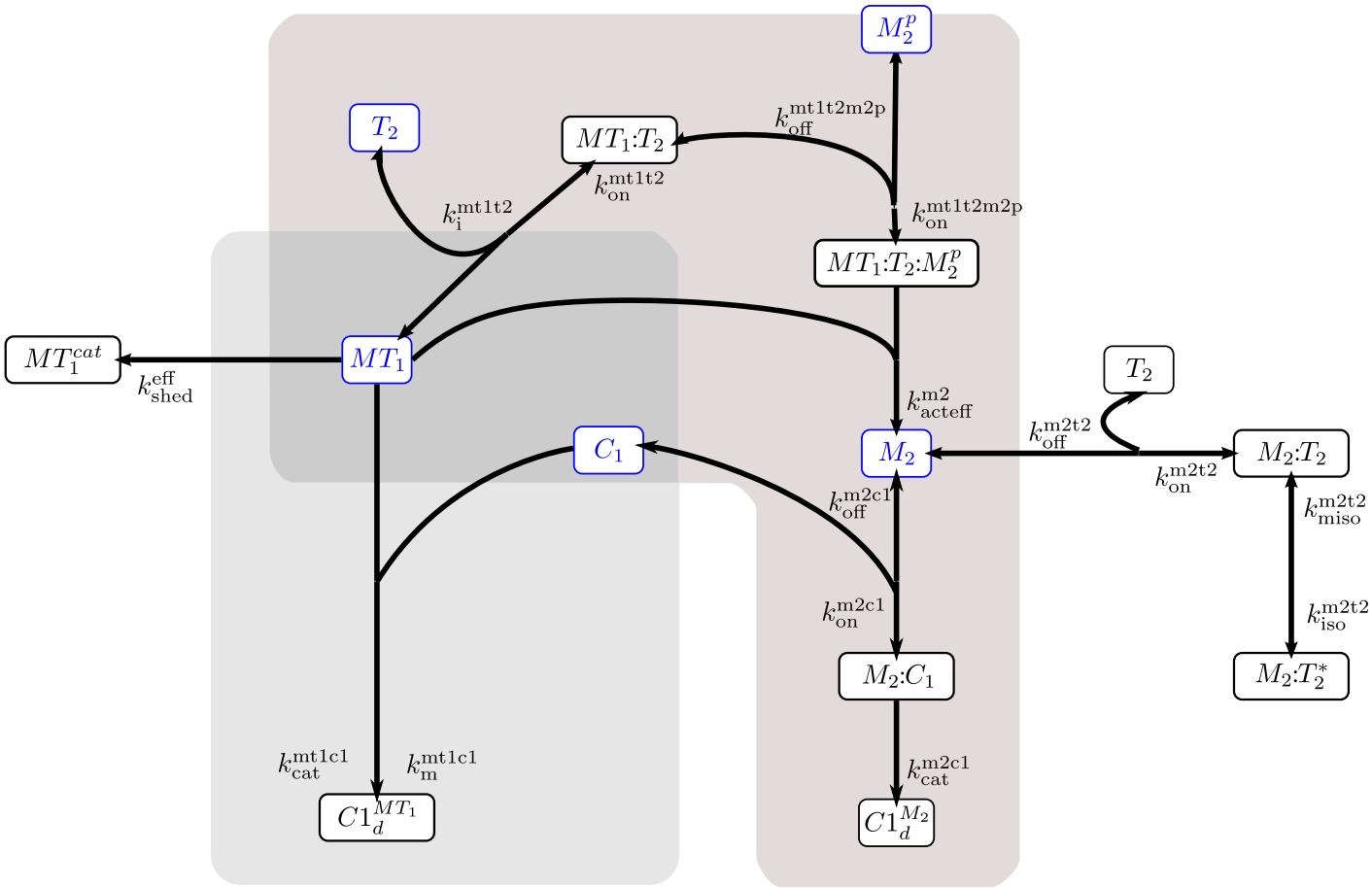
Synergism, global analysis

Varying all other parameters around 10% of nomimal value, and using quasi-Monte-Carlo sampling, we measure the robustness of the regions found



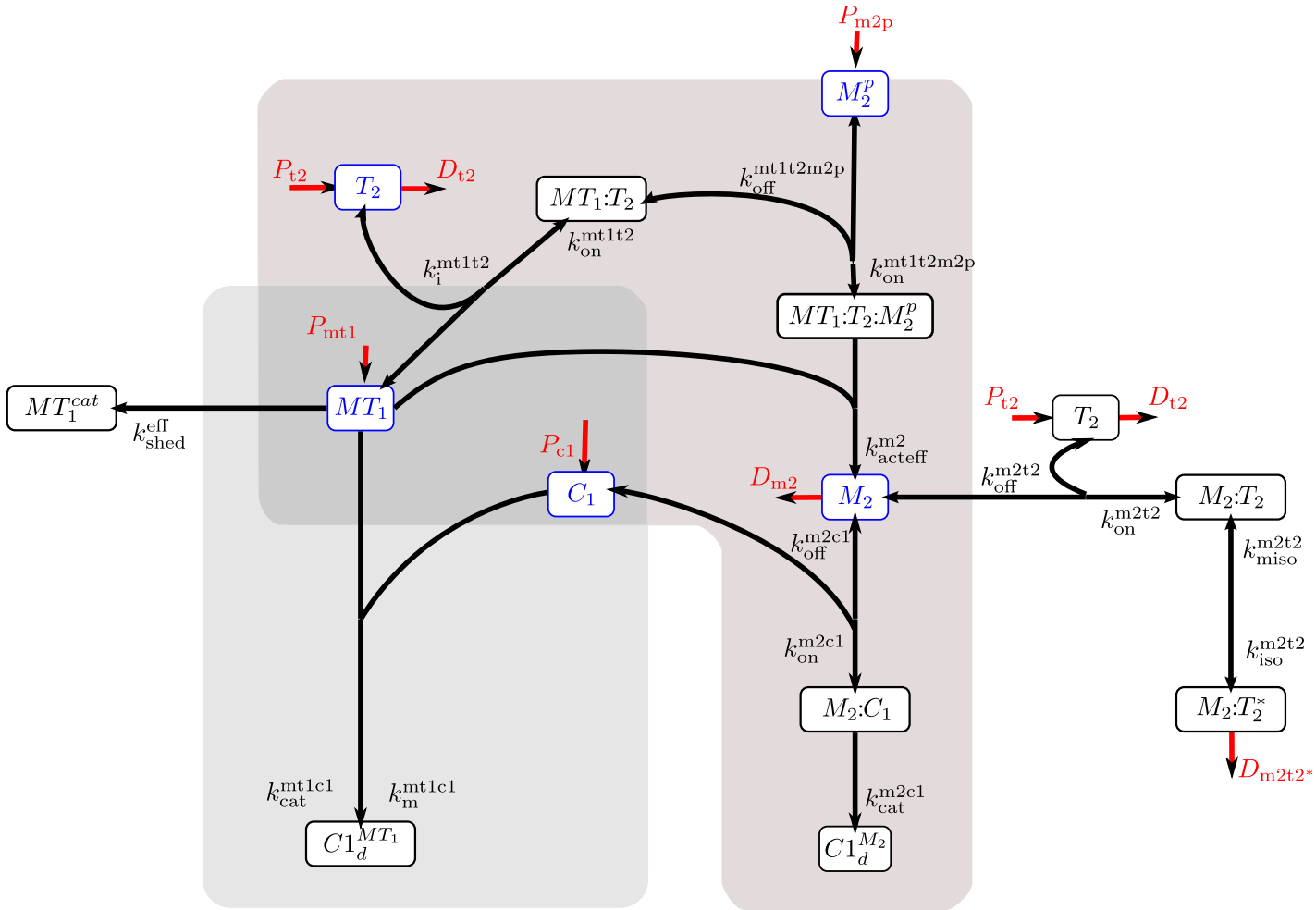
Open Model

To extend the model, we introduced production and degradation terms



Open Model

To extend the model, we introduced production and degradation terms



Detecting oscillations in M_2^P

We used the formula

$$\varphi_{\neg div} = \mathsf{alw}(M_2^P[t] < M_{2\mathsf{max}}^P)$$

to guarantee that the oscillation remains in a given range of amplitudes, in conjunction with

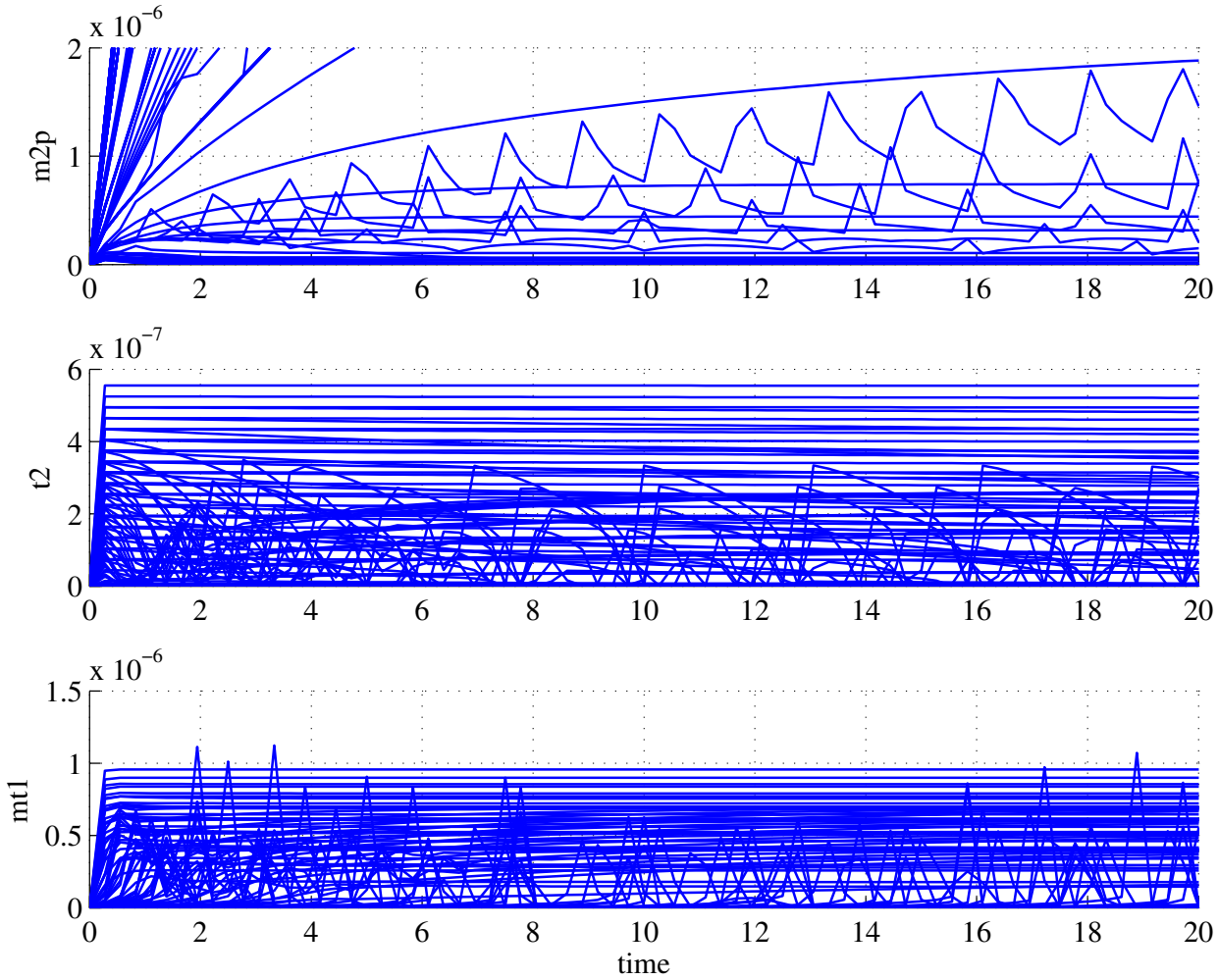
$$\mathsf{ev}\ \mathsf{alw}\ \left(\mathsf{ev}_{[0,6h)}\ \left(\frac{dM_2^P}{dt}[t] > k_h \wedge \mathsf{ev}_{[0,6h)}\ \left(\frac{dM_2^P}{dt}[t] < k_l\right)\right)\right)$$

The first "eventually" removes the transient phase before the oscillations and the "always" filters damped oscillations

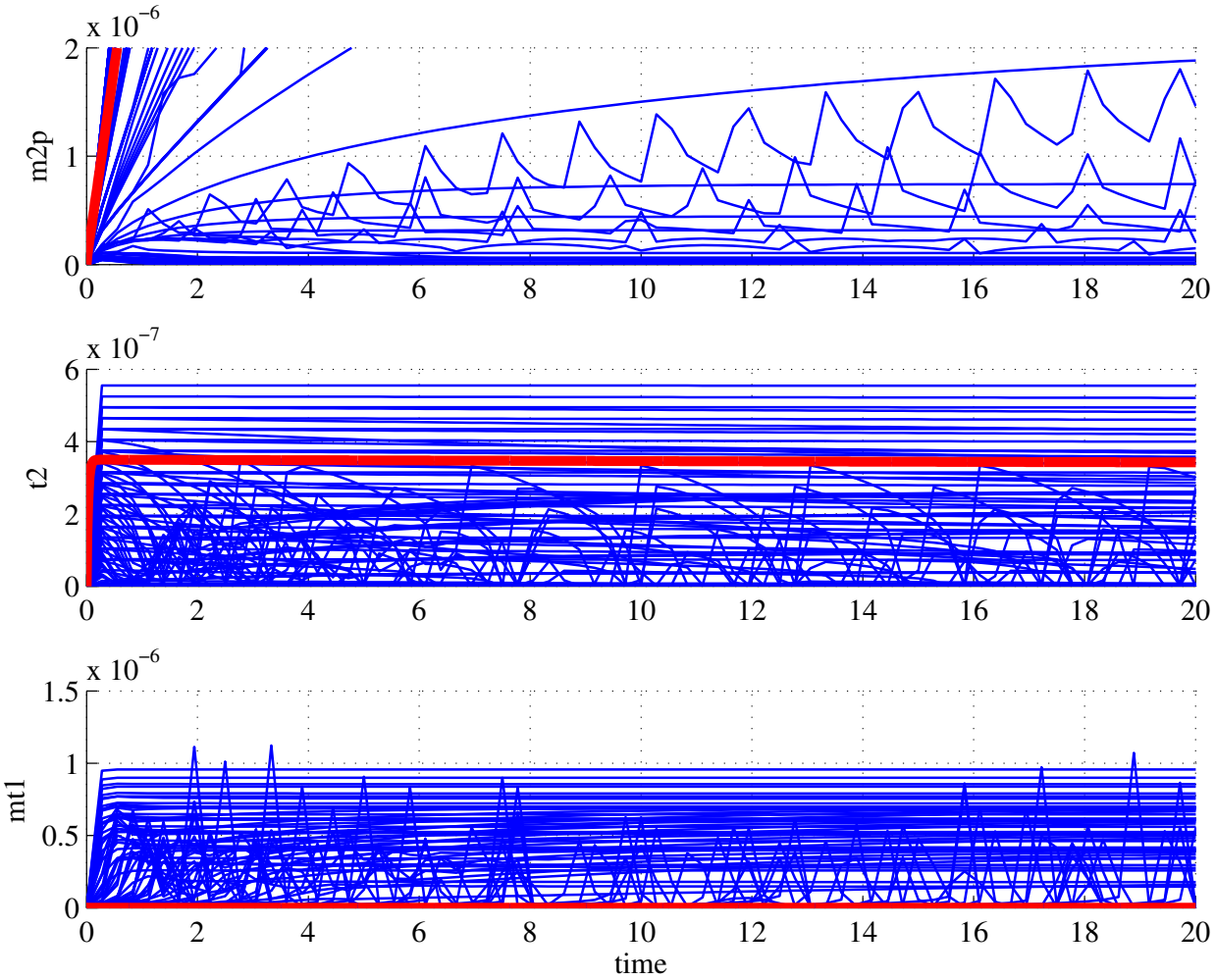
Then requires that the concentration of M_2^P alternates between periods when the it strictly increases and periods when it strictly decreases

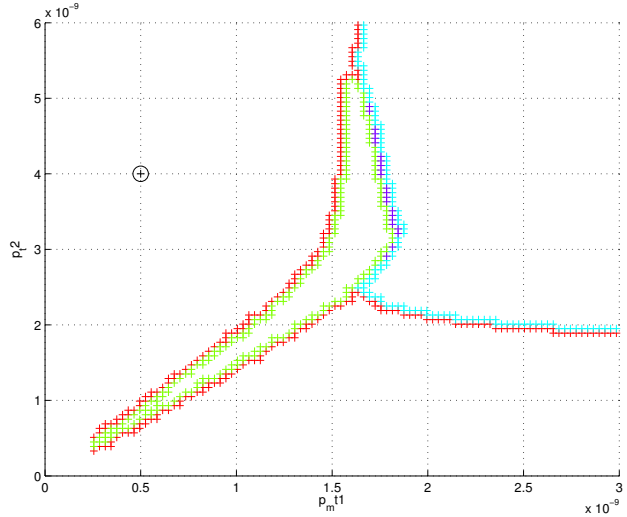
The formula filters oscillations with a period greater than $12h$

Oscillations Map

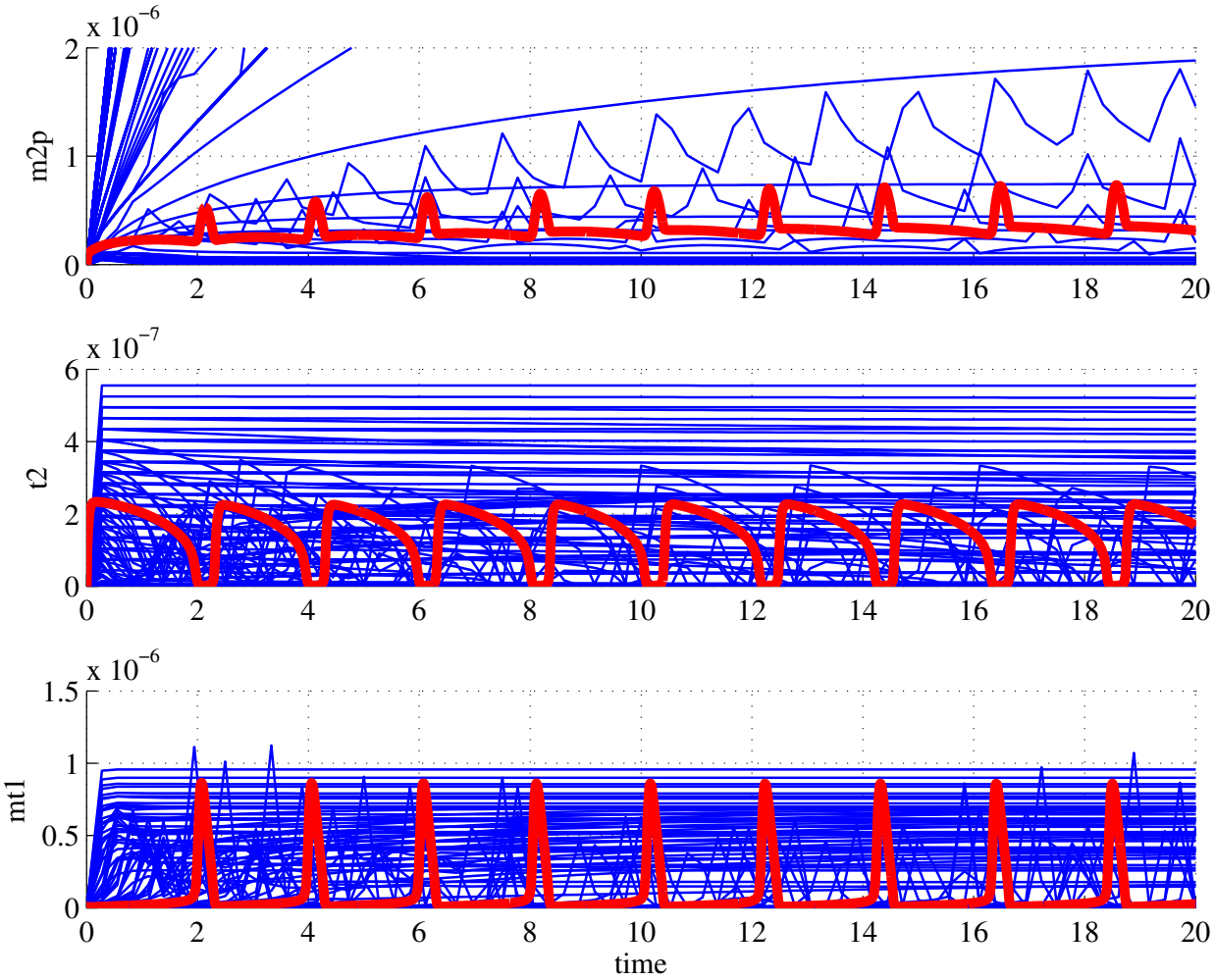


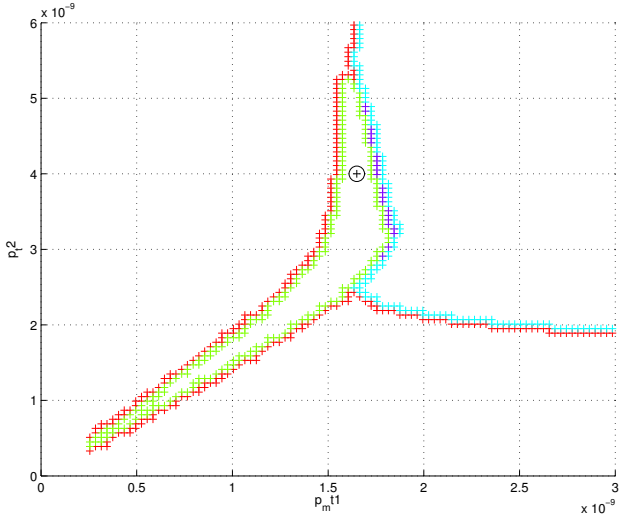
Oscillations Map



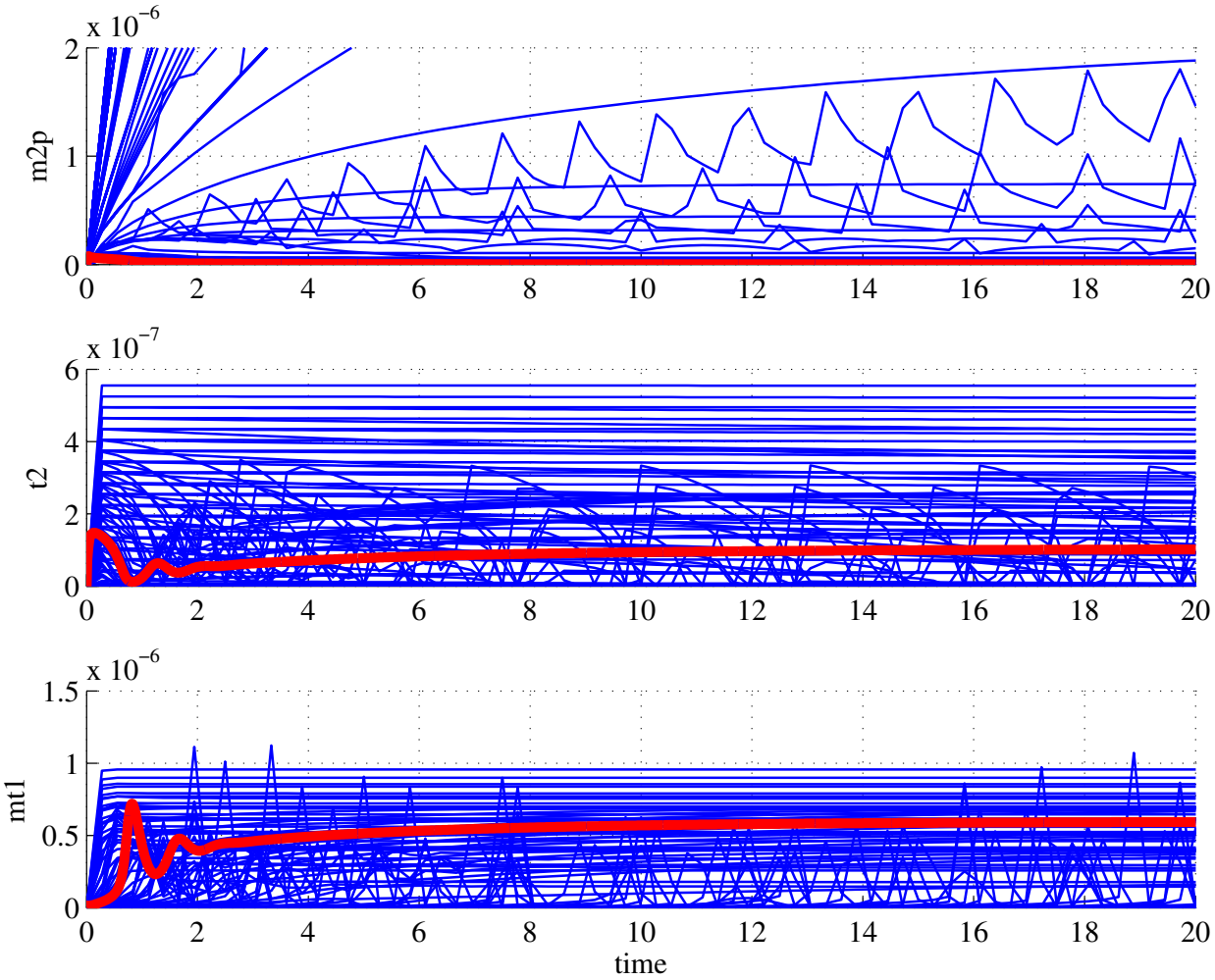


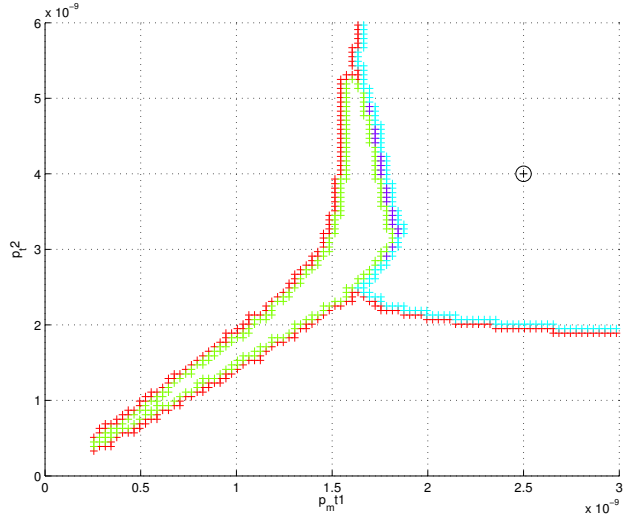
Oscillations Map



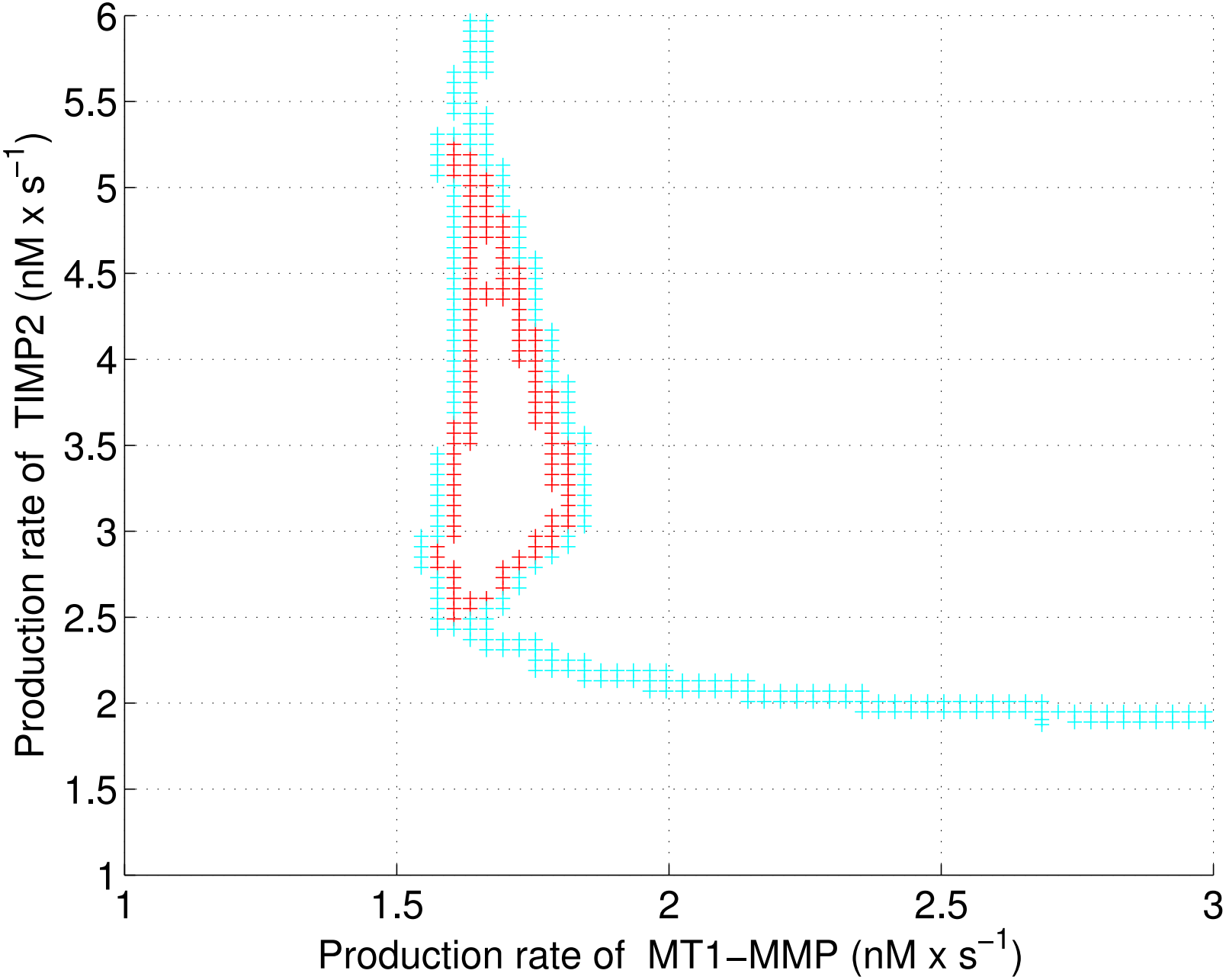


Oscillations Map

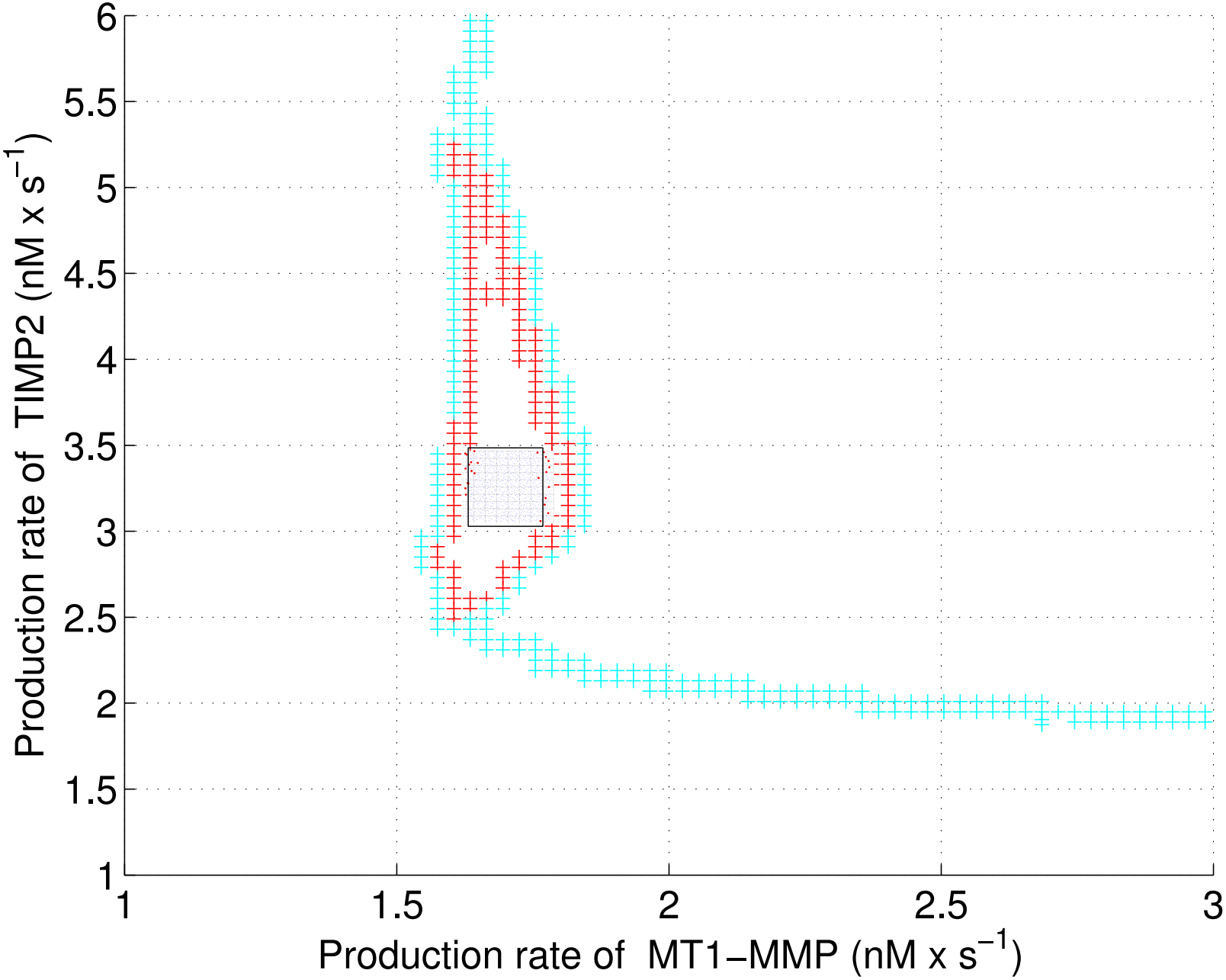




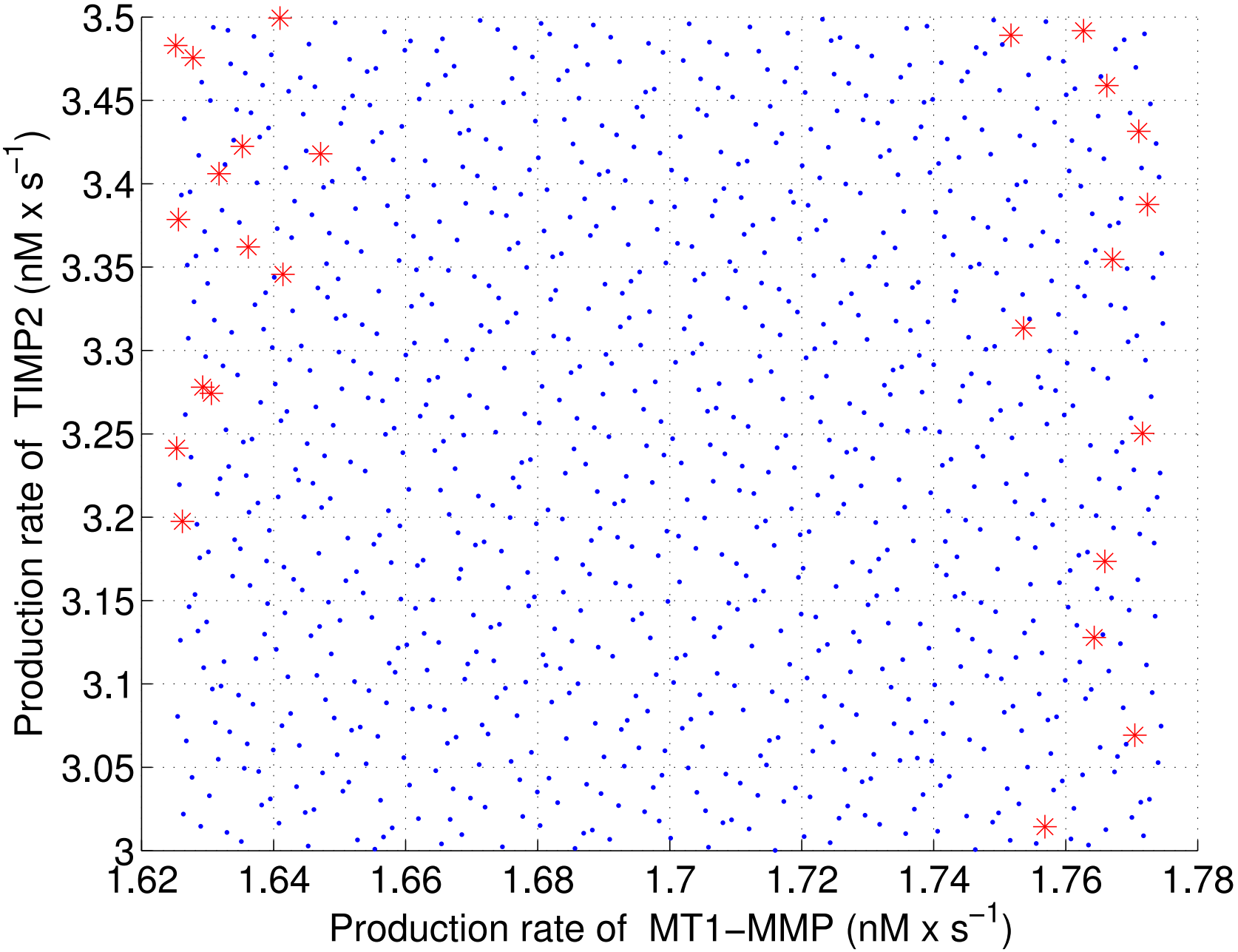
Oscillation, Robustness



Oscillation, Robustness



Oscillation, Robustness



Summary

This work combines classical dynamical systems theory:

- Deterministic models of ordinary differential equations
- Uncertain initial conditions and parameters
- Numerical simulation, local and global sensitivity analysis

with

- A convenient query language to specify spatial and temporal constraints on variables and parameters
- A satisfaction function which computes by how much a simulation satisfies or violate a property
- Heuristics to synthesize sets of parameters generating trajectories satisfying a property

Summary

This work combines classical dynamical systems theory:

- Deterministic models of ordinary differential equations
- Uncertain initial conditions and parameters
- Numerical simulation, local and global sensitivity analysis

with

- A convenient query language to specify spatial and temporal constraints on variables and parameters
- A satisfaction function which computes by how much a simulation satisfies or violate a property
- Heuristics to synthesize sets of parameters generating trajectories satisfying a property

Bibliography

A. Donzé, C.J. Langmead, G. Clermont
Parameter Synthesis in Nonlinear Dynamical Systems: Application to Systems Biology. Journal of Computational Biology, 2010.

A. Donzé, O. Maler
Robust Satisfaction of Temporal Logic over Real-Valued Signals, FORMATS'10.

A. Donzé, E. Fanchon, L. Gatepaille, O. Maler, P. Tracqui
Robustness Analysis and Behavior Discrimination in Enzymatic Reaction Networks, Submitted

Toward Systems Biology, Grenoble 30-31 Mai 1er Juin

http://www-tsb-workshop.imag.fr