Hybrid stochastic systems

Marc Bouissou

DCDS 2013 - York

Outline

- What is a hybrid stochastic system?
- The need for studying such systems
- A mathematical framework: PDMP
- Monte Carlo simulation schemes
- Modeling Tools: benchmarks
- Two new approaches
- Conclusion: the way to the perfect tool

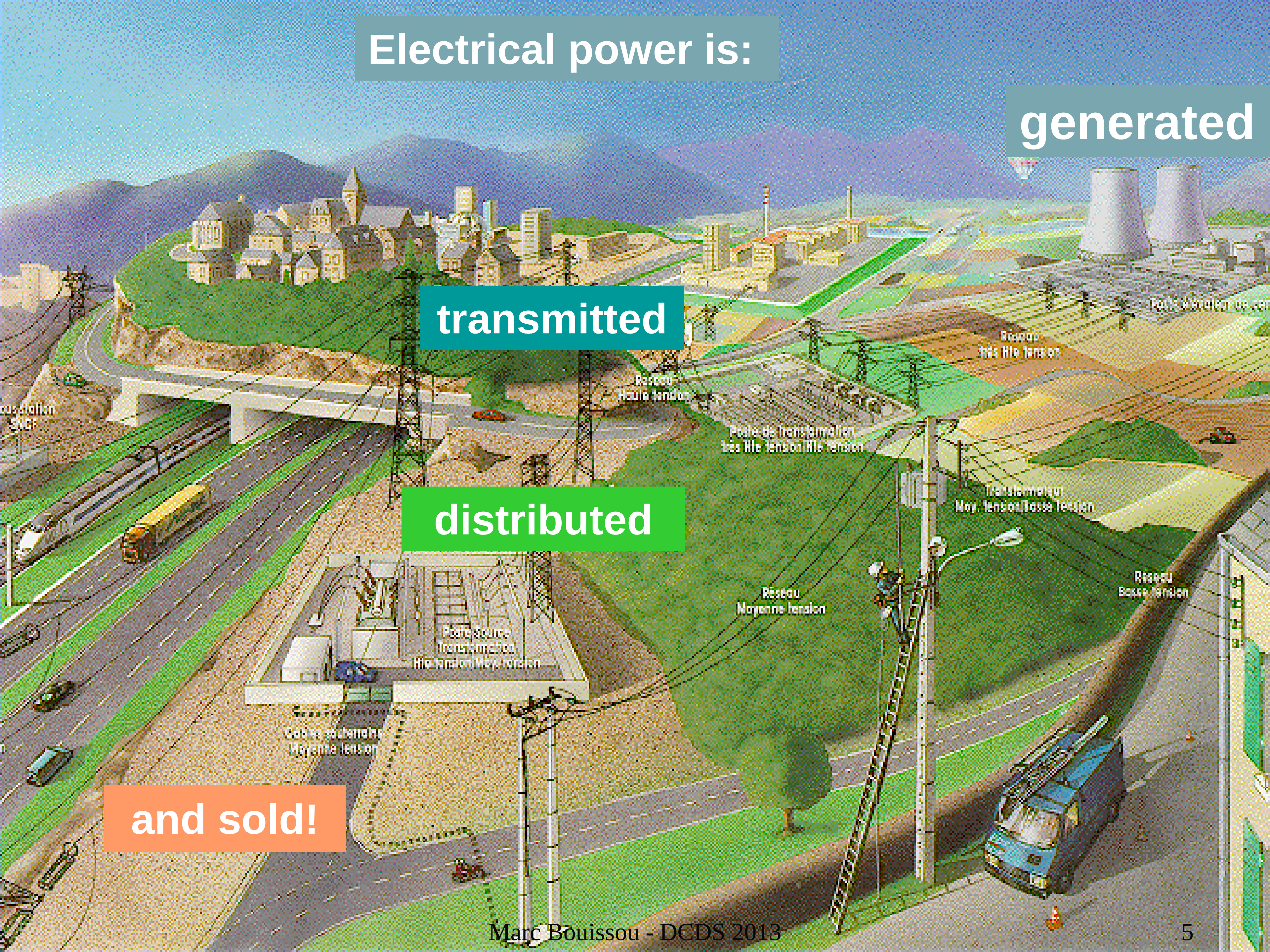
What is a hybrid stochastic system?

- Hybrid
  - Continuous / discrete
  - Most systems are of that kind
  - Few of them cannot be simplified as discrete (for dependability aspects)
- Stochastic
  - Subject to random processes (failures, repairs…)

The need for studying them

- Strong interactions between the continuous and discrete processes
  - Continuous models are not adapted
  - Discrete models are insufficient
- High stakes
  - Safety (nuclear power plants, oil & gas…)
  - Money (electrical grid)

Electrical power is:

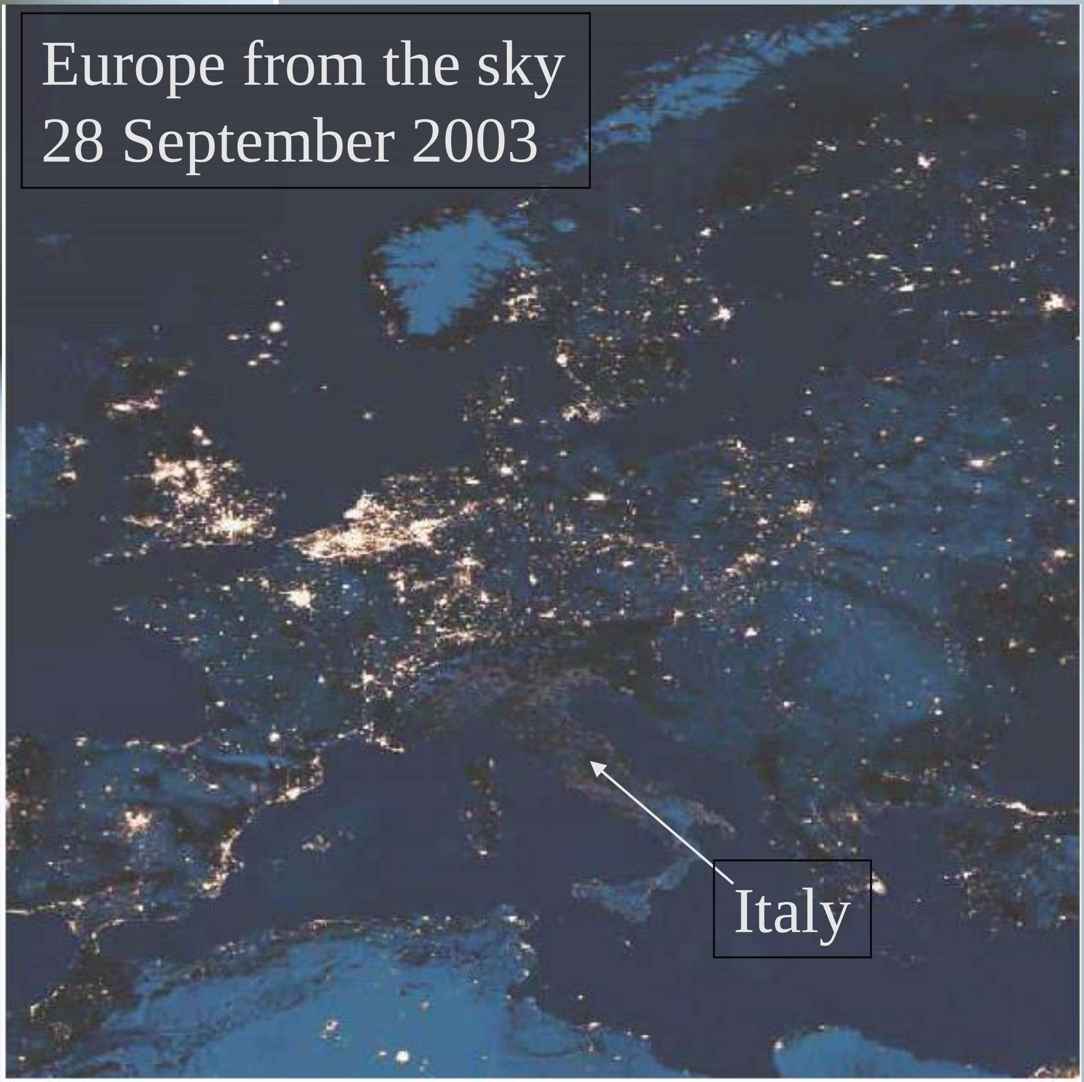


- generated
- transmitted
- distributed
- and sold!

Limits of discrete models

- Discrete stochastic models are generally sufficient for generation and distribution
- Sometimes, analysts are even satisfied with combinatorial models
- But for transmission…

Black outs

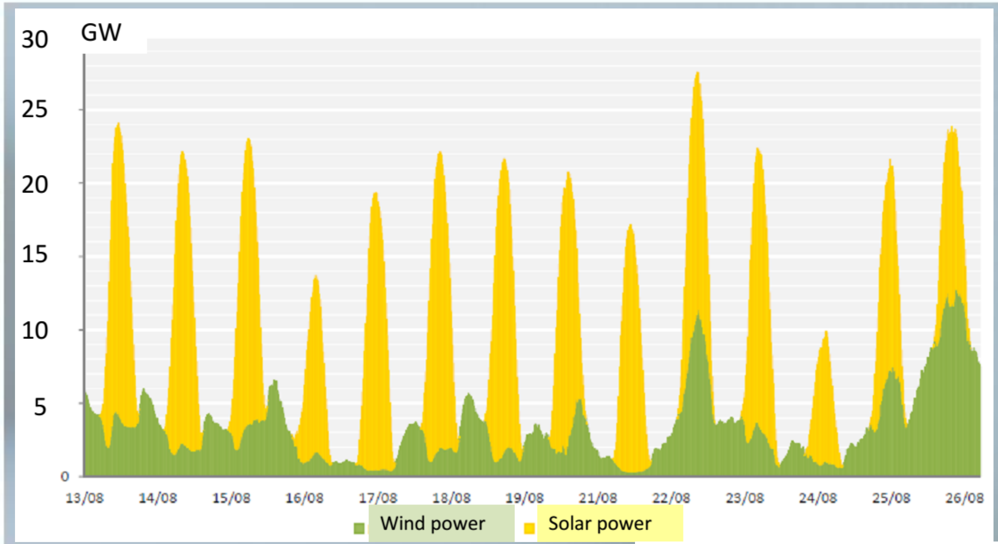


- Loss of power for a large number of customers (>100000 ?)
  - Duration: 1h ? to several days
  - Spread: a middle sized town? to a whole interconnected grid

Black out stakes

- Virtually no casualties until now, **but**
  - a black out => strong constraints on network components (economic stake)
  - electricity not sold – never recovered
  - the safety of nuclear power plants is weakened during the black out

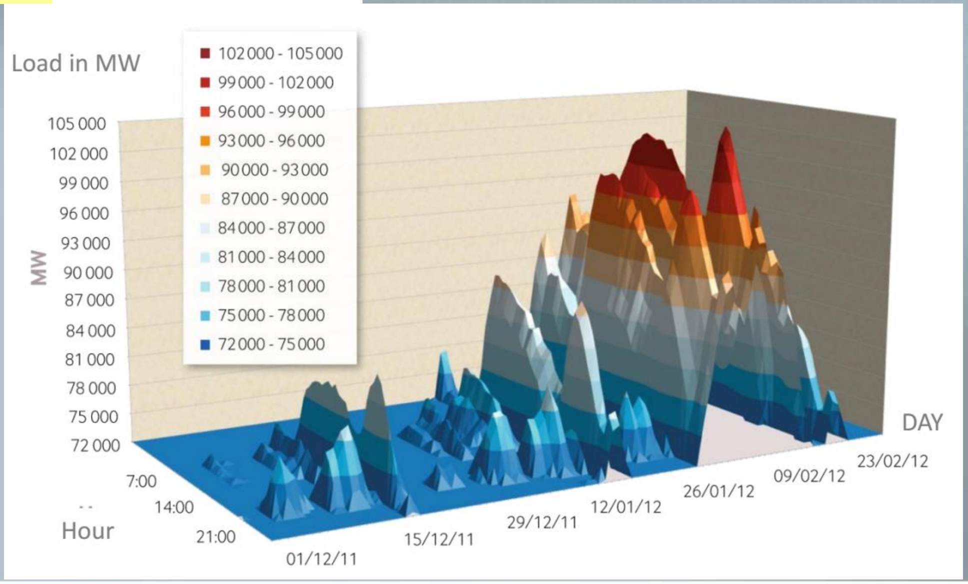
The grid: a very dynamic system



Generation from renewables Aug. 2012 Germany

Load profiles in France around Christmas 2011

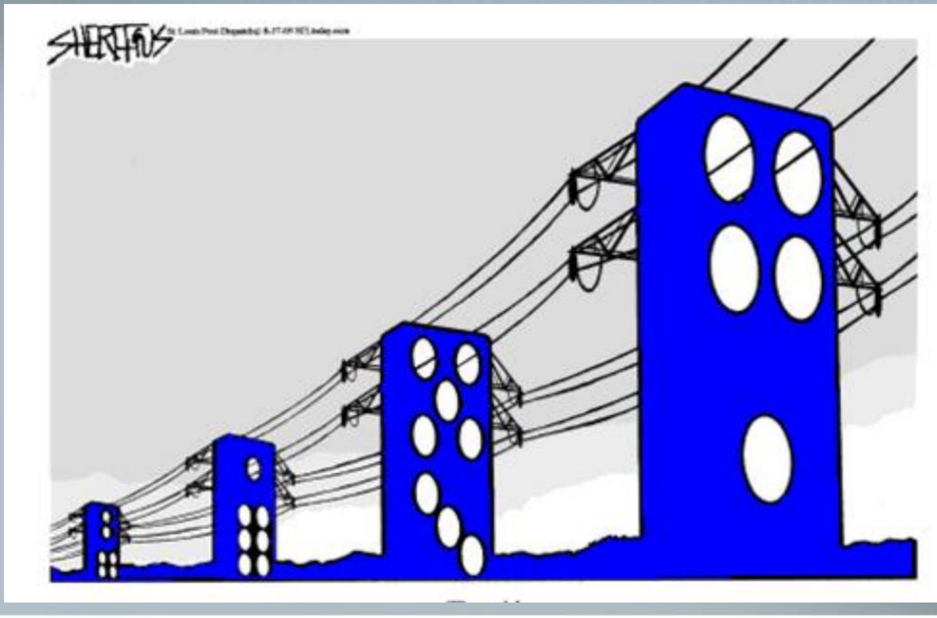
30GW variation in a few hours!



Main mechanisms producing black outs

- Initiator: rupture of production/consumption balance or loss of a transmission element, Then…
- Loss of synchronism
  - Very quick, long distance effects
- Tension collapse
  - Progressive, "local"

Characteristics of black outs are very diverse



The grid is a Hybrid Stochastic system

- System state characterized by discrete and continuous variables
  - The discrete part acts on the continuous one
    - Topology change, failures => differential equations change
  - The continuous part acts on the discrete one
    - Protection thresholds
    - Failure rates depend on temperature (canicule effect), cables length depends on Joule effect

Other examples

- Level 2 PSA of a NPP
  - Evaluation of the probability of radioactive elements dissemination
  - Requires the modeling of the interaction between continuous physical processes and discrete events (failures, operator actions…)
- Process control systems (chemical, oil…)

What is « dynamic reliability » ?

- Models and calculation methods taking into account the bi-directional interaction between
  - discrete events causing sudden state changes

    and
  - continuous physical processes

State vector of the system = $(X, I)_t$, where :

X = vector of continuous variables

I = index of discrete state

Piecewise Deterministic Markov processes: PDMP

A mathematical framework for hybrid stochastic systems

The theoretical model in dynamic reliability

Standard model (with continuous trajectories for continuous variables)

$$\frac{dX}{dt} = g(X, I)$$

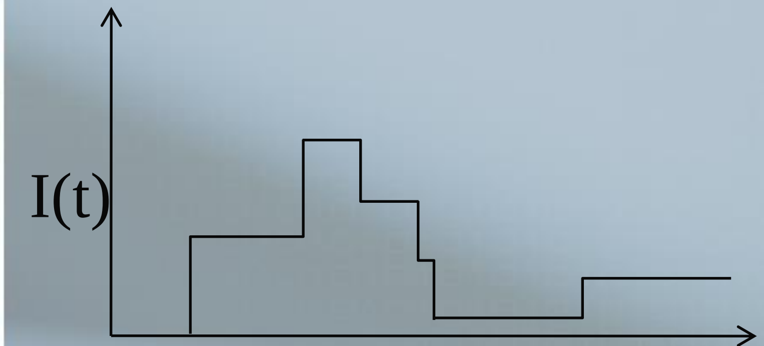
$$\Pr(I(t+\Delta t) = j / I(t) = i)) = a(i, j, X(t)) + o(\Delta t)$$

« Piecewise deterministic Markov process »
(Davis 1984)

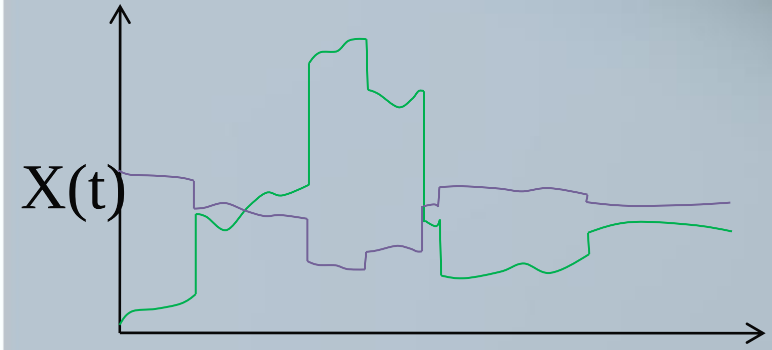
The time *t* itself is often included in *X*:
Allows to model non exponential distributions

Extended model: discontinuities are allowed for « continuous » variables when I changes

Trajectory of a PDMP



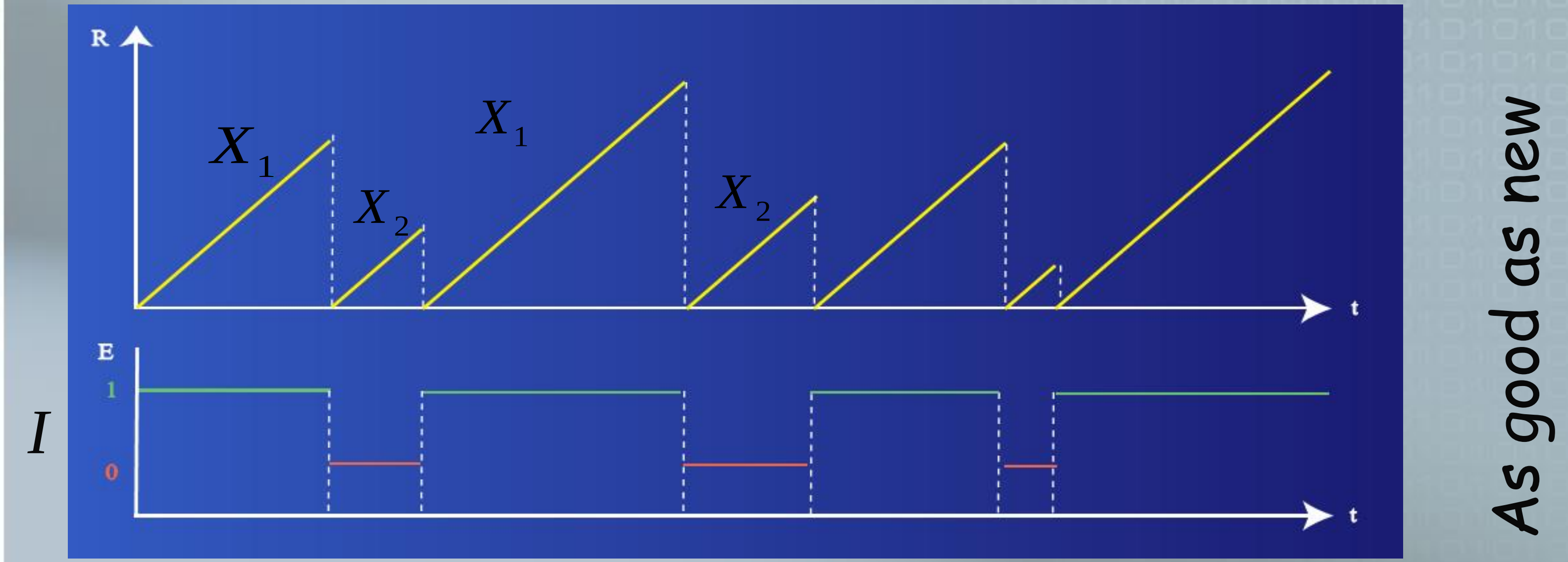
The discrete part: whatever the number of discrete variables, the system states can be indexed on N



The « continuous » part: X(t) is a vector of variables which evolve along continuous trajectories between jumps of I(t)

Single component with arbitrary failure and repair distributions

- Failure rate $\lambda(t)$ and repair rate $\mu(t)$



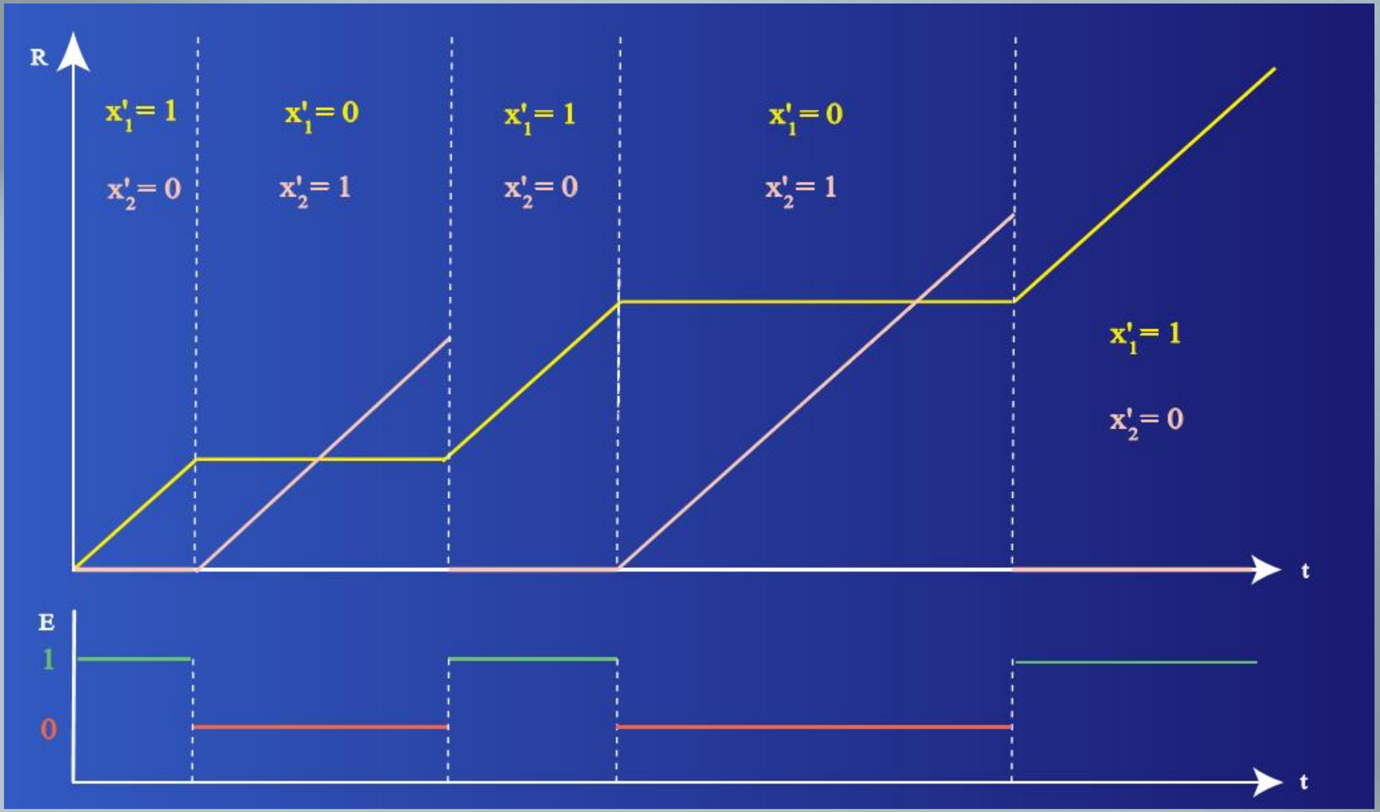
$$\frac{dX_1}{dt} = I \quad \text{and} \quad \frac{dX_2}{dt} = 1 - I$$

$$\Pr(I(t+\Delta t)=1 / I(t)=0)) = \lambda(X_1) + o(\Delta t)$$

$$\Pr(I(t+\Delta t)=0 / I(t)=1)) = \mu(X_2) + o(\Delta t)$$

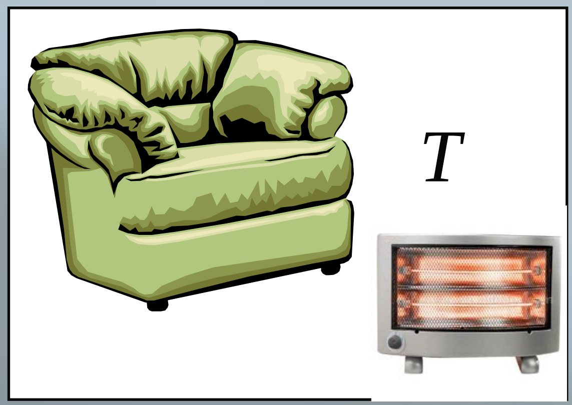
Modeling various hypotheses on maintenance effects with PDMP

- The age X_1 of the component is not reset to 0 at each failure



As bad as old

A more physical example: heated room



T_E External temperature

Heater:

- on at Tmin, off at Tmax
- subject to random failures (in operation and on demand*) and repairs
- exponential distributions for times to failure (rate lambda) and times to repair (rate mu)

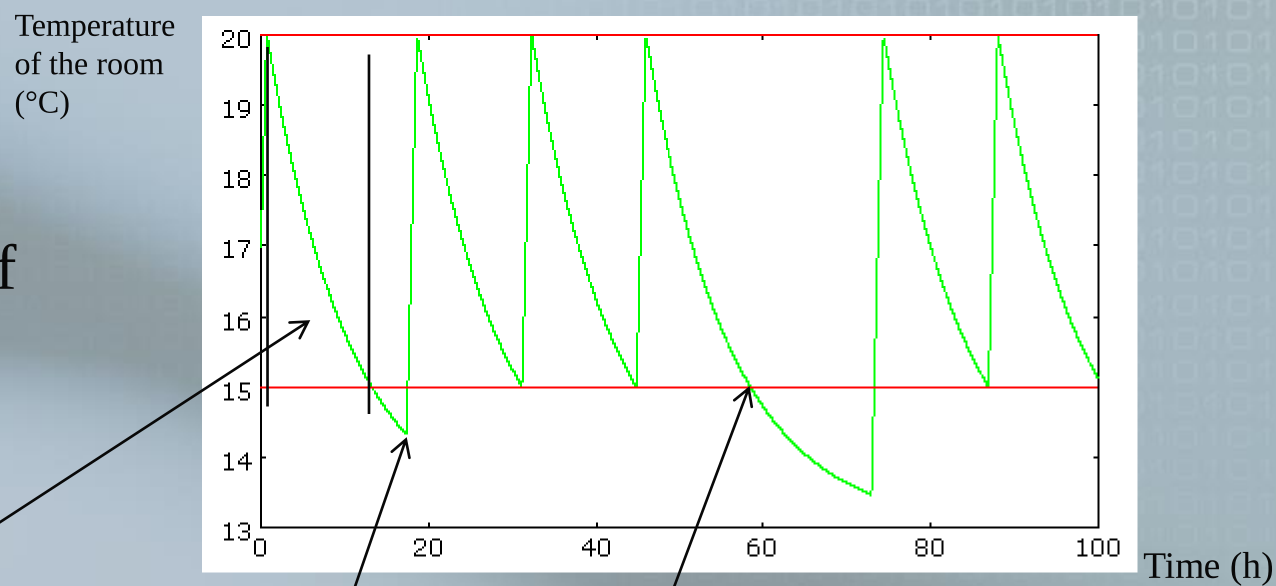
$$\frac{dT}{dt} = heater\_on(t).Power.K1 - (T(t) - T_E).K2$$

$$Ex: \frac{dT}{dt} = heater\_on(t) \times 5 - (T(t) - 13) \times 0.1$$

(time in hours, temperatures in Celsius degrees)

*This is a variant of the initial statement, to introduce the need for probabilistic instantaneous choices

An example of single (random) trajectory



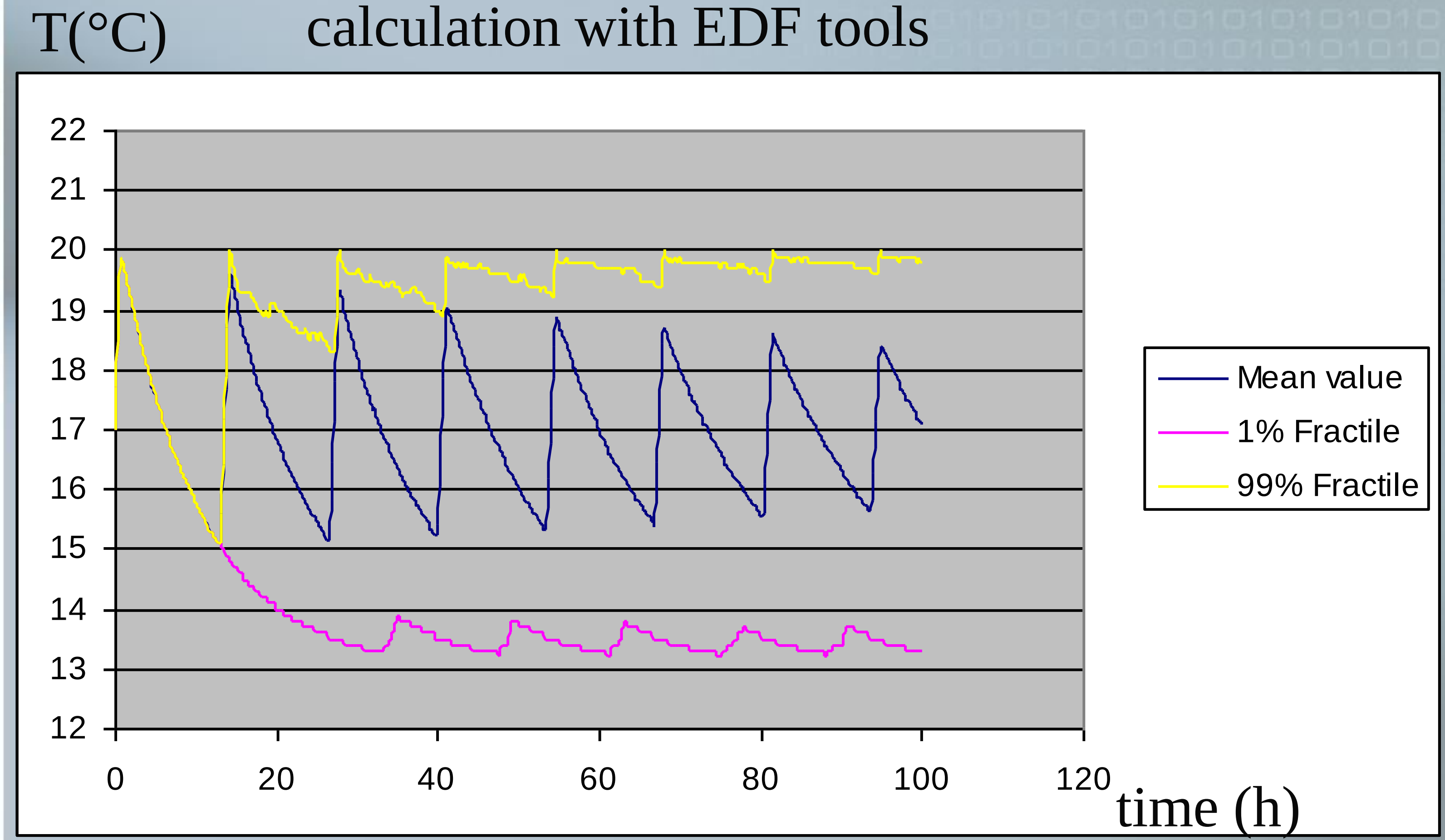
A failure in operation of the heater occurred somewhere here

And was repaired at that time

A failure on demand of the heater occurred here

Statistics on such trajectories

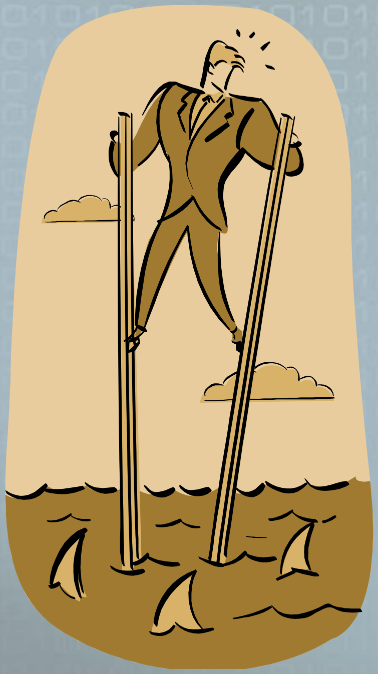
10000 random trajectories,
calculation with EDF tools



(case without failure on demand)

Dynamic reliability is a hard topic

- Mixture of probabilities, differential equations
- No convenient formalism to build models in practice
- The only possible method to solve large problems is MCS and the use of MCS is not so obvious



The two following parts are dedicated to:

- MCS strategies
- Modeling frameworks (tools)

Time handling

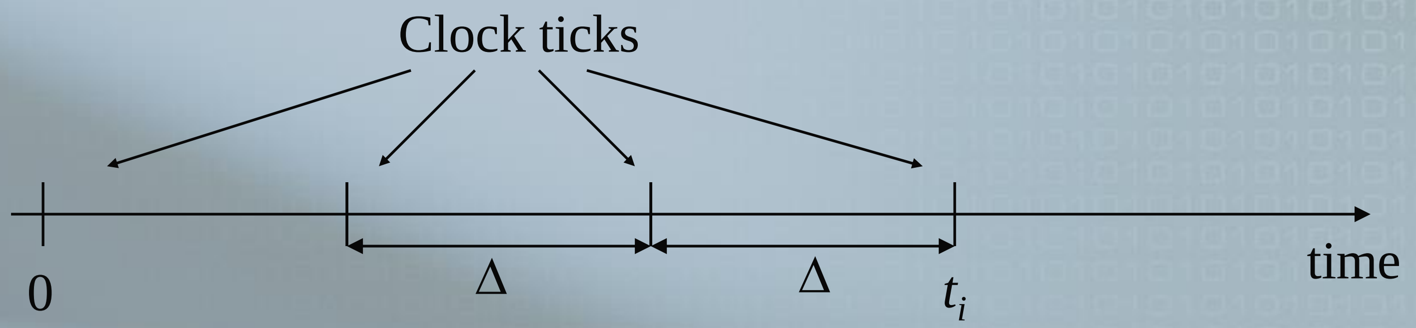
2 categories of approaches:

- **Clock-based**: update of the model at each clock tick
- **Next-event technique**: the model is only examined and updated when it is known that a state (or behavior) changes. Time moves from event to event.

First solving method : time discretization

The simplest way to manage Monte Carlo simulation

Principle of time discretization



At clock tick i, perform the following calculations :

$$X(t_i) \leftarrow X(t_{i-1}) + \Delta.g(X(t_{i-1}), I(t_{i-1}))$$ (Deterministic value)

$$I(t_i) \sim I(\Delta, I(t_{i-1}), X(t_{i-1}))$$ (Random value)

If one of the variables has hit a threshold,
Change (X, I) as needed

Advantages

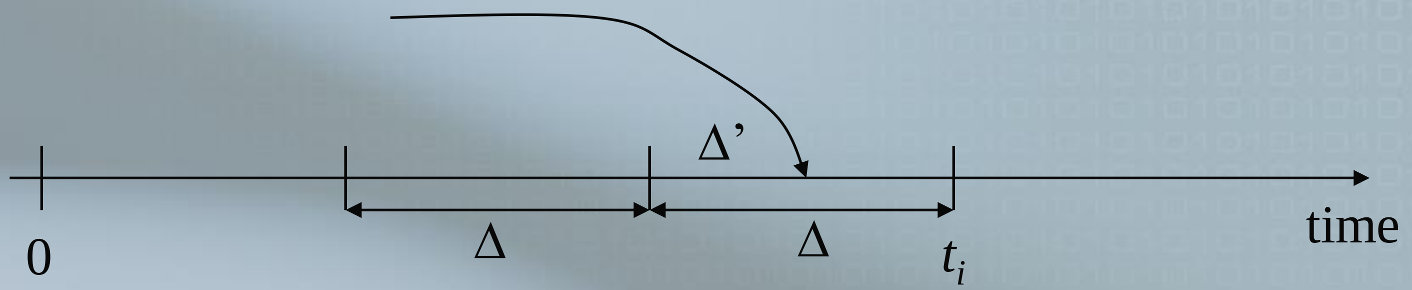
- Easy to understand: the markovian dynamic reliability model is explicitly represented
- Easy to implement

Problems…

- The probability that two random events happen in the same time interval is not zero
  - IS a problem if sequential behavior
- Cpu time: many calculations of random numbers instead of… one for an event which is not influenced by physical variables
- Non exponential distributions require:
  - An explicit function giving the hazard rate
  - Additional dimensions in X, corresponding to the starting date of random processes

Improvement

Event with an occurrence rate independent from physical variables



- Saves many random numbers calculations
- Avoids (in most cases) the problem of random events falling in the same time interval
- But requires an intermediary calculation for the state of the whole system with time step Δ'

Second solving method : state space discretization

This amounts to having a variable time step

Principle

X_d = discretized version of X

$$\frac{dX}{dt} = g(X, I)$$

$$\Delta t_i = \frac{\Delta x_d^i}{g_i(X, I)}$$

⇒ Time before next change of X_d =min(Δt_i)

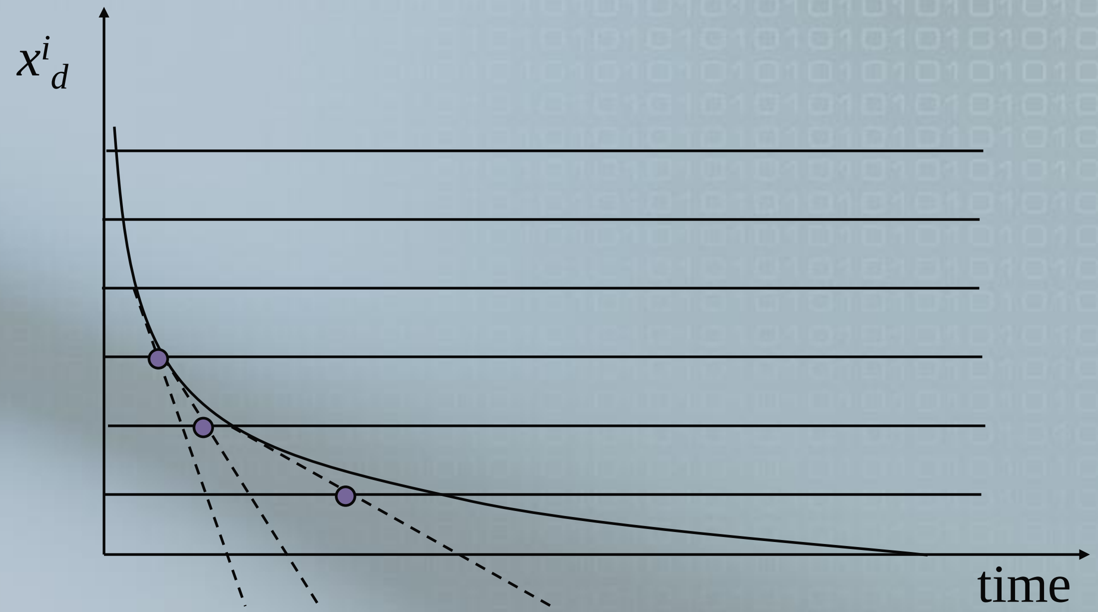
One can then perform a standard event driven simulation, each change of one of the continuous variables causing an « event » in the scheduler

If the model is a Petri net there must be two timed transitions for each variable (to increment/decrement it)

What if the deterministic variables are not linear in time ?

$$\frac{dX}{dt} = g(X, I)$$

$$\Delta t_i = \frac{\Delta x_d^i}{g_i(X, I)}$$



At each change of X_d, Δt must be re-evaluated

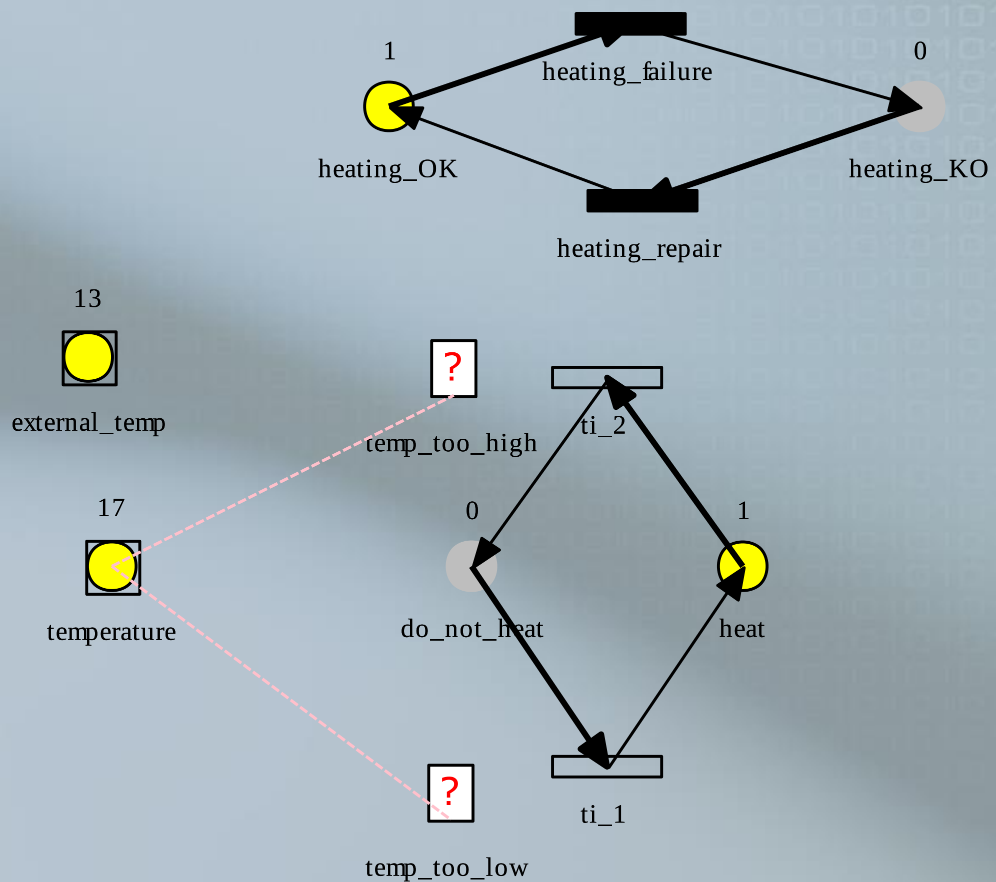
Example: exponential evolution $\frac{dx}{dt} = kx \Rightarrow \Delta t \approx \frac{\Delta x_d}{kx_d}$

(exact solution: $\Delta t = \frac{1}{k} Ln(1 + \frac{\Delta x_d}{x_d})$)

Advantages/drawbacks

- Advantages
  - Can be implemented with (nearly) standard discrete system simulation tools
  - Non exponential distributions easy to implement
  - Precision can be improved if analytical solution of differential equations known
  - Discretization can be chosen in order to put thresholds exactly « on » discrete values
- Drawbacks
  - It is impossible to model phenomena such as the increase of a failure rate with temperature (approximation not mastered)

These two methods were compared in [1]



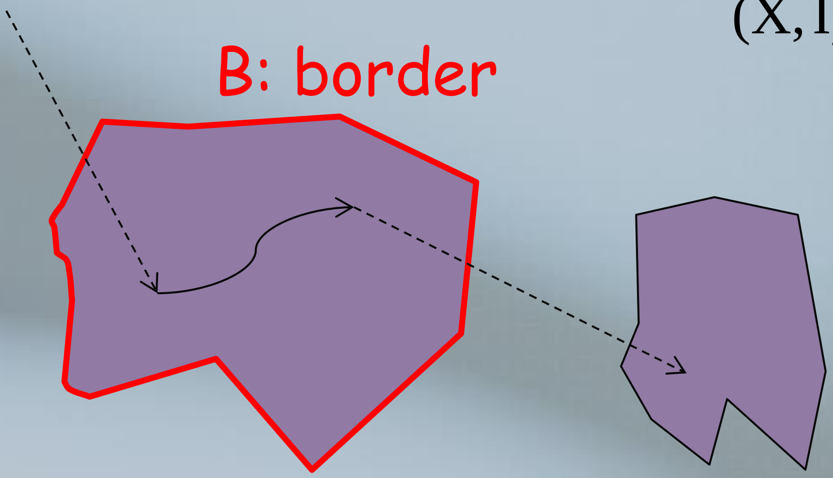
[1] M. Bouissou, Comparison of two Monte Carlo schemes for simulating PDMP, MMR 2007

Third solving method: event driven simulation

Principle

- At each state « jump » (change of I, or discontinuity of X), the process initiates a new trajectory of the deterministic part
- Competition in time between the fact that this deterministic part reaches a threshold and the random discrete events
- The dates of random discrete events must be re-evaluated using the evolution of their occurrence rates under the hypothesis that the differential equations are unchanged

Time S of the next jump

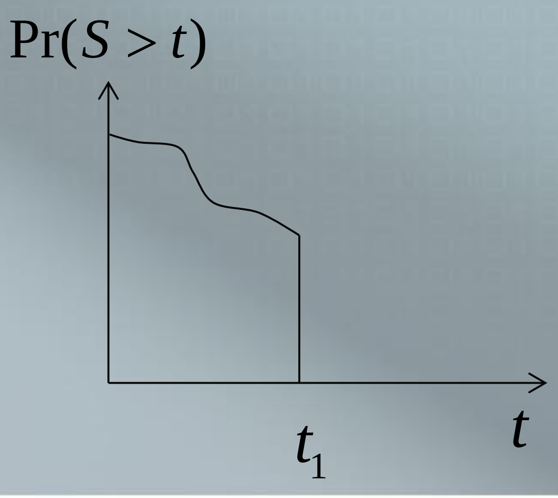


$(X, I)_t =$ deterministic trajectory

$$t_1 = \inf\{t > 0 \mid (X, I)_t \in B\}$$

$S =$ time of next jump

$$\Pr(S > t) = \begin{cases} \exp\left(-\int_0^t \lambda((X, I)_u)\,du\right) \text{ if } t < t_1 \\ 0 \text{ if } t \geq t_1 \end{cases}$$



Advantages/drawbacks

- Advantages
  - No approximation, no choice of discretization step (except for the solution of diff. eq.)
  - Non exponential distributions easy to implement
  - Minimizes cpu time
- Drawbacks
  - Hard to implement => usually requires ad-hoc programs => how not to be suspicious about the correctness of such programs?

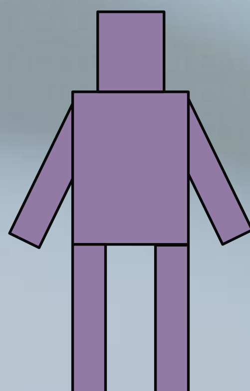
Modeling tools

Looking for a user friendly tool that would implement the last MCS strategy

How about *user-friendly* tools?

Deterministic land

Probabilistic land



$$\frac{dy}{dx} = \ldots$$

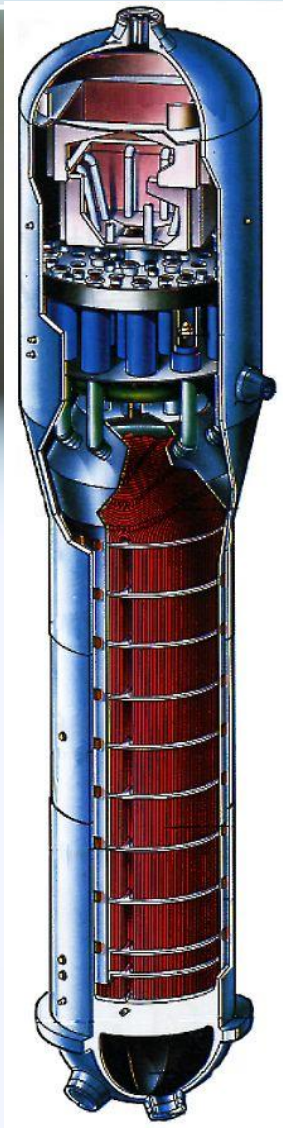
$$Pr\ (X) = \ldots$$

Benchmarks

- Benchmark conducted by EDF on a use case of middle complexity: the control system of the input flow of a steam generator in a NPP (APPRODYN project)

- 2 ESREL papers: Critical comparison of two user-friendly tools to study PDMP
  - 2012: comparison of Vensim (det.) and KB3 (prob.)
  - 2013: comparison of Modelica (det.) and PyCATSHOO (prob.)

Steam generator control



- 40 pages use case description
- Nominal behavior, equations of the water level controller
- Transients due to startup and shutdown of the plant
- State graphs of components, failure modes, failure and repair rates
- Undesirable event: the level becomes too high or too low

Methods tried on this case

- Hybrid stochastic automata, implementation via Scilab/Scicos
  - Only a simplified model could be built
  - Combinatorial explosion
- PDMP: Simulink and Stateflow
  - Worked quite well, compositional approach
  - The most readable models of the benchmark
  - Too slow calculations

Methods tried on this case

- Stochastic Petri nets: MOCA-RP
  - 228 places, 281 transitions, 664 arcs and 81 variables, organized in 45 "modules"
  - The continuous equations of the controller had to be replaced by discrete approximations

And always that suspicion: are the models valid?

Conclusion: nothing really worked!

The ESREL papers

- Based on the "heated room" test case
- The second paper makes a global synthesis about the 4 tested tools
- Three of the tools are based on an object oriented modeling language
- Vensim, a tool created for "system dynamics" is not flexible enough to allow creating reusable models

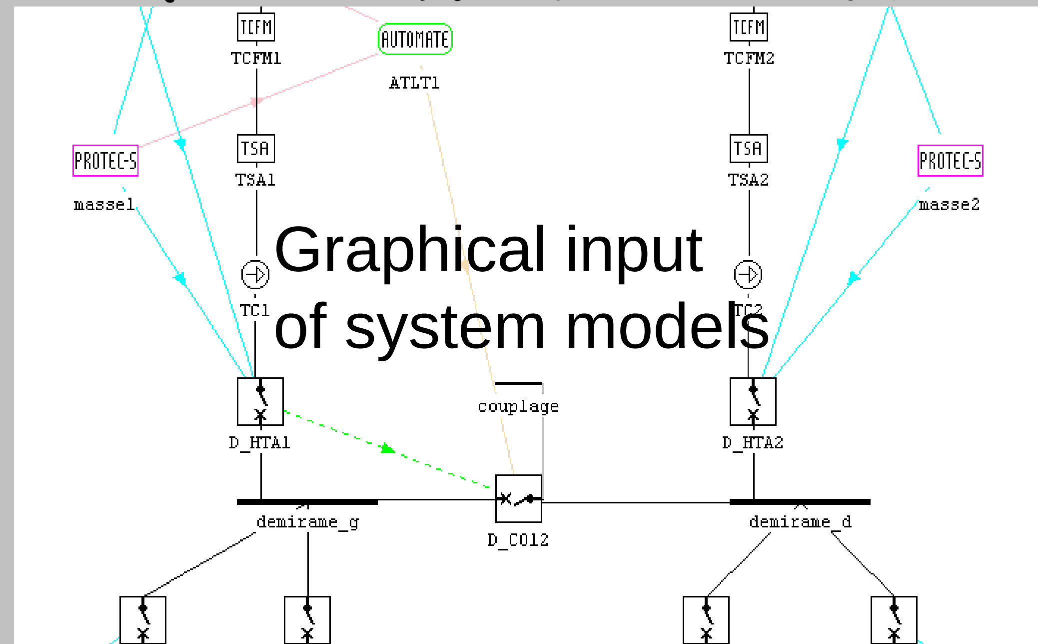
Test case resolution with the KB3 workbench

How to solve the problem in 1 hour

Principles of the KB3 workbench

Library (in Figaro language)

Generic description of components



Graphical input of system models

Figaro 0 (**textual** model)

Fault tree

Standard fault tree processors:

Risk-Spectrum, …

- ✓Minimal cut sets
- ✓Reliability, Availability

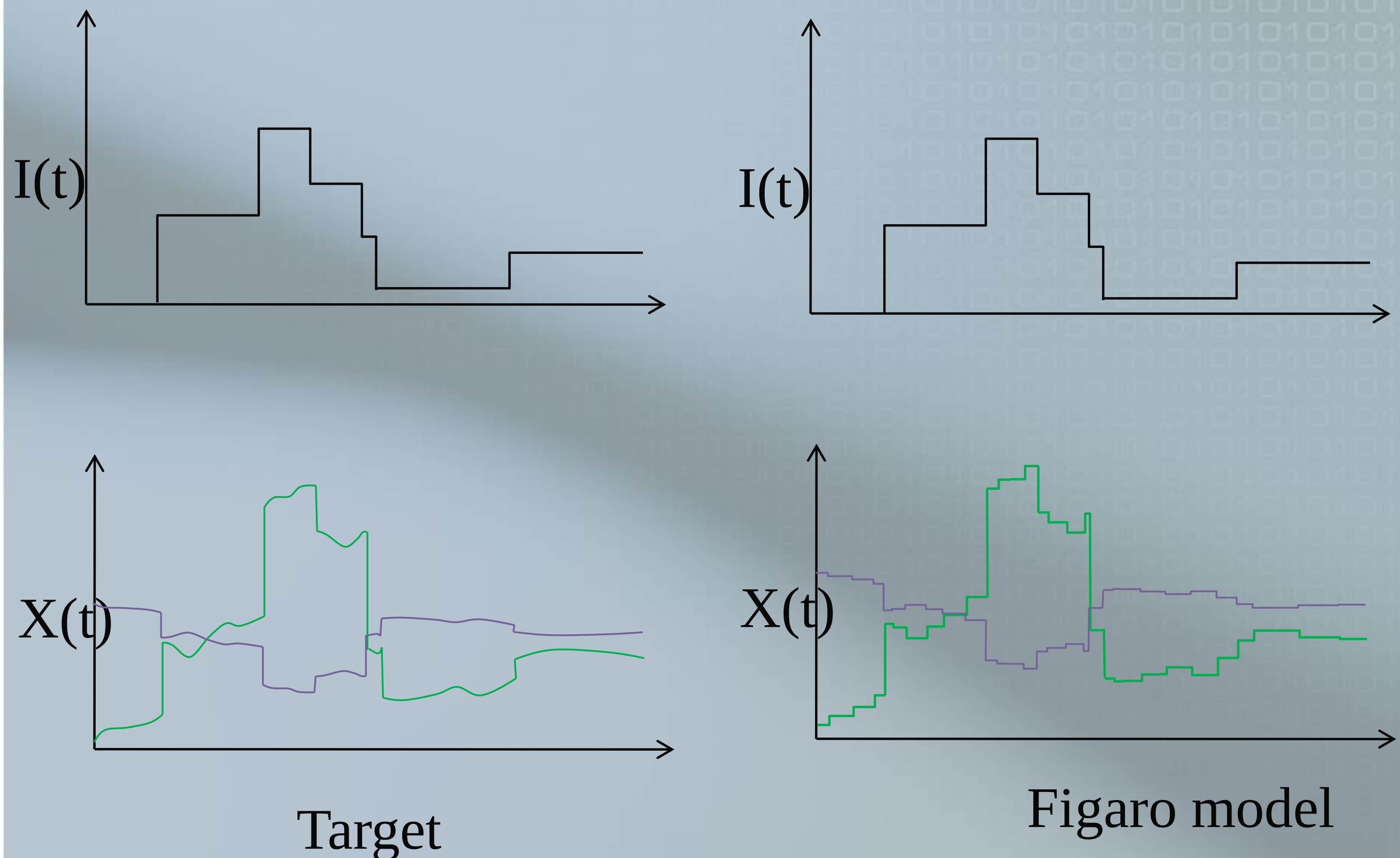
Sequences Generator: FIGSEQ

- ✓Most probable sequences
- ✓Reliability, MTTF
- ✓Asymptotic availability

Monte-Carlo simulator: YAMS

- ✓Most probable sequences
- ✓Reliability, availability
- ✓Mean values of numeric variables…

The kind of processes that can be described in Figaro



Limitation: a small time step must be used

A simple KB for dynamic reliability: « hybrid » Petri nets

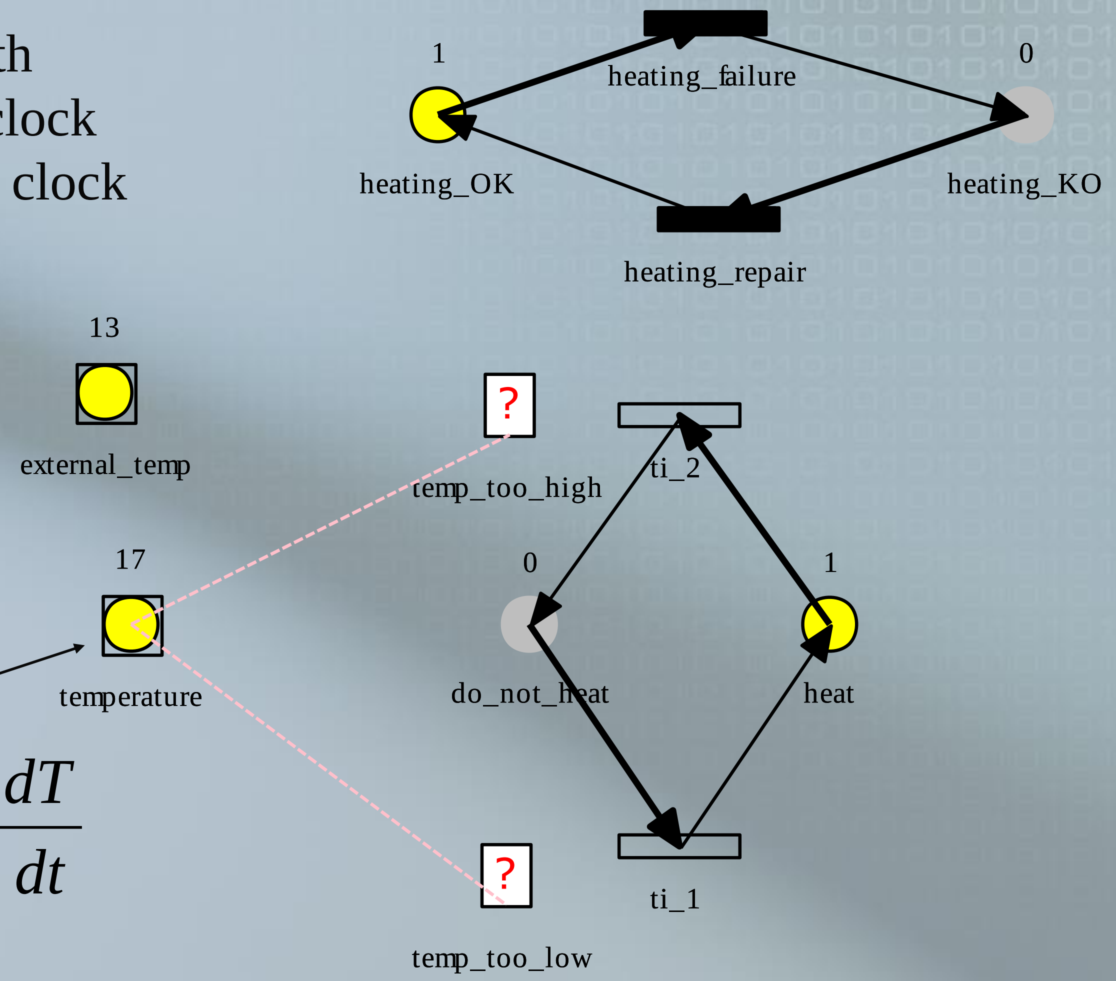
Includes:

- standard Petri nets
- Boolean messages
- Boolean functions on messages
- Randomly distributed parameters
- Continuous variables
- Special behavior of timed transitions

KB size (lines of FIGARO language):
- Petri nets: 215 lines
- Hybrid Petri nets: 405 lines

Heated room: model with the « Hybrid Petri net » KB

Any model built with this KB includes a clock (here: time between clock ticks = 1mn)



The expression of $\frac{dT}{dt}$ is input here

New approaches developed at EDF

- The PyCATSHOO tool
- Modelica extensions

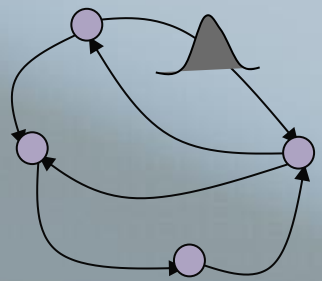
PyCATSHOO: motivation and ambitions

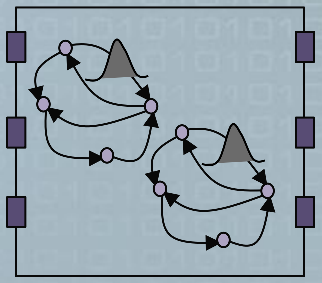
- **Motivation:** A new EDF R&D tool aimed at overcoming the limitations of KB3 in a hybrid context
  - In terms of required functionalities dealing with stochastic hybrid systems
  - In terms of openness

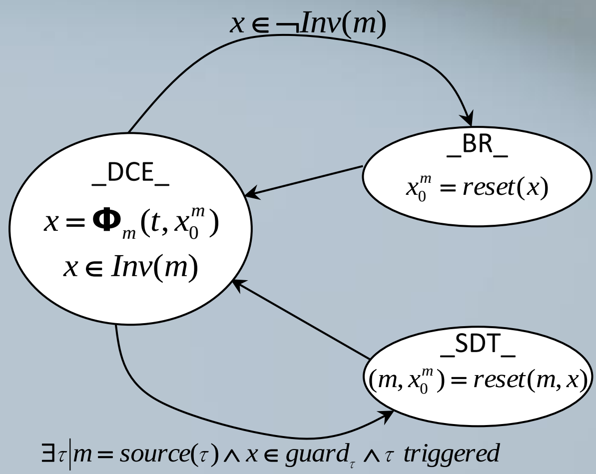
Ambitions

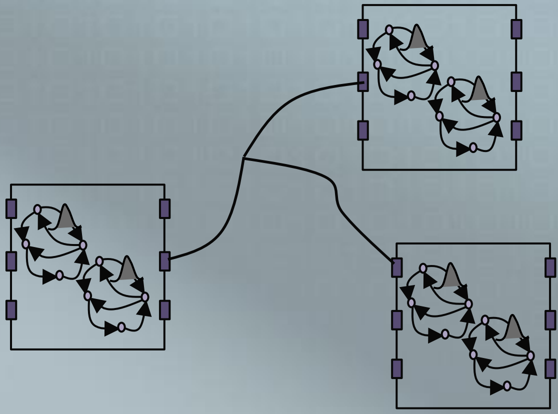
- To provide the basic components required to model pure discrete stochastic behavior: States, Transitions, probability distributions, etc.
- To provide user-friendly means to model PDMP
- To give access to a wide range of scientific computation tools

PyCATSHOO: principles









Automatically implements the 3rd MCS scheme

Resolution with Modelica

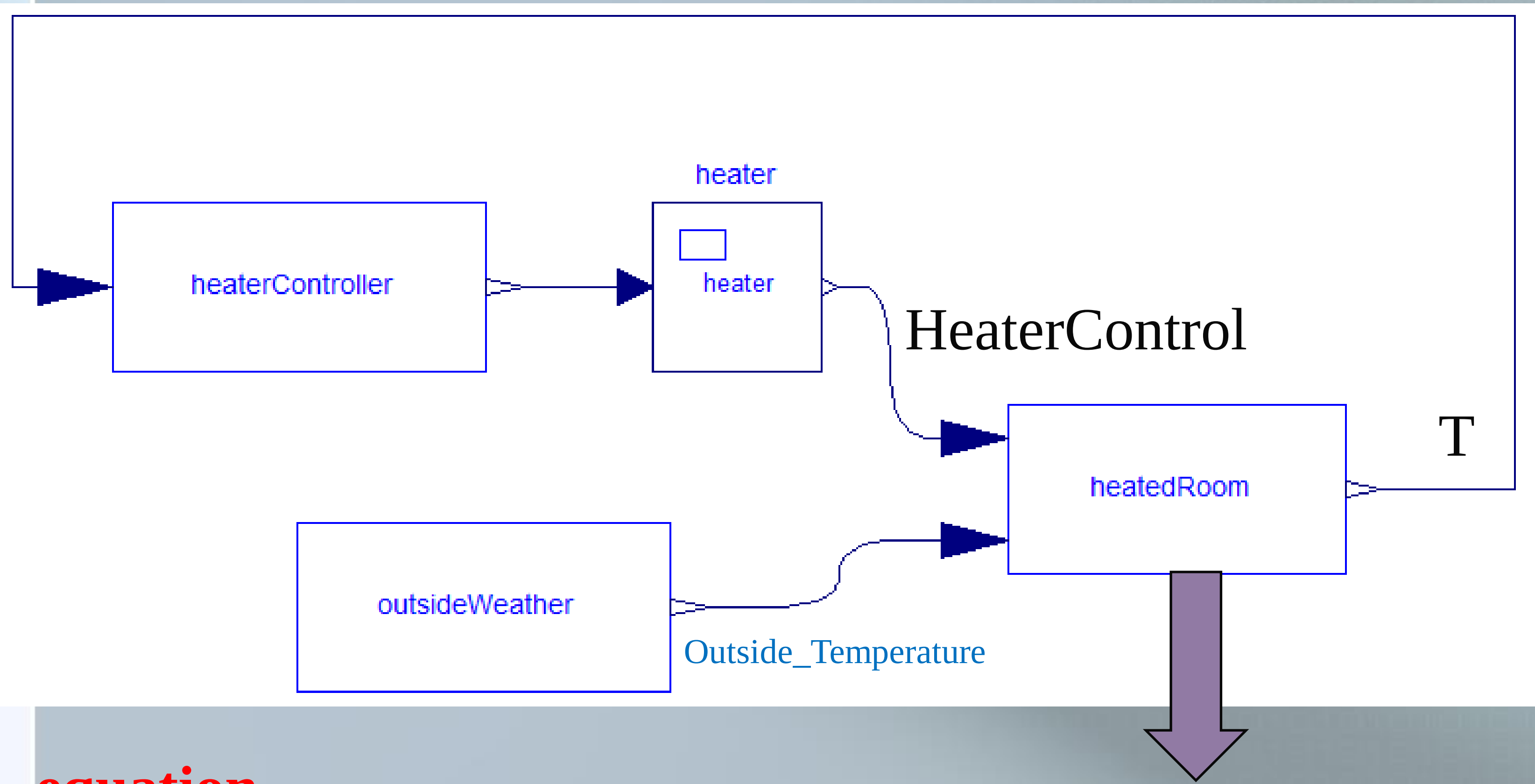
Modelica was originally designed for building/solving deterministic models

Main features of Modelica tools*

- Integrated tools:
  - GUI for model building
  - Integrated solver
  - Graphical outputs
  - Sensitivity analysis features ?
- Variable time step
- No room for aleatory concepts

*Dymola, Simulation X, Open Modelica…
See www.modelica.org

Modelica model



equation

der(T)= 5*HeaterControl + 0.1*(Outside_Temperature - T) ;

The heater

```
algorithm
when initial() then
  F := seed;
//each calculation of F will yield a pseudo random number in [0,1]
end when;
// Attention: the two following rules must not be merged in a single one!
when initial() then   //calculating the first random working time
    F := mod(a*F+c, m);
    x := F / m;
    X:= (-log(1-x))/lambda;
end when;
when  working then //random draw of the next working time
    F := mod(a*F+c, m);
    x := F / m;
    X:= (-log(1-x))/lambda;
end when;
// X is the working time
when working and (time - starttime_working) > X then
    working := false;
    starttime_notworking := time;
end when;
```

…. Similar instructions for repairs

```
// Input-output relation equation
if working then
  y =  u;
 else
   y =  0.;
end if;
```

Conclusions on this experiment

- In principle Modelica is able to solve the problem **but**
- Model building is difficult, error prone
- Models are
  - Hardly readable by humans
  - Unreadable by machines (except for simulation)

Modelica

```
when  working then
    F := mod(a*F+c, m);
    x := F / m;
    X:= (-log(1-x))/lbda;
end when;
when working and (time -
 starttime_working) > X then
    working := false;
    starttime_notworking := time;
end when;
```

FIGARO

```
IF working
MAY_HAPPEN failure
   INDUCING
       working ←FALSE
   DIST EXP(lbda);
```

PyCATSHOO

```
self.addTransition  ("failure",
"working"  , "not_working",
law = HCExpoPLaw
(rate=lbda))
```

Modelica for PDMP

- Already existing features (from 3.3)
  - State machines with hierarchy of states (concepts of D. Harel's Statecharts)
- Missing concepts
  - Probabilistic concepts*, i.e. :
    - For immediate transitions: branching probabilities (e.g. a component required to start may or may not start)
    - For delayed transitions: probability distribution of the delay (e.g. the time to failure of a component, exponentially distributed)

The change to be made is similar to the change from Petri nets to Stochastic Petri nets

* Already available in FIGARO, AltaRica, PyCATSHOO

The MODRIO project

- Launched in Sept. 2012 by EDF
- Aims: extend the use of Modelica models from pure design to
  - Proof of properties
  - Exploitation of systems
  - Dependability analysis
    - Hybrid stochastic systems
    - Fault-tree and Bayesian network generation

Conclusion

- More and more needs for hybrid stochastic systems simulation
- No user friendly tool available yet

But

- Extensions of Modelica tools
- PyCATSHOO

Are both promising

References

- Davis M.H.A. Markov Models and Optimization. Chapman& Hall, 1993.
- Bouissou M. Comparison of two Monte Carlo schemes for simulating Piecewise Deterministic Markov Processes. Proc. of MMR 2007
- Zhang H., Dufour F., Dutuit Y., Gonzalez K. Piecewise Deterministic Markov Processes and dynamic reliability. Proceedings of the institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, 222(4), pp. 545-551, 2008.
- Chraibi H. Dynamic reliability modeling and assessment with PyCATSHOO: Application to a test case. PSAM 2013, Tokyo.
- Bouissou M. Automated Dependability Analysis of Complex Systems with the KB3 Workbench: the Experience of EDF R&D. Proc. of The International Conference on Energy and Environment, CIEM 2005, Bucharest (Romania), 2005.
- KB3 tool. http://rdsoft.edf.fr (French and English versions can be downloaded).
- Numerical integration methods for solving PDMP: papers by C. Cocozza-Thivent, R. Eymard, S. Mercier, W. Lair