StocHy: automated verification of stochastic processes

Nathalie Cauchi, Alessandro Abate

TACAS 2019

University of Oxford, Oxford, UK

Introduction

uk.reuters.com

Exclusive: Data shows angle of attack similar in Boeing 737 crashes...

engadget.com

Report: Boeing's crucial 737 Max safety analysis was flawed

Sensor Cited as Potential Factor in Boeing Crashes Draws Scrutiny
The Washington Post
Todd C. Frankel
March 17, 2019

Introduction

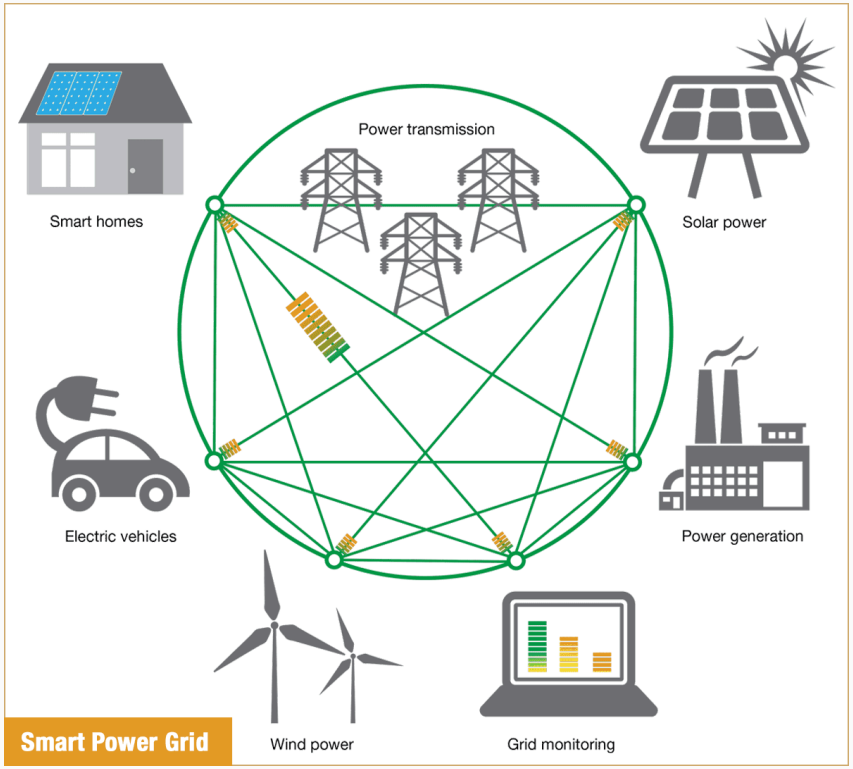
uk.reuters.com

Exclusive: Data shows angle of attack similar in Boeing 737 crashes...

engadget.com

Report: Boeing's crucial 737 Max safety analysis was flawed

Sensor Cited as Potential Factor in Boeing Crashes Draws Scrutiny
The Washington Post
Todd C. Frankel
March 17, 2019



Introduction

System dynamics:

- complex continuous dynamics
- discrete modes
- uncertainty

Specifications:

- can be formally defined

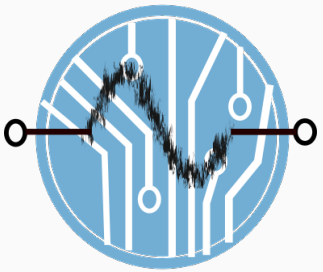
stochastic hybrid systems + **formal methods**

Introduction

Problems:

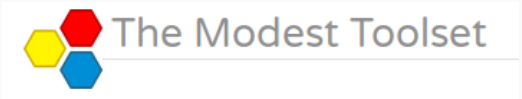
- complexity with modelling **Stoc**hastic **Hy**brid Systems
- undecidability of verification or synthesis problems
- curse of dimensionality

A new software tool

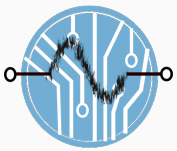


StocHy

Introduction: Related work

Tool	continuous variables	discrete variables	time	scope
	deterministic	multiple	continuous	modelling & verification & JANI support
	-	multiple	continuous or discrete	probabilistic model checking
	-	multiple	continuous or discrete	probabilistic model checking
FAUST^2	stochastic	single	discrete	probabilistic reachability analysis

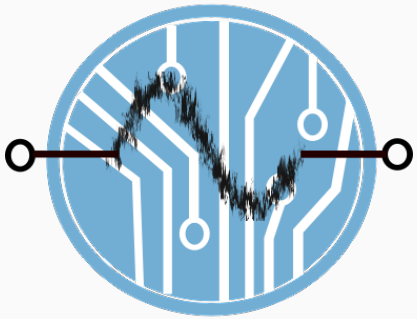
Introduction: Related work

Tool	continuous variables	discrete variables	time	scope
The Modest Toolset	deterministic	multiple	continuous	modelling & verification & JANI support
Storm	-	multiple	continuous or discrete	probabilistic model checking
	-	multiple	continuous or discrete	probabilistic model checking
FAUST^2	stochastic	single	discrete	probabilistic reachability analysis
 StocHy	stochastic	multiple	discrete	probabilistic reachability, simulation

Introduction: Contribution

verification

- abstraction based
- novel algorithm with tighter bounds and more scalability



StocHy

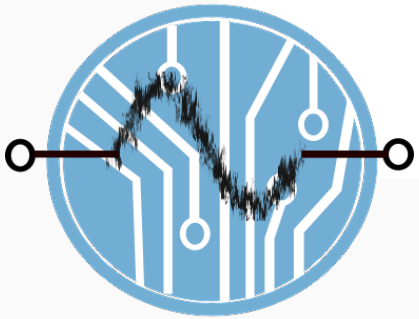
Introduction: Contribution

verification

- abstraction based
- novel algorithm with tighter bounds and more scalability

synthesis

- abstraction based
- optimisation via sparse matrices



StocHy

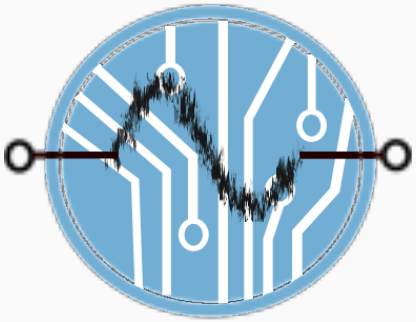
Introduction: Contribution

verification

- abstraction based
- novel algorithm with tighter bounds and more scalability

simulation

- automatically generates statistics
- visualisation via time varying histograms



StocHy

synthesis

- abstraction based
- optimisation via sparse matrices

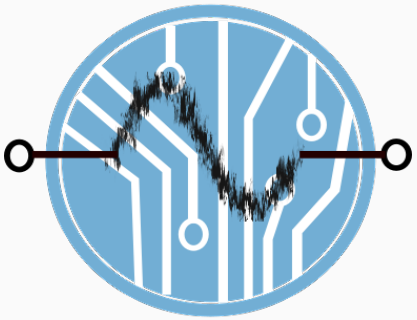
Introduction: Contribution

verification

- abstraction based
- novel algorithm with tighter bounds and more scalability

simulation

- automatically generates statistics
- visualisation via time varying histograms



StocHy

synthesis

- abstraction based
- optimisation via sparse matrices

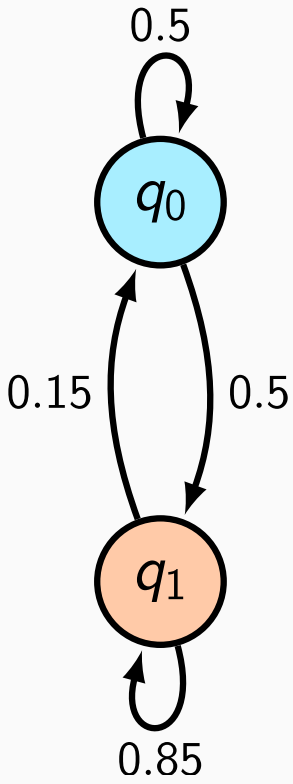
features

- modular
- C++ implementation
- extendable
- multiple options

An example of a simple SHS :

2 modes $\mathcal{Q} = \{q_0, q_1\}$

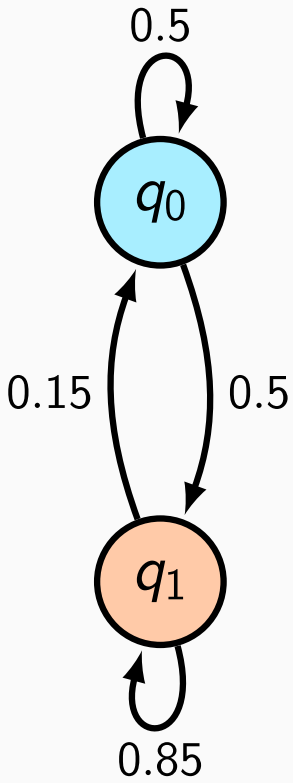
$$T_q = \begin{bmatrix} 0.5 & 0.5 \\ 0.15 & 0.85 \end{bmatrix}$$



An example of a simple SHS :

2 continuous variables $n = 2$
0 control actions $\mathcal{U} = \emptyset$

$$x_{k+1} = F_{q_i}(x_k) + \Sigma_{q_i} w_k$$
$$w_k \sim \mathcal{N}(0, 1)$$
$$T_x = \mathcal{N}(F_{q_i}(x_k), \Sigma_{q_i})$$



$$x_{1,k+1} = -0.5x_{1,k} + 0.1w_{1,k}$$
$$x_{2,k+1} = \cos(x_{1,k} + 0.8x_{2,k}) + w_{2,k}$$

$$x_{1,k+1} = 0.7x_{1,k} + 0.2x_{2,k} + w_{1,k}$$
$$x_{2,k+1} = 0.1x_{1,k}x_{2,k} + 2w_{2,k}$$

Theory: Models, Abstractions, Simulations

a discrete-time stochastic hybrid system (SHS) is

$$\mathcal{H} = (\mathcal{Q}, n, \mathcal{U}, T_x, T_q), \quad \textit{where}$$

- $\mathcal{Q} = \{q_1, q_2, \ldots, q_m\}$, $m \in \mathbb{N}$ is a finite set of modes
- $n \in \mathbb{N}$ is the dimension of the continuous space $\mathbb{R}^n$ of each q the hybrid state space is then $\mathcal{D} = \cup_{q \in \mathcal{Q}} \{q\} \times \mathbb{R}^n$
- $\mathcal{U}$ is a continuous set of actions, e.g. $\mathbb{R}^v$
- T_q is the discrete transition probability kernel
- T_x is the continuous transition probability kernel

Theory: Models, Abstractions, Simulations

a discrete-time stochastic hybrid system (SHS) is

$$\mathcal{H} = (\mathcal{Q}, n, \mathcal{U}, T_x, T_q), \quad \textit{where}$$

- $\mathcal{Q} = \{q_1, q_2, \ldots, q_m\}$, $m \in \mathbb{N}$ is a finite set of modes
- $n \in \mathbb{N}$ is the dimension of the continuous space $\mathbb{R}^n$ of each q the hybrid state space is then $\mathcal{D} = \cup_{q \in \mathcal{Q}} \{q\} \times \mathbb{R}^n$
- $\mathcal{U}$ is a continuous set of actions, e.g. $\mathbb{R}^v$
- T_q is the discrete transition probability kernel
- T_x is the continuous transition probability kernel

How do we analyse and reason over SHS?

Theory: Models, Abstractions, Simulations

- SHS evolve over uncountable space
  - need to perform abstractions
- quantative and finite abstractions into Markov processes
- characterised by a state space, transition probabilities and (possibly) actions
- two different abstract models:
  - Markov Decision processes (MDP)
  - Interval Markov Decision process (IMDP)
- handles complex reachability properties
- can handle time bounded and unbounded properties

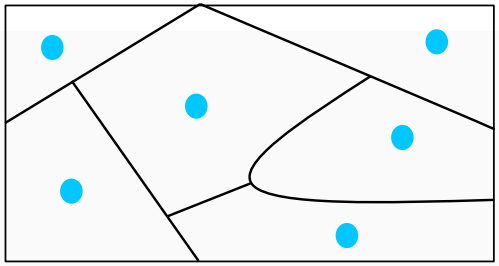
Theory: Models, Abstractions, Simulations

abstraction into MDP

- define desired abstraction error
- grid state-space (uniform or adaptive)
- compute transition probabilities via marginalisation
- hinges on computation of Lipschitz constants (h_s) of transition kernel
- N-step error

$$\varepsilon = h_s \delta \mathcal{L}(A) N$$

δ max diameter of partition, $\mathcal{L}(A)$ volume of set



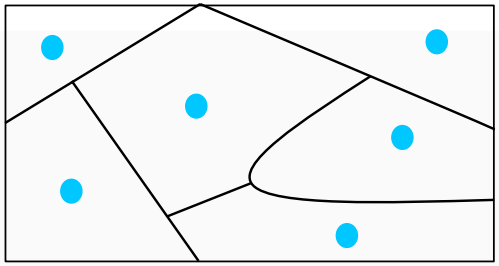
Theory: Models, Abstractions, Simulations

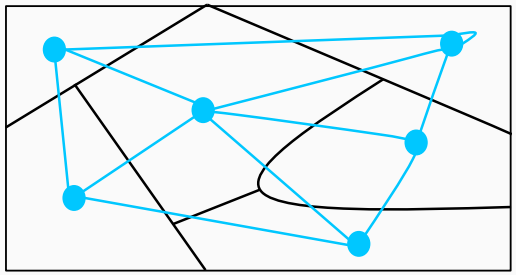
abstraction into MDP

- define desired abstraction error
- grid state-space (uniform or adaptive)
- compute transition probabilities via marginalisation
- hinges on computation of Lipschitz constants (h_s) of transition kernel
- N-step error

$$\varepsilon = h_s \delta \mathcal{L}(A) N$$

δ max diameter of partition, $\mathcal{L}(A)$ volume of set





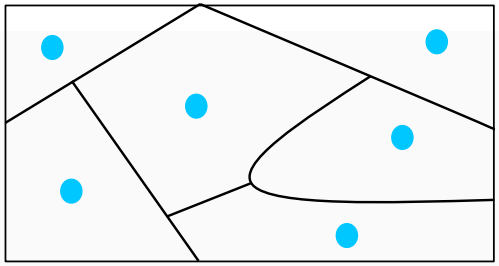
Theory: Models, Abstractions, Simulations

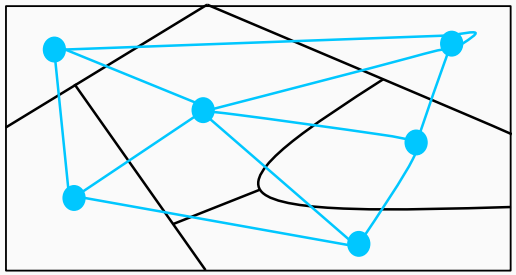
abstraction into MDP

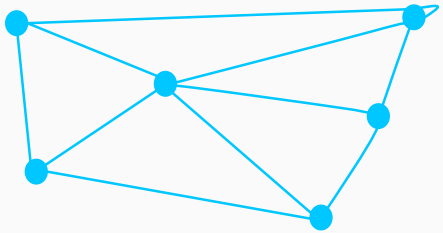
- define desired abstraction error
- grid state-space (uniform or adaptive)
- compute transition probabilities via marginalisation
- hinges on computation of Lipschitz constants (h_s) of transition kernel
- N-step error

$$\varepsilon = h_s \delta \mathcal{L}(A) N$$

δ max diameter of partition, $\mathcal{L}(A)$ volume of set







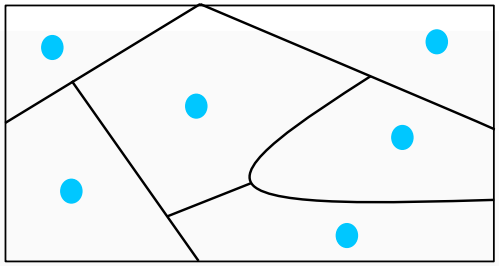
Theory: Models, Abstractions, Simulations

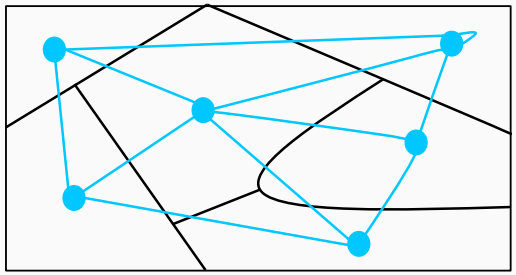
abstraction into MDP

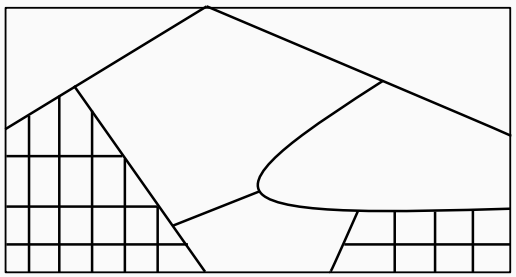
- define desired abstraction error
- grid state-space (uniform or adaptive)
- compute transition probabilities via marginalisation
- hinges on computation of Lipschitz constants (h_s) of transition kernel
- N-step error

$$\varepsilon = h_s \delta \mathcal{L}(A) N$$

δ max diameter of partition, $\mathcal{L}(A)$ volume of set





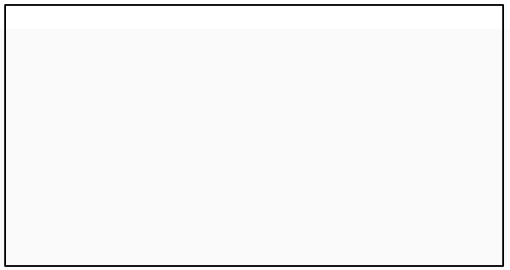


Theory: Models, Abstractions, Simulations

abstraction into IMDP

- discretise state space
- quantify abstraction error
- embed error as uncertainty
- compute exact min $\check{p}(q)$ and max $\hat{p}(q)$ transition probabilities for each state q
- tighter error bounds

$$\varepsilon = \hat{p}(q) - \check{p}(q)$$

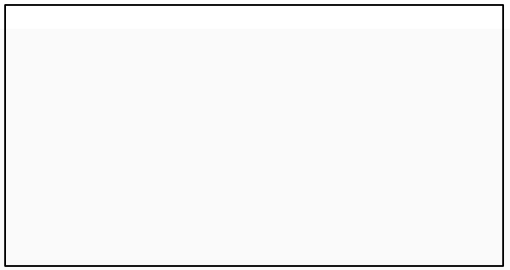


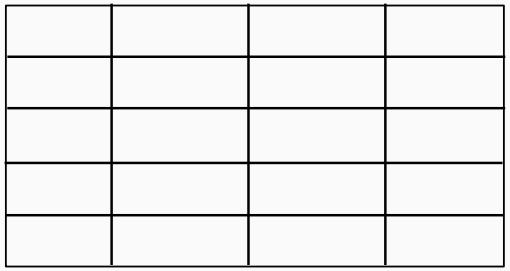
Theory: Models, Abstractions, Simulations

abstraction into IMDP

- discretise state space
- quantify abstraction error
- embed error as uncertainty
- compute exact min $\check{p}(q)$ and max $\hat{p}(q)$ transition probabilities for each state q
- tighter error bounds

$$\varepsilon = \hat{p}(q) - \check{p}(q)$$



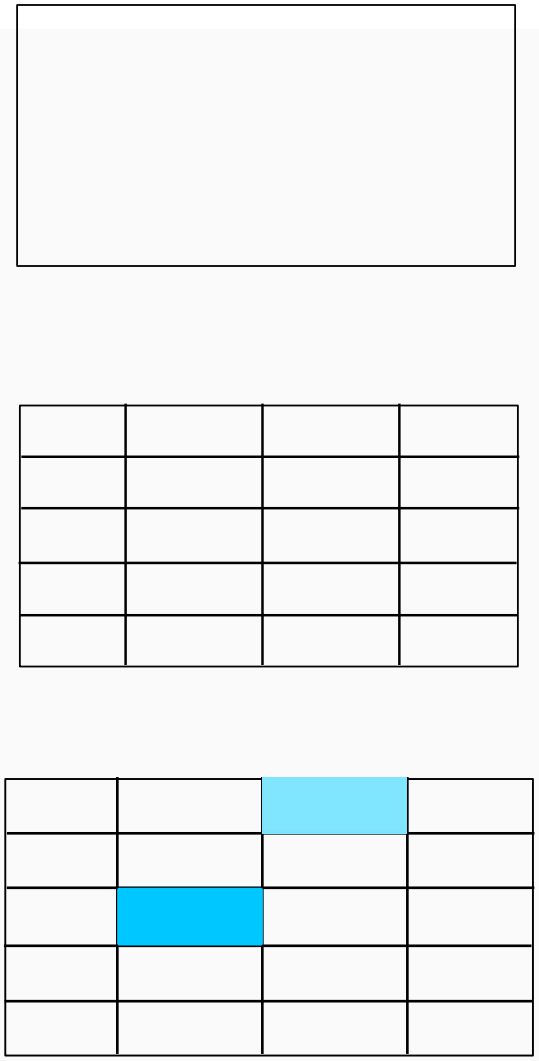


Theory: Models, Abstractions, Simulations

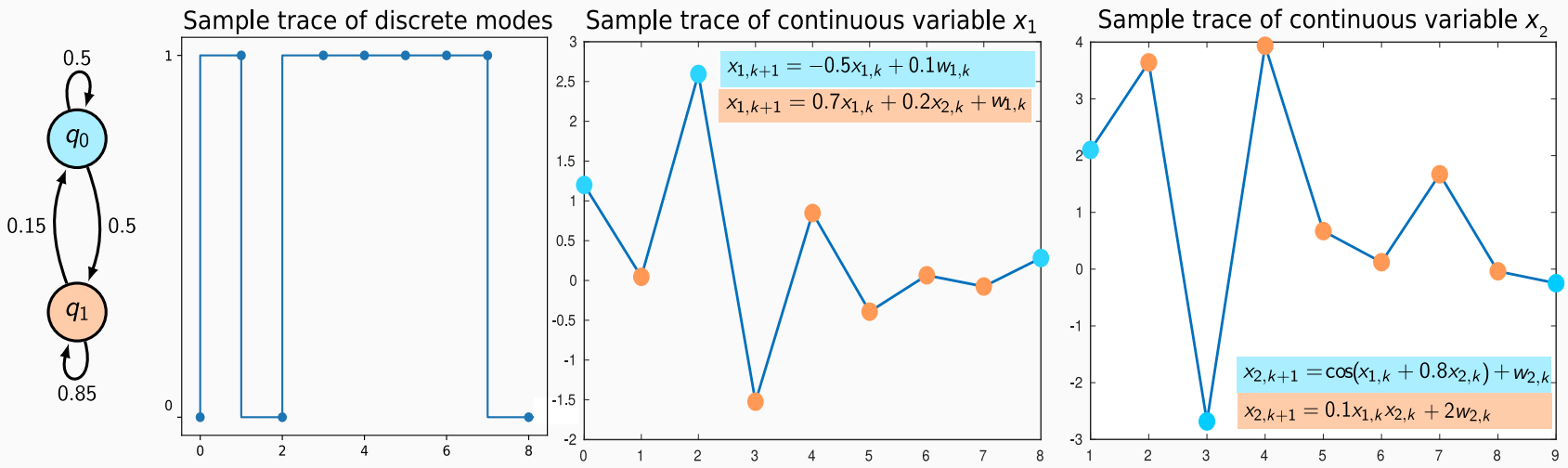
abstraction into IMDP

- discretise state space
- quantify abstraction error
- embed error as uncertainty
- compute exact min $\check{p}(q)$ and max $\hat{p}(q)$ transition probabilities for each state q
- tighter error bounds

$$\varepsilon = \hat{p}(q) - \check{p}(q)$$



simulation of SHS:



- we employ Monte Carlo techniques for simulation
- over a fixed time horizon
- construct histograms at each time step

Overview of StocHy

installation

1. obtain dependencies by running: `get_dep.sh`
2. run stochy by executing: `run.sh`

tested on machines under Linux OS

Overview of StocHy

installation

1. obtain dependencies by running: `get_dep.sh`
2. run stochy by executing: `run.sh`

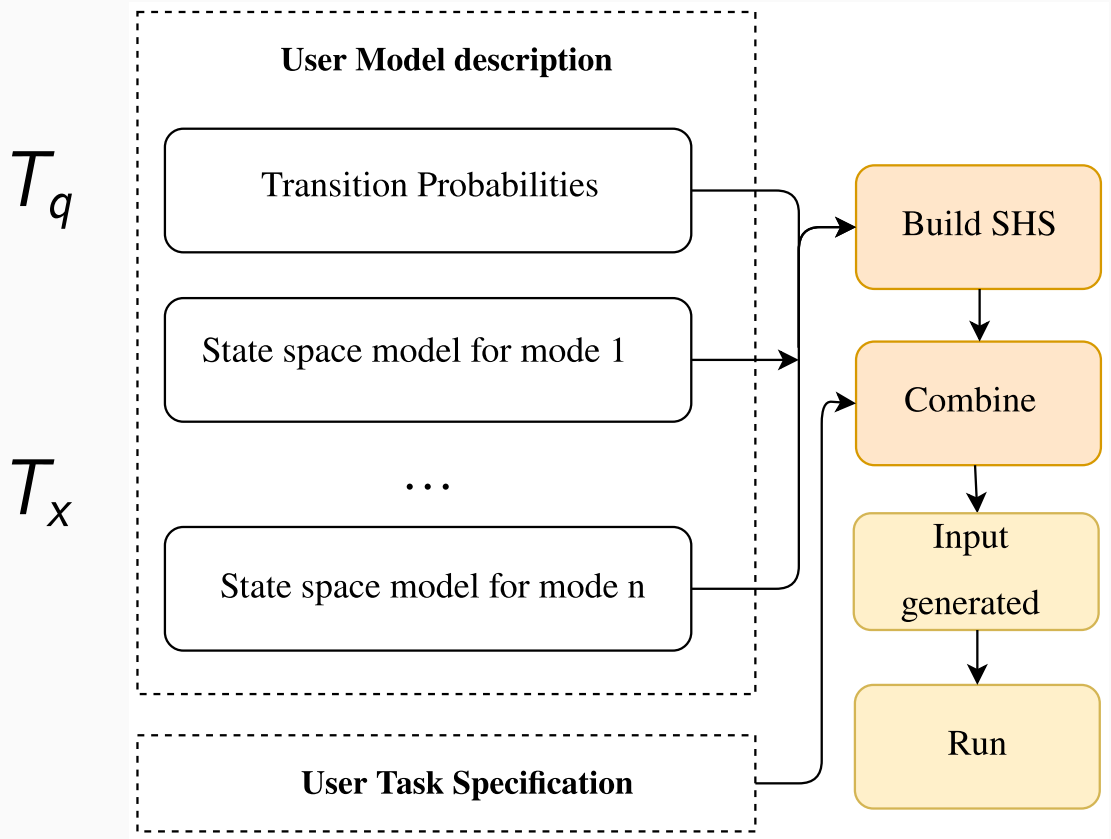
tested on machines under Linux OS

modularity

- comprises of independent libraries for different tasks :
  1. MDP
  2. IMDP
  3. simulator
- allows for seamless extensions with new or existing tools and methods

Overview of StocHy

user interaction via: `main`



Overview of StocHy

techniques for optimisation

- vector algebra for efficient handling of linear operations
- sparse matrices
- uses armadillo which applies multi-threading for speed up
- easy evaluation of stochastic kernels via symbolic methods

Overview of StocHy

techniques for optimisation

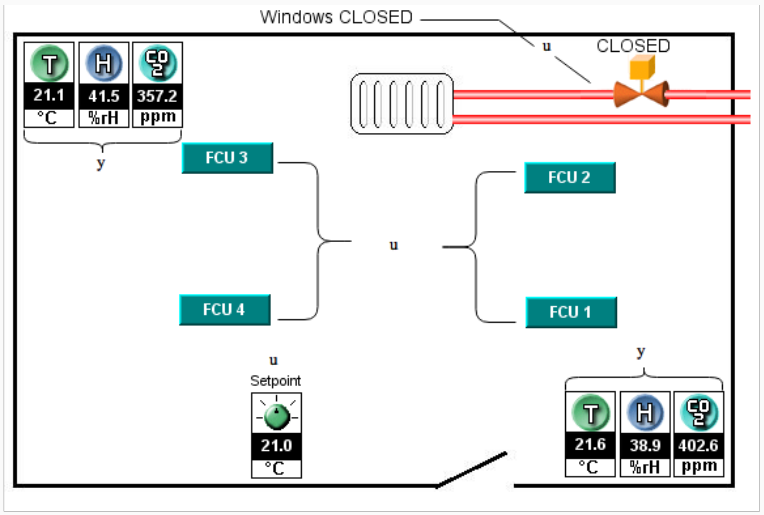
- vector algebra for efficient handling of linear operations
- sparse matrices
- uses armadillo which applies multi-threading for speed up
- easy evaluation of stochastic kernels via symbolic methods

output interface

- summary of results in command line
- python scripts for plot generation
- store results in data files
- option export to PRISM or STORM

Case studies

- inspired from our smart buildings laboratory
- mixed digital controls & continuous processes
- models with different trade-offs of complexities
- set of benchmarks presented at ADHS'18[1]



[1] Benchmarks for stochastic models from building automation systems; N Cauchi, A Abate IFAC-PapersOnLine 51 (16), 49-54

Case study 1: formal verification

model CO_2 and temperature dynamics in a zone

- 1 discrete mode, 2 continuous variables, no control action
- continuous variable evolve according to

$$x_{1,k+1} = x_{1,k} + \frac{\Delta}{V}(-\rho_m x_{1,k} + \rho_c(C_{out} - x_{1,k})) + \sigma_1 w_{1,k}$$

$$x_{2,k+1} = x_{2,k} + \frac{\Delta}{C_z}(\rho_m C_{pa}(T_{set} - x_{2,k}) + \frac{\rho_c}{R}(T_{out} - x_{2,k})) + \sigma_2 w_{2,k}$$

x_1: zone CO_2 level

x_2 : zone temperature level

$\sigma_{(.)}$: variance of noise $w_{.,k} \sim \mathcal{N}(0,1)$

$\rho_{(.)}$: fixed air flow

C_{out} :outside CO_2 level

T_{set}: set temperature

T_{out}: outside temperature

Case study 1: formal verification

model CO_2 and temperature dynamics in a zone

- 1 discrete mode, 2 continuous variables, no control action
- continuous variable evolve according to

$$X_{k+1} = AX_k + F + GW_k$$

task to check for zone comfort

- check if CO_2 and temperature levels remain within

$$\varphi_1 := \square^{\leq K} ([405, 540] \times [18, 24])$$

- verification task via MDP library

Case study 1: formal verification

implementation:

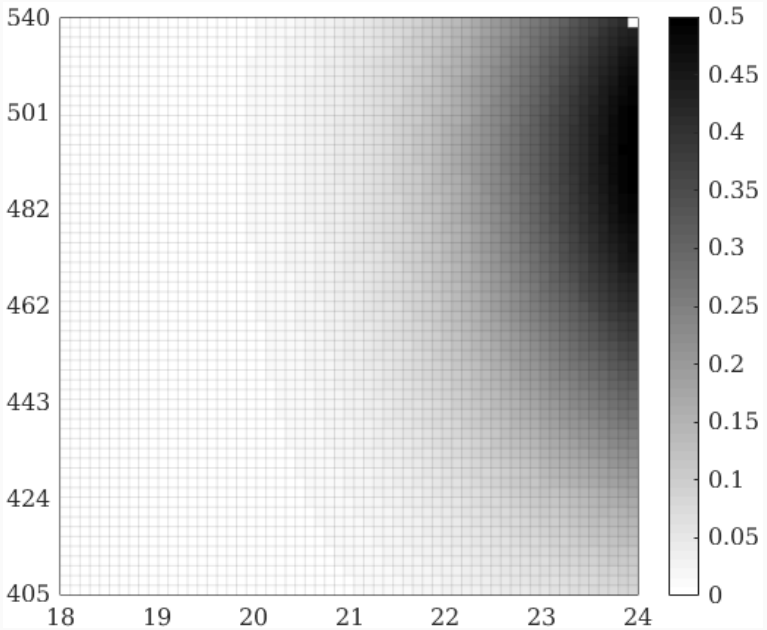
1. define state-space model
   - directly within main file
   - read from `.mat` file
2. define task as verification using MDP with $k = 3$

```
1  // Dynamics definition
2  arma::mat Aq0 = {{0.935, 0},{0, 0.9157}};
3  arma::mat Fq0 = {{0.03},{0.11}};
4  arma::mat Gq0 = {{1.7820, 0},{0, 0.5110}};
5  ssmodels_t model(Aq0,Fq0,Gq0);
6  shs_t<arma::mat,int> cs1SHS(model);
7  // safe set
8  arma::mat safe = {{405,540},{18,24}};
9  // task spec
10 taskSpec_t mySpec(mdp,3,verify_safety,
11 safe,eps,uniform);
12 //combine
13 inputSpec_t<arm::mat,int>
       myInput(mySHS,mySpec);
14 //run
15 performtask(myInput);
```

main file

results:

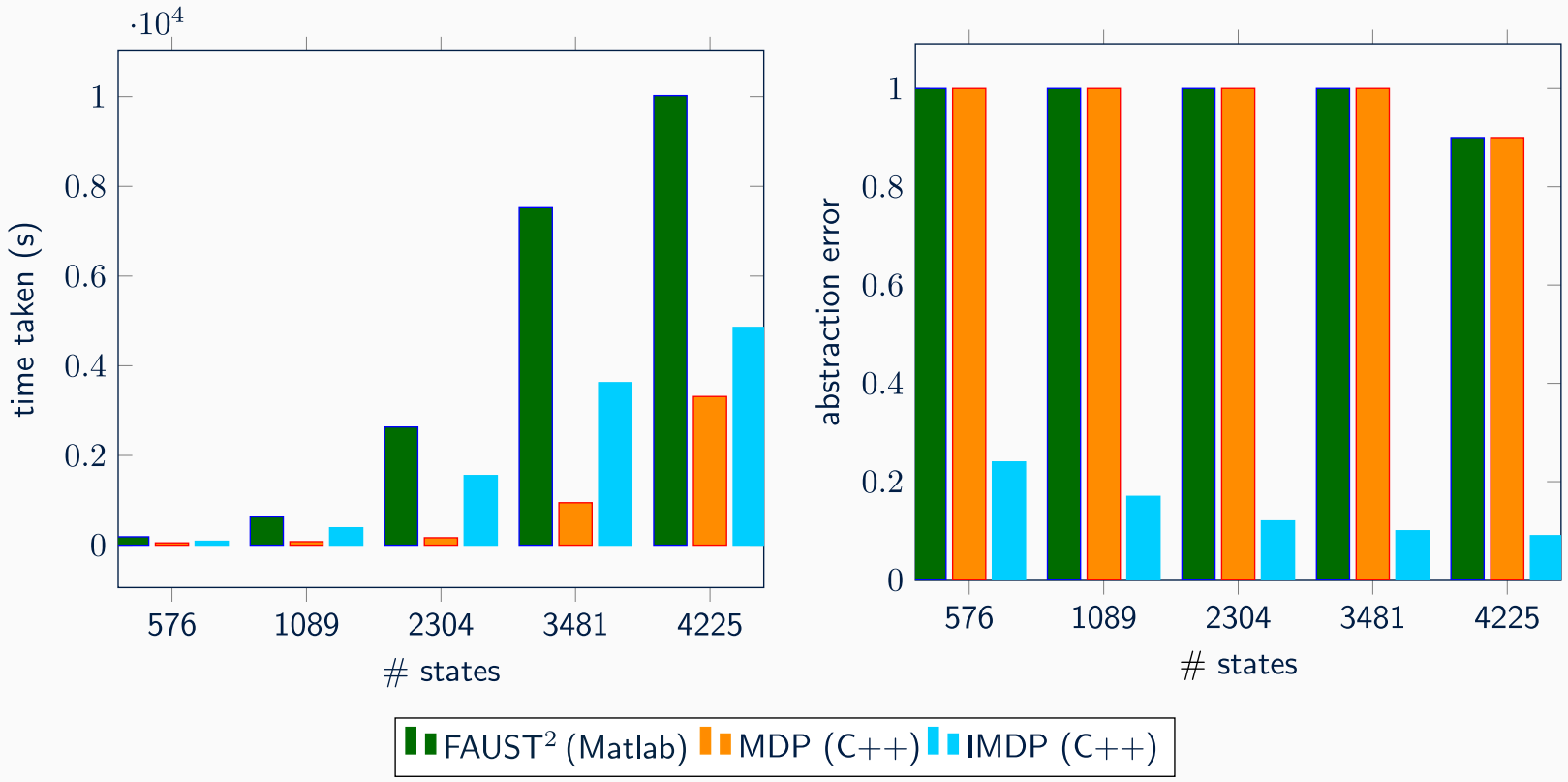
- maximal safety probability: 0.5
- total number of states: 3481
- computational time: 946.29 [s]



probability of satisfiability

comparison:

- repeated verification using both abstraction methods
- varied maximal abstraction error
- benchmarked against FAUST^2



Case study 2: strategy synthesis

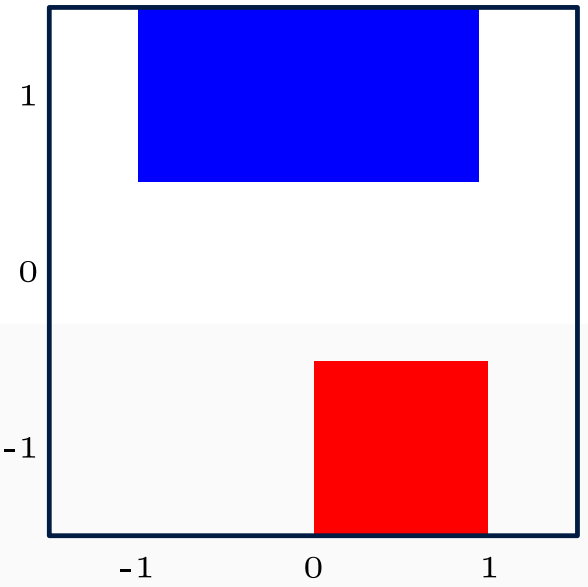
switched mode model

- 2 discrete modes: $\mathcal{Q} = \{q_0, q_1\}$
- 2 continuous variables evolve according to:

$$X_{k+1} = A_{q_i} X_k + G_{q_i} W_k$$

$$A_{q_0} = \begin{bmatrix} 0.43 & 0.52 \\ 0.65 & 0.12 \end{bmatrix} \quad G_{q_0} = \begin{bmatrix} 1 & 0.1 \\ 0 & 0.1 \end{bmatrix}$$

$$A_{q_1} = \begin{bmatrix} 0.65 & 0.12 \\ 0.52 & 0.43 \end{bmatrix} \quad G_{q_1} = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}$$



Case study 2: strategy synthesis

switched mode model

- 2 discrete modes: $\mathcal{Q} = \{q_0, q_1\}$
- 2 continuous variables evolve according to:

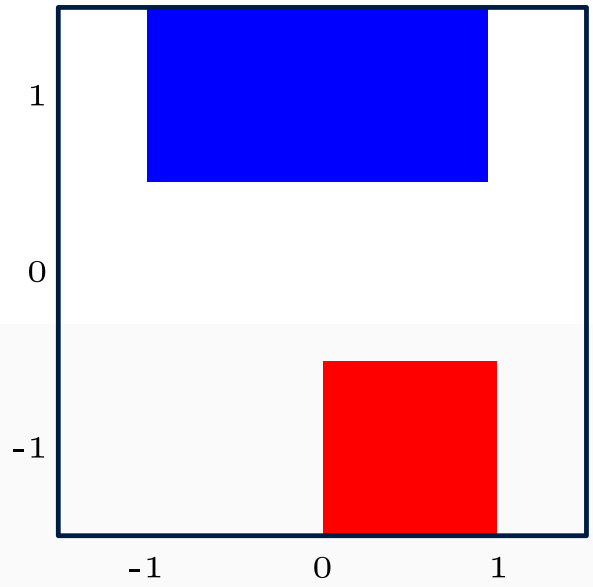
$$X_{k+1} = A_{q_i} X_k + G_{q_i} W_k$$

constrained reachability

- optimal strategy to satisfy

$$\varphi_2 := (\neg \mathit{red})\ \mathsf{U}\ \mathit{blue}$$

- synthesis task via IMDP library



Case study 2: strategy synthesis

implementation:

1. define state-space model
   - directly within main file
   - read from `.mat` file
2. define task as synthesis using IMDP

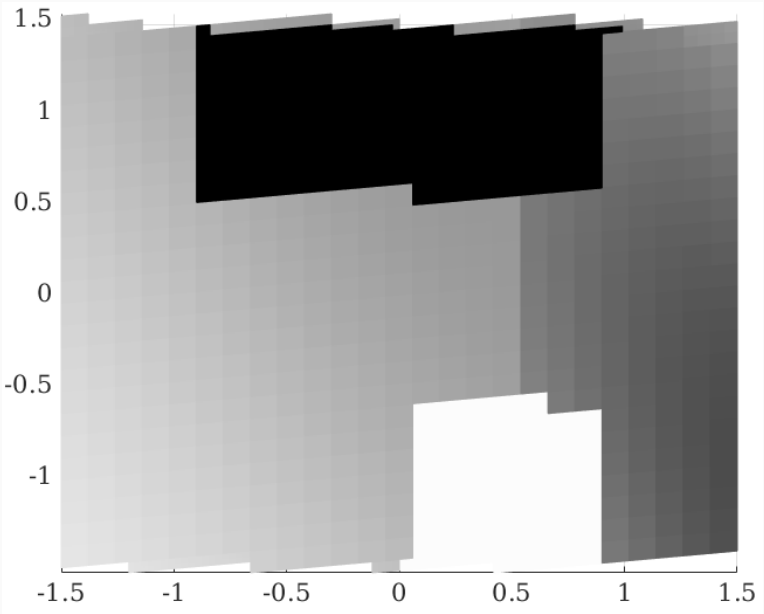
```
1  // Dynamics definition
2  shs_t<arma::mat,int>
       mySHS('../CS2.mat');
3  // domain
4  arma::mat bound = {{-1.5, 1.5}, {-1.5,
       1.5}};
5  // Define grid size for each dimension
6  arma::mat grid = {{0.12, 0.12}};
7  // task spec
8  taskSpec_t
       mySpec(imdp,-1,synthesis_reachavoid,
9  bound,grid);
10 //combine
11 inputSpec_t<arm::mat,int>
       myInput(mySHS,mySpec);
12 //run
13 performtask(myInput);
```

main file

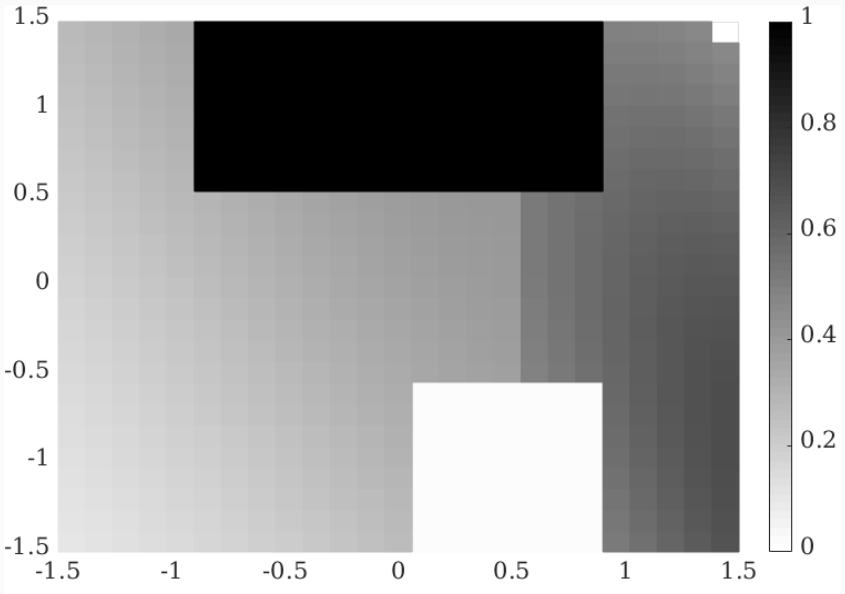
Case study 2: strategy synthesis

results:

- maximal abstraction error: 0.21
- total number of states: 2410
- computational time: 1639.30 [s]



lower bound probability for q_0



lower bound probability for q_1

Case study 3: scaling in continuous dimension of model

model with varying dimensions

- 1 discrete mode, d continuous variables, no control action
- continuous variable evolve according to

$$X_{k+1} = 0.81_d X_k + 0.21_d W_k$$

Case study 3: scaling in continuous dimension of model

model with varying dimensions

- 1 discrete mode, d continuous variables, no control action
- continuous variable evolve according to

$$X_{k+1} = 0.81_d X_k + 0.21_d W_k$$

model checking for different d dimensions

- property

$$\varphi_3 := \square\, [-1, 1]^d$$

- verification via IMDP method

Case study 3: scaling in continuous dimension of model

implementation:

1. define state-space model
   - directly within main file
   - read from `.mat` file
2. define task as verification via IMDP
3. repeat for all dimensions (automated)

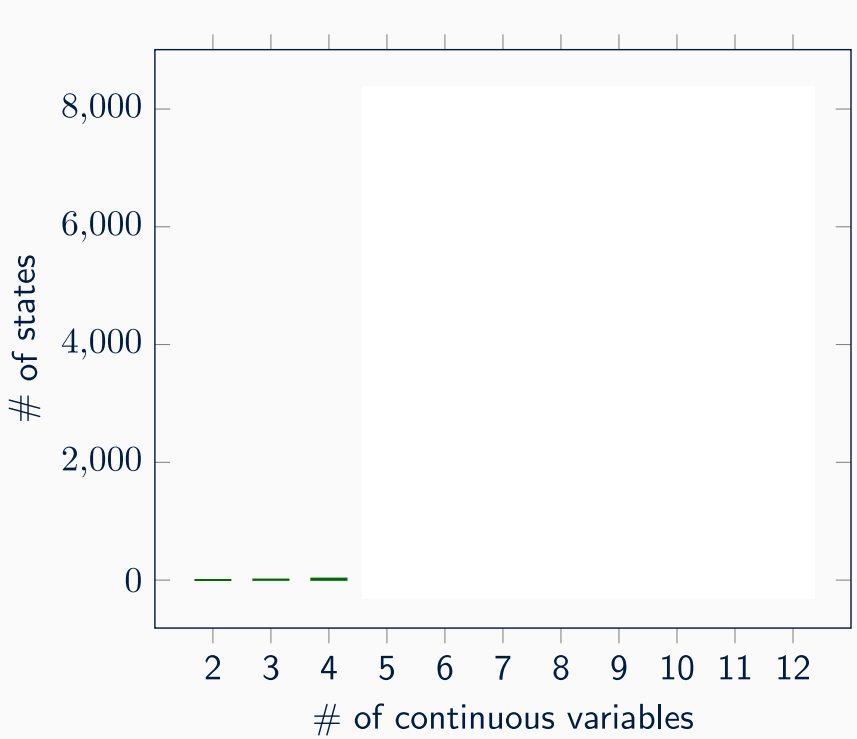
```
1  // Dynamics definition
2  shs_t<arma::mat,int> mySHS('../CS3.mat');
3  // domain
4  arma::mat bound = {{-1, 1}, {-1, 1}};
5  // Define grid size for each dimension
6  arma::mat grid = {{0.1, 0.1}};
7  // task spec
8  taskSpec_t mySpec(imdp,-1,verify_safety,
9  bound,grid);
10 //combine
11 inputSpec_t<arm::mat,int>
       myInput(mySHS,mySpec);
12 //run
13 performtask(myInput);
```

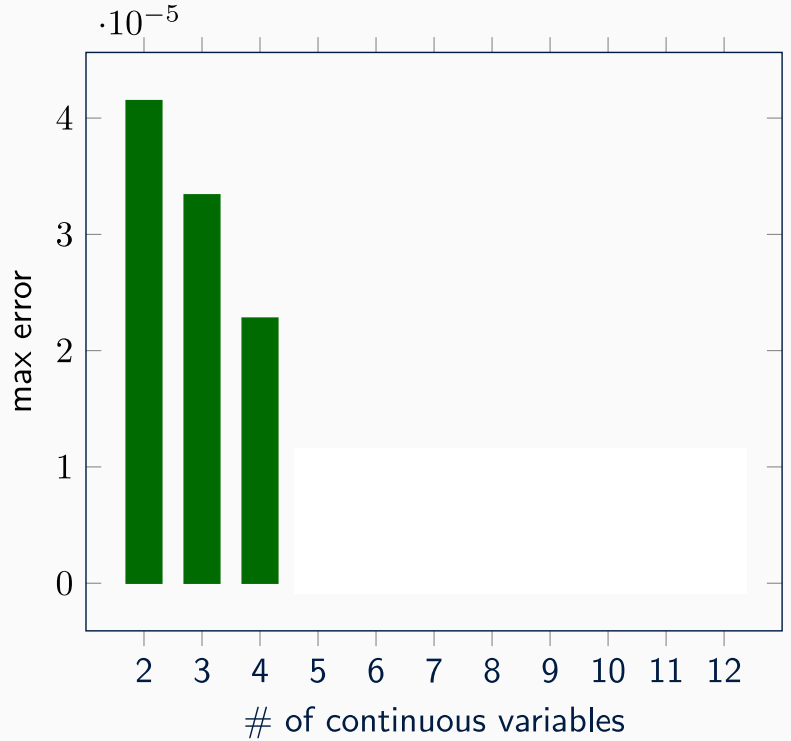
main file

Case study 3: scaling in continuous dimension of model

results:

- abstract models with manageable state spaces
- scalability with respect to the continuous dimension d
- marked improvement over state-of-the-art tools

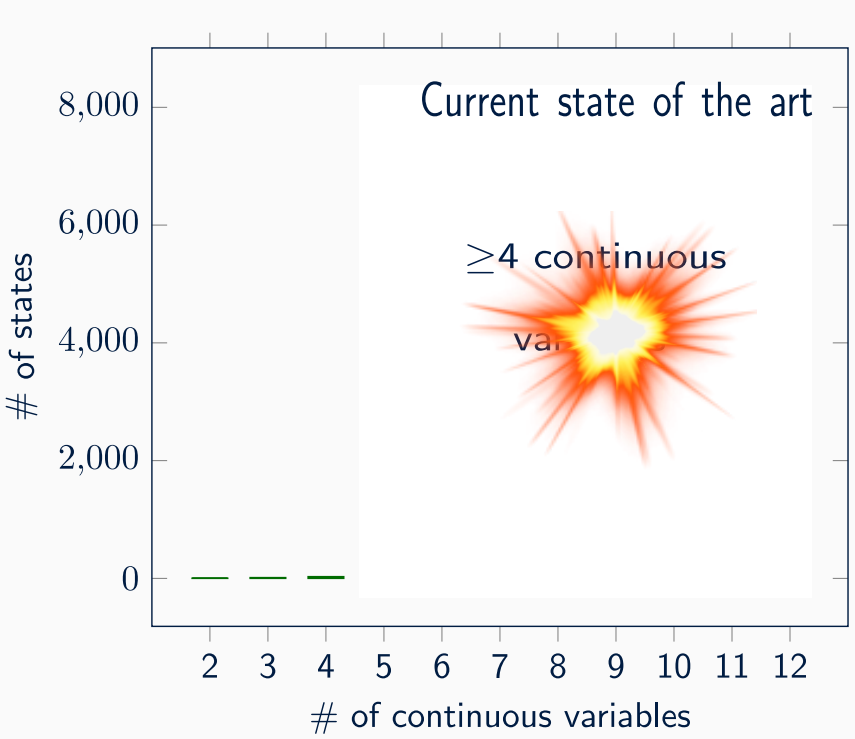


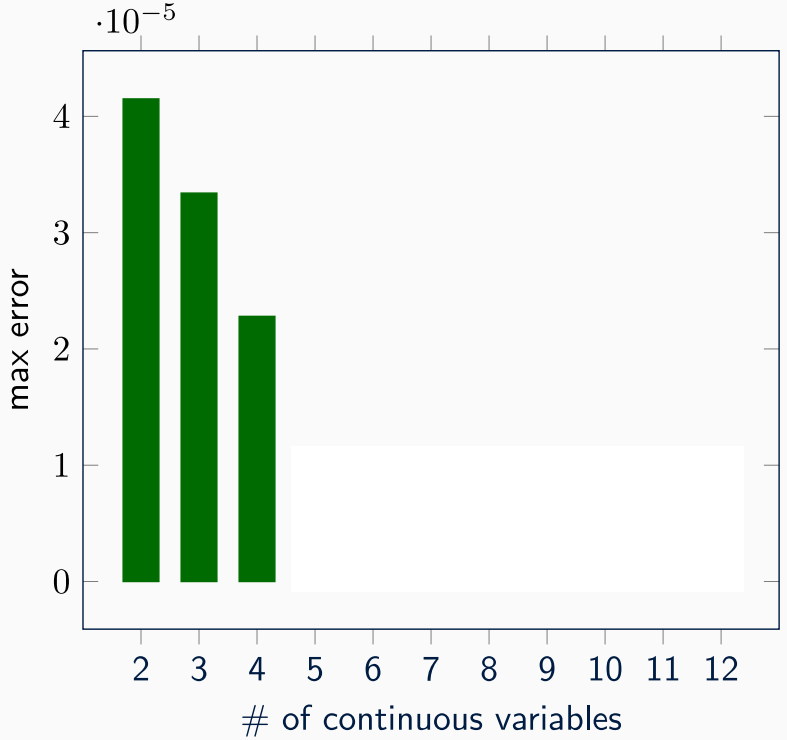


Case study 3: scaling in continuous dimension of model

results:

- abstract models with manageable state spaces
- scalability with respect to the continuous dimension d
- marked improvement over state-of-the-art tools

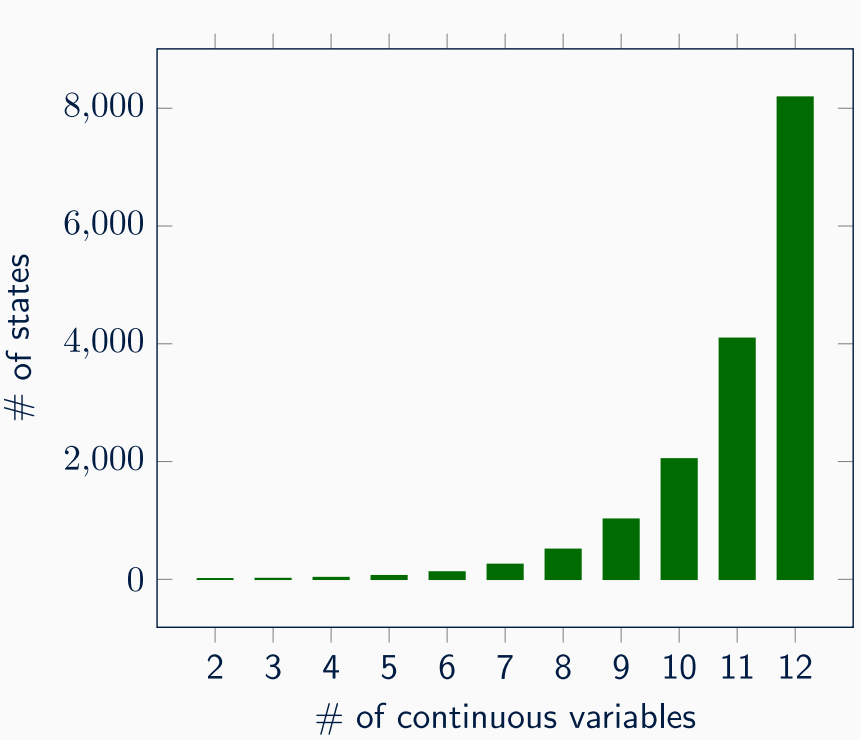


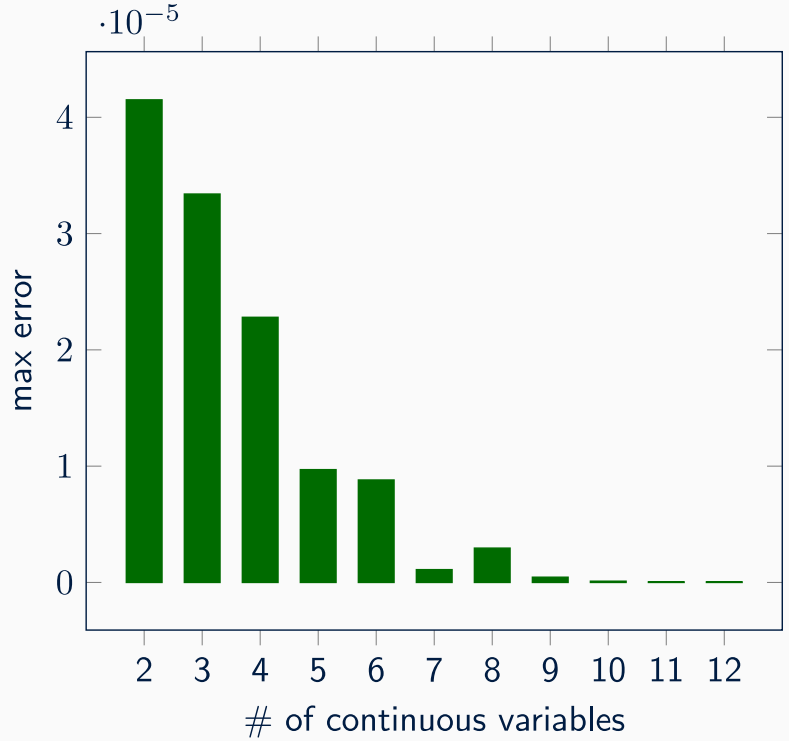


Case study 3: scaling in continuous dimension of model

results:

- abstract models with manageable state spaces
- scalability with respect to the continuous dimension d
- marked improvement over state-of-the-art tools



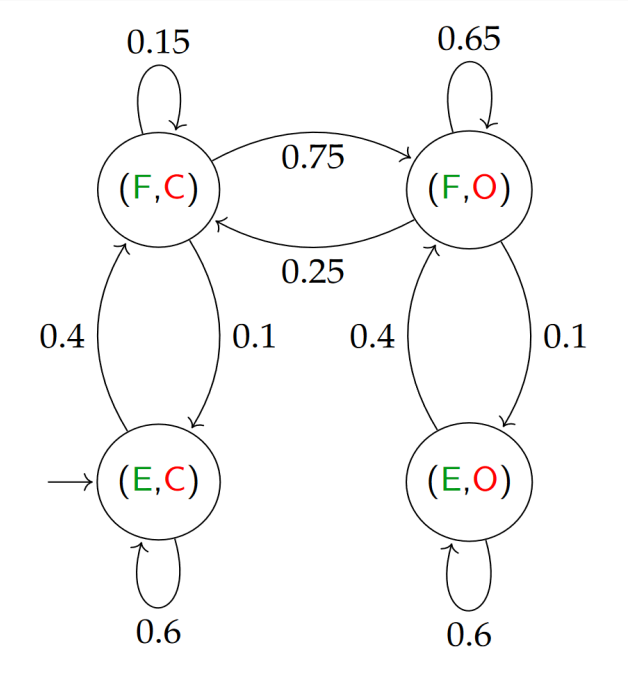


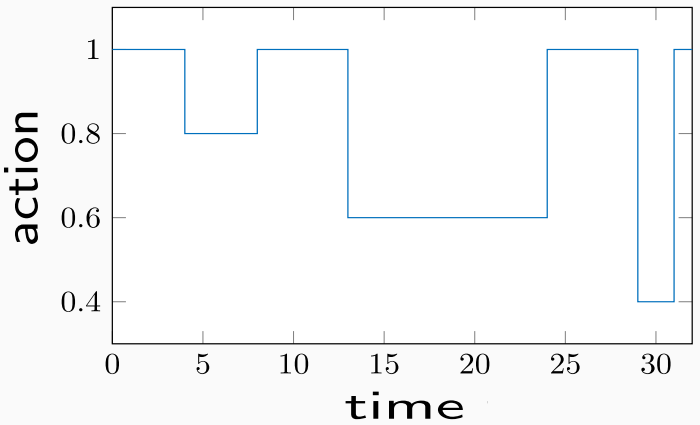
Case study 4: simulations

extended model of CO_2 and temperature in zone

- 4 discrete modes, 2 continuous variables

$$X_{k+1} = A_{q_i} X_k + B_{q_i} U_k + X_k \sum_{i=1}^{v} N_{q,i} U_{k,i} + F_q + G_{q_i} W_k$$





Case study 4: simulations

implementation:

1. define state-space model
   - directly within main file
   - read from `.mat` file
2. define task as simulation
3. define initial conditions
4. define time horizon

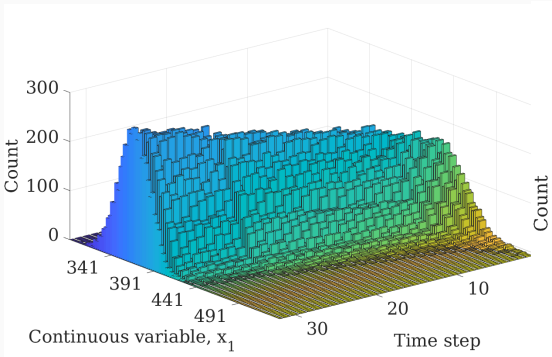
```
1  // Initial continuous variables
2  arma::mat x_init
       =readInputSignal("../x_init.txt");
3  // initial discrete mode q_0 = (E,C)
4  double q_init = 0;
5  // definition of control signal
6  arma::mat u
       =readInputSignal("../u.txt");
7  // combine data signals
8  exdata_t data(x_init,u,q_init);
9  shs_t<arma::mat, int>
       mySHS("../CS4.mat", data);
10 // task spec
11 taskSpec_t mySpec(simulator,32);
12 //combine
13 inputSpec_t<arm::mat,int>
       myInput(mySHS,mySpec);
14 //run
15 performtask(myInput);
```

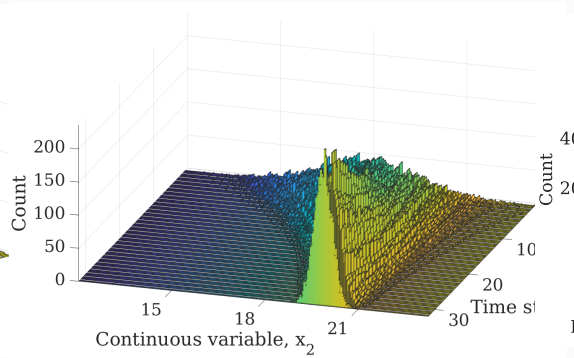
main file

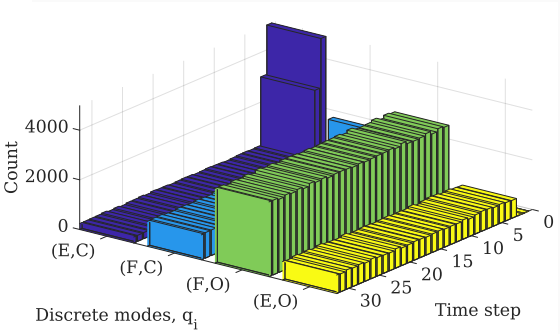
Case study 4: simulations

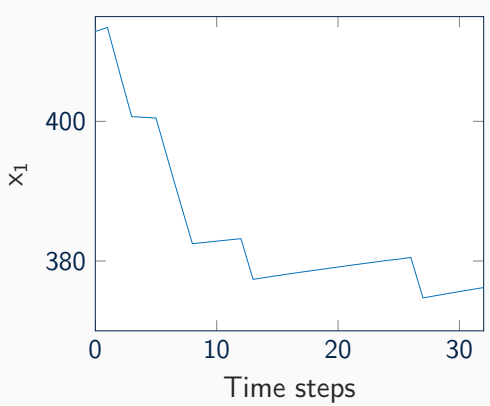
results:

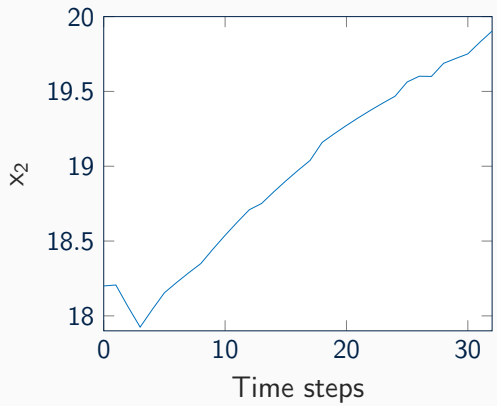
- computational time: 48.6 [s]

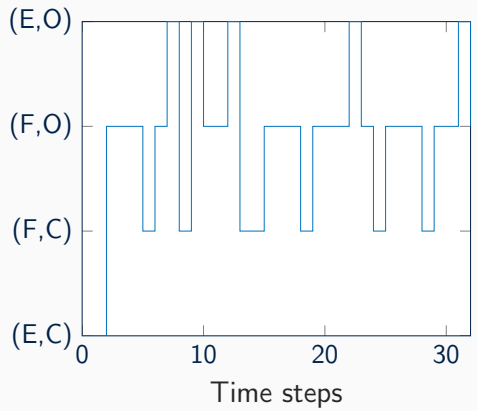












Conclusion

contributions

- new tool called StocHy
- aid for understanding behaviour of SHS
- extendable and modular
- further examples and benchmarks together with tool at

`gitlab.com/natchi92/StocHy`

Conclusion

contributions

- new tool called StocHy
- aid for understanding behaviour of SHS
- extendable and modular
- further examples and benchmarks together with tool at

`gitlab.com/natchi92/StocHy`

future work

- employ a graphical user-interface
- analysis of continuous-time (current work in progress)

Conclusion

contributions

- new tool called StocHy
- aid for understanding behaviour of SHS
- extendable and modular
- further examples and benchmarks together with tool at

`gitlab.com/natchi92/StocHy`

Thank you!

`nathalie.cauchi@cs.ox.ac.uk`