Cyber-Physical Systems

Laura Nenzi

Università degli Studi di Trieste
II Semestre 2018

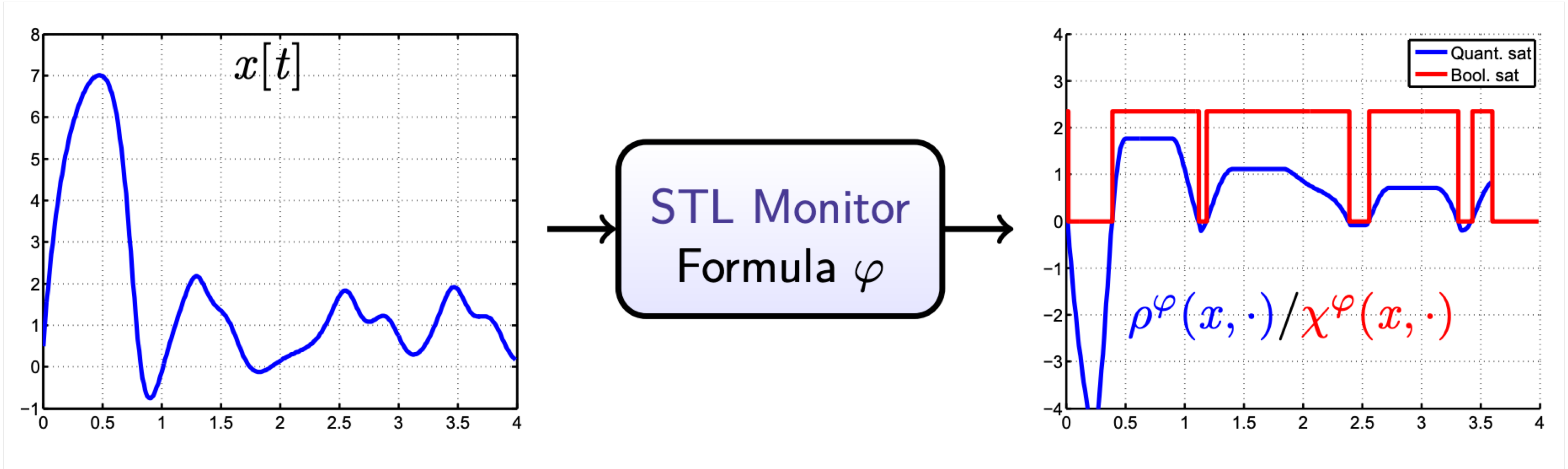
Lecture 10: STL applications

[Many Slides due to J. Deshmukh, S. Silvetti and E. Bartocci]

Terminology

- **Syntax**: A set of syntactic rules that allow us to construct formulas from specific ground terms
- **Semantics**: A set of rules that assign meanings to well-formed formulas obtained by using above syntactic rules
- **Model-checking/Verification**: $M \vDash \phi \iff \forall \mathbf{x} \in trace(M) \;\; \beta(\varphi, \mathbf{x}, 0) = 1$
- **Monitoring**: computing β for a single trace $\mathbf{x} \in trace(M)$
- **Statistical Model Checking**: "doing statistics" on $\beta(\varphi, \mathbf{x}, 0)$ for a finite-subset of $trace(M)$

STL Monitor



A robust STL monitor is a transducer that transform x into Boolean or a quantitative signal

Closed-loop Models

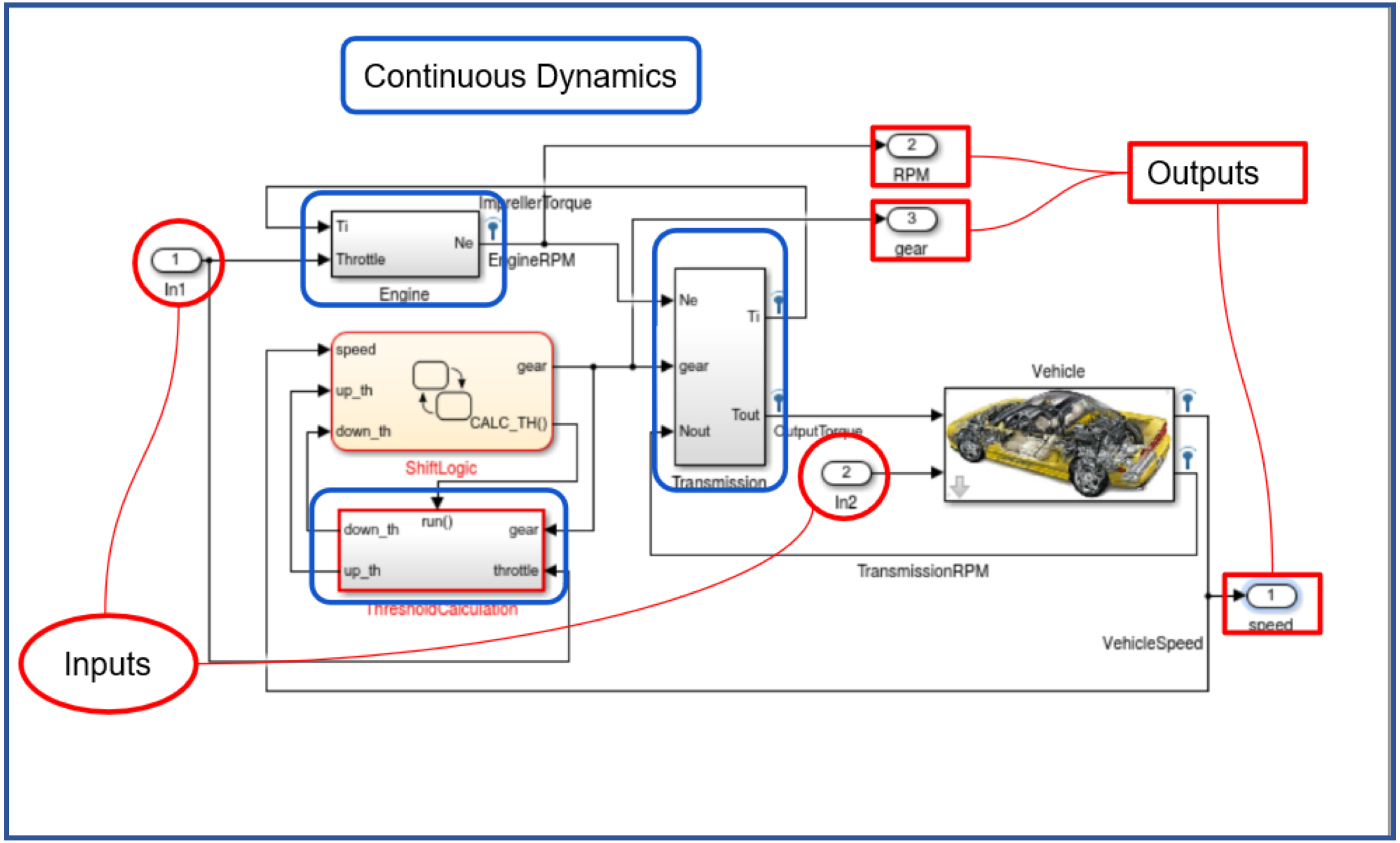
- Closed-loop Models contain:
  - Dynamics describing Physical Processes (Plant)
  - Code describing Embedded Control, Sensing, Actuation
  - Models of connection between plant and controller (hard-wired vs. wired network vs. wireless communication)

Example

Inputs:

Throttle

Brake



Outputs:

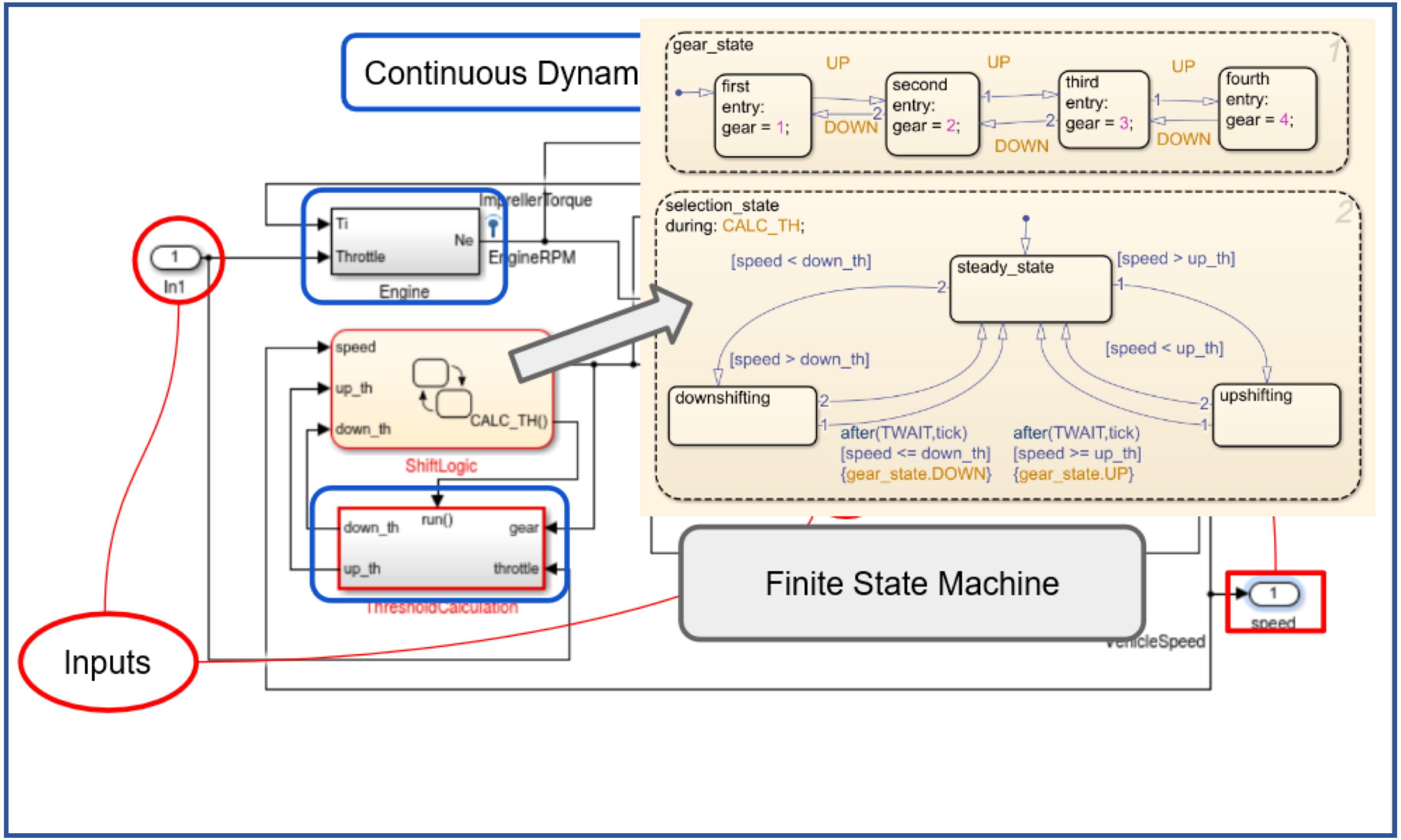
RPM

Gear

Speed

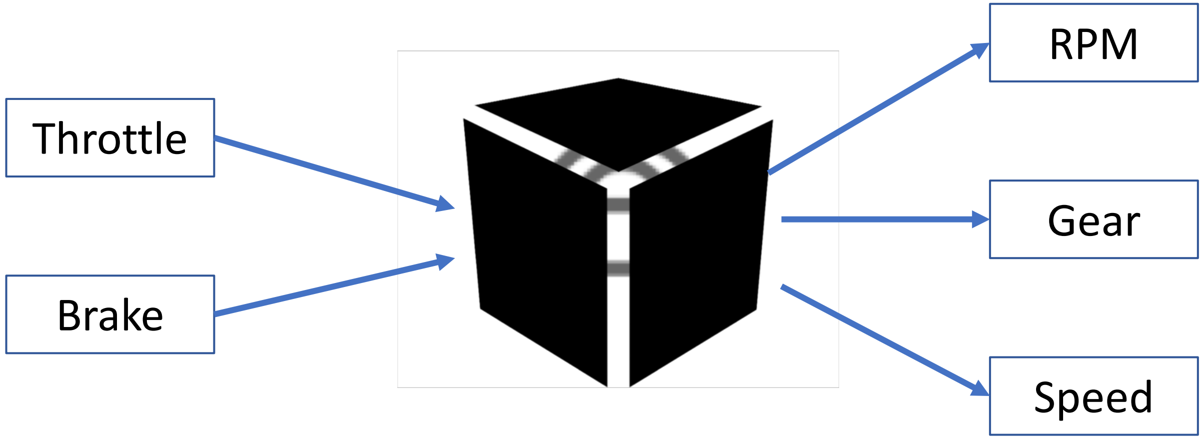
Simulink model of a Car Automatic Gear Transmission Systems

Example



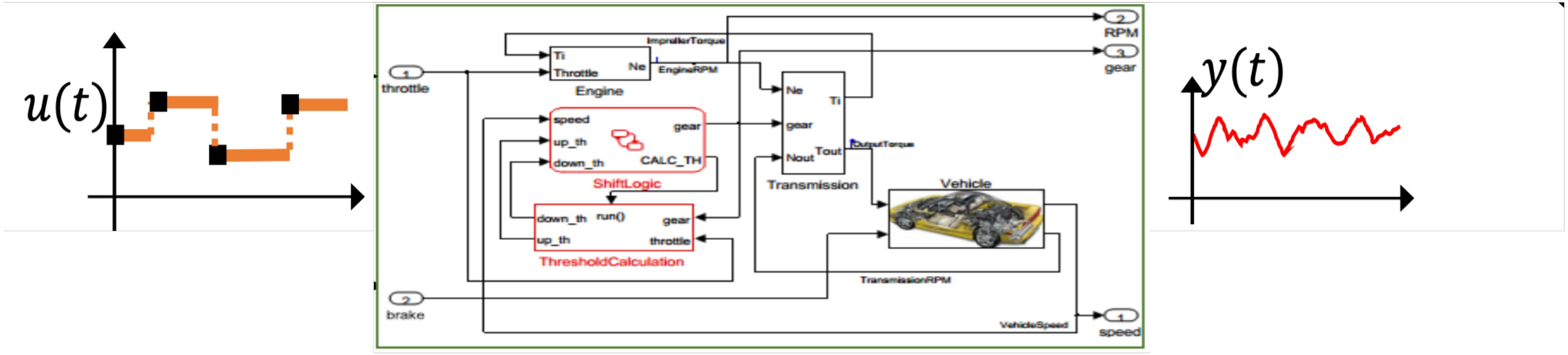
Continuous Dynam
gear_state
first
entry:
gear = 1;
second
entry:
gear = 2;
third
entry:
gear = 3;
fourth
entry:
gear = 4;
UP
DOWN
selection_state
during: CALC_TH;
steady_state
[speed < down_th]
[speed > up_th]
[speed > down_th]
[speed < up_th]
downshifting
upshifting
after(TWAIT,tick)
[speed <= down_th]
{gear_state.DOWN}
after(TWAIT,tick)
[speed >= up_th]
{gear_state.UP}
In1
Ti
Throttle
Ne
Engine
ImpellerTorque
EngineRPM
speed
up_th
down_th
CALC_TH()
ShiftLogic
down_th
up_th
run()
gear
throttle
ThresholdCalculation
Inputs
Finite State Machine
VehicleSpeed
speed

Black Box Assumption



Black Box Assumption

- For simplicity, consider the composed plant model, controller and communication to be a model M that is excited by an input signal $\mathbf{u}(t)$ and produces some output signal $\mathbf{y}(t)$



Verification vs. Testing

- For simplicity, $\mathbf{u}$ is a function from $\mathbb{T}$ to $\mathbb{R}^m$; let the set of all possible functions representing input signals be U
- Verification Problem:

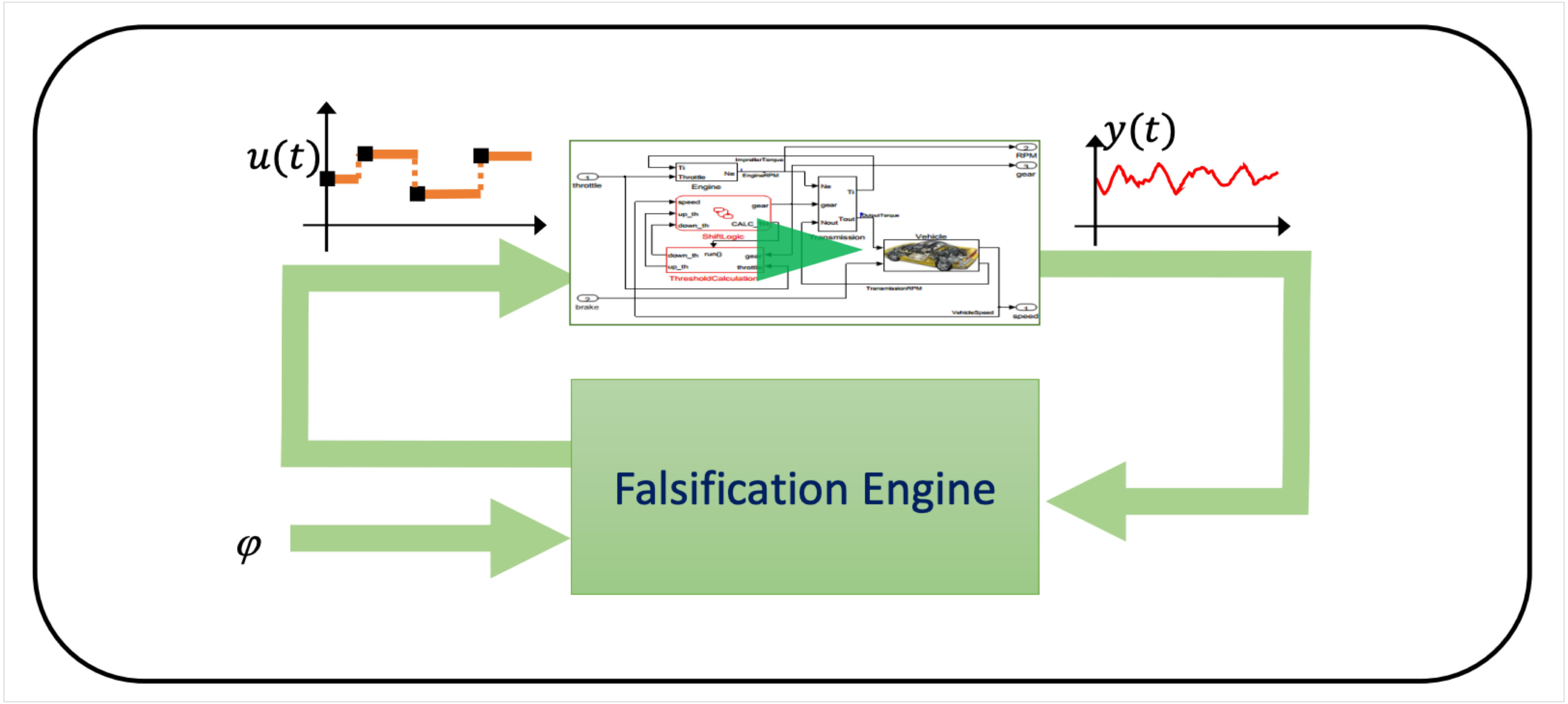
  Prove the following: $\forall \mathbf{u} \in U: \left(\mathbf{y} = M(\mathbf{u})\right) \wedge \varphi(\mathbf{u}, \mathbf{y})$
- Falsification/Testing Problem:

  Find a witness to the query: $\exists \mathbf{u} \in U : \left(\mathbf{y} = M(\mathbf{u})\right) \wedge \neg\varphi(\mathbf{u}, \mathbf{y})$
- These formulations are quite general, as we can include the following "*model uncertainties*" as input signals: Initial states, tunable parameters in both plant and controller, time-varying parameter values, noise, etc.,

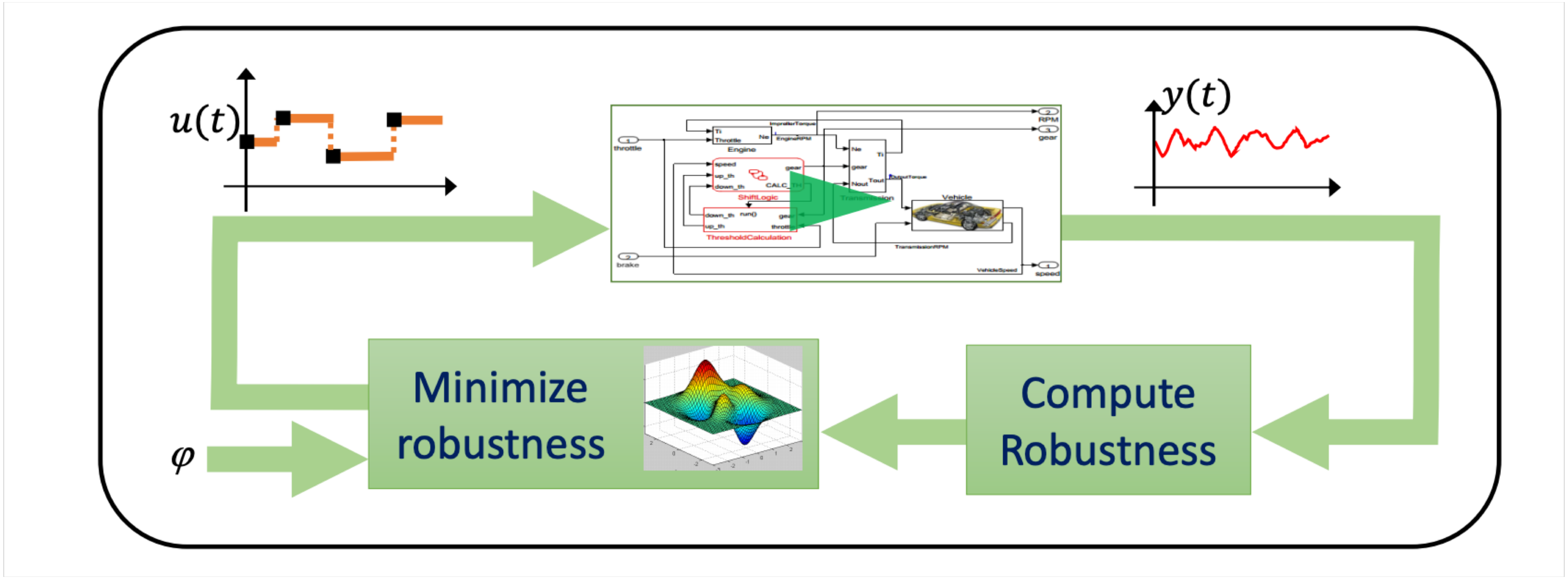
Challenges with real-world systems

- If plant model, software and communication is simple (e.g. linear models), then we can do formal analysis
- Most real-world examples have very complex plants, controllers and communication!
- Verification problem, in the most general case is ***undecidable***
  - it is proved to be impossible to construct an algorithm that always leads to a correct yes-or-no answer to the problem

Falsification/Testing



Falsification by optimization



Use robustness as a cost function to minimize with Black-box/Global Optimizers

Falsification/Testing

- Falsification or testing attempts to find one or more $\mathbf{u}$ signals such that $\neg\varphi(\mathbf{u}, M(\mathbf{u}))$ is true.

- In verification, the set $\mathbb{T}$ (the time domain) could be unbounded, in falsification or testing, the time domain is necessarily bounded, i.e. $\mathbb{T} \subseteq [0, T]$, where T is some finite numeric constant

- In verification the co-domain of $\mathbf{u}$, could be an unbounded subset of $\mathbb{R}^m$, in falsification, we typically consider some compact subset of $\mathbb{R}^m$

- For the i^{th} input signal component, let D_i denote its compact co-domain. Then the input signal $\mathbf{u}$ is a function from $\mathbb{T}$ to $D_1 \times \cdots \times D_m$, where $\mathbb{T} \subseteq [0, T]$
In simple words: input signals range over bounded intervals and over a bounded time horizon

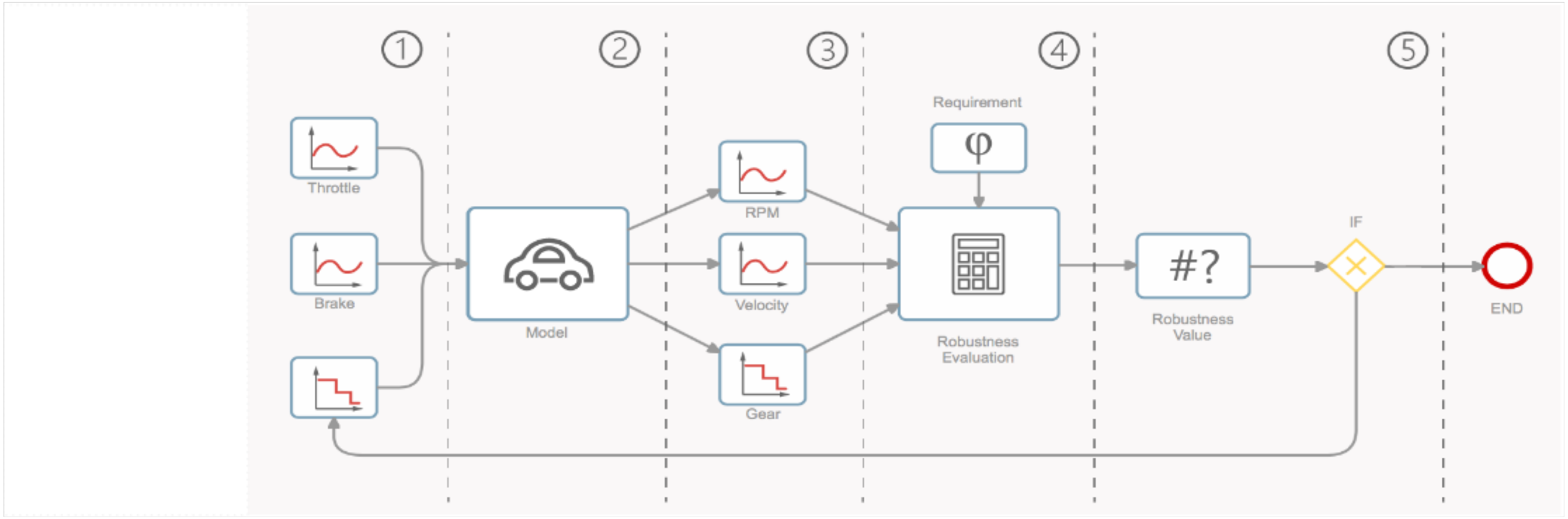
Falsification re-framed

Given:

- Set of all such input signals : U
- Input signal $\mathbf{u}$: function from $\mathbb{T}$ to $D_1 \times \cdots \times D_m$, where $\mathbb{T} \subseteq [0, T]$
- Model M that maps $\mathbf{u}$ to some signal $\mathbf{y}$ with the same domain as $\mathbf{u}$, and co-domain some subset of $\mathbb{R}^n$
- Property φ that can be evaluated to true/false over given $\mathbf{u}$ and $\mathbf{y}$

Check: $\exists \mathbf{u} \in U : \left(\mathbf{y} = M(\mathbf{u})\right) \wedge \neg\varphi(\mathbf{u}, \mathbf{y})$

Falsification CPS



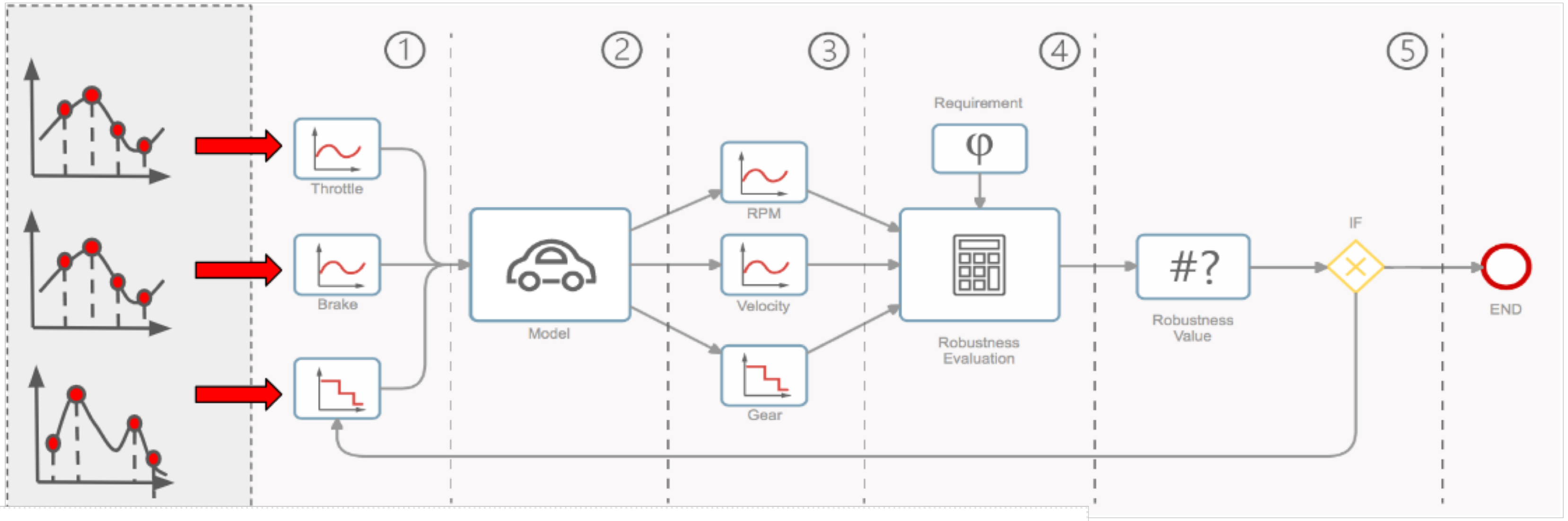
Goal:

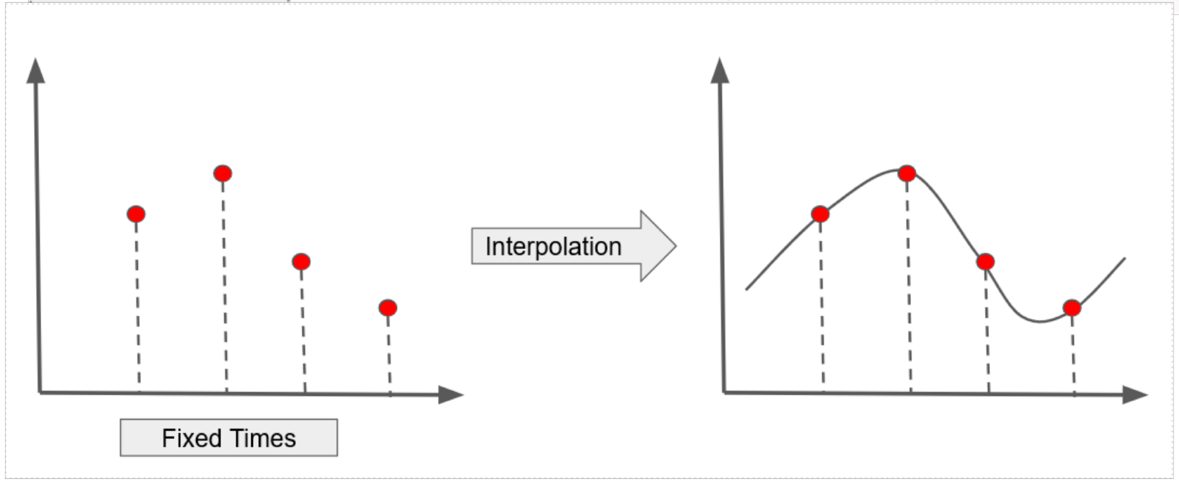
Find the inputs (1) which falsify the requirements (4)

Problems:

- Falsify with a low number of simulations — Active Learning
- Functional Input Space — Adaptive Parameterization

Adaptive Parameterization

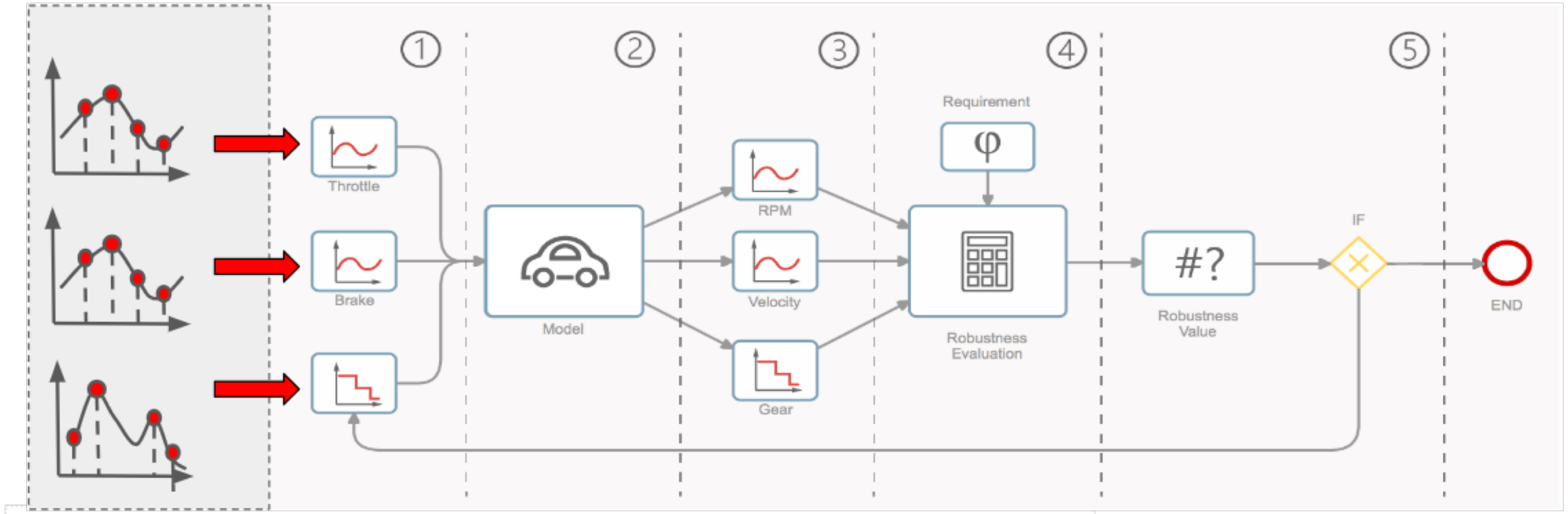


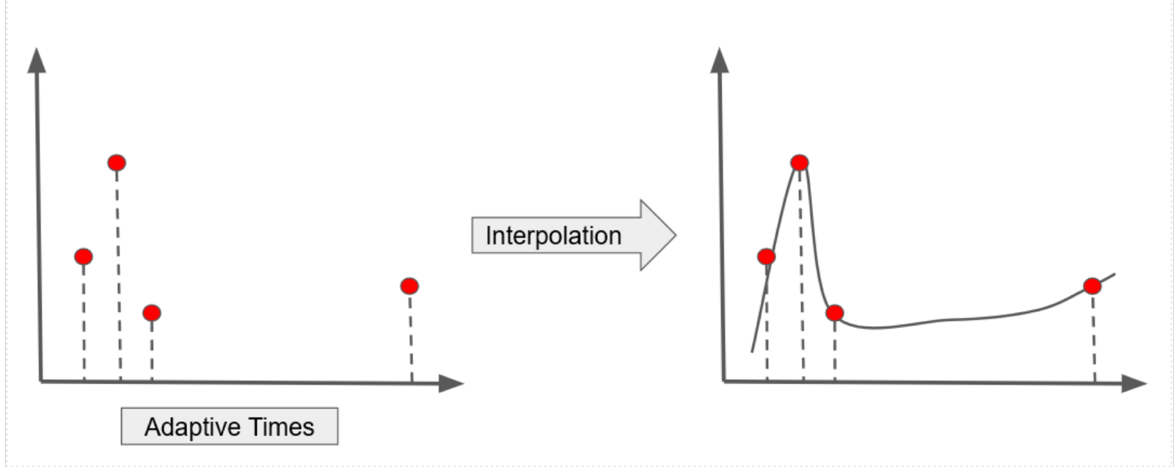


N Control points

N variable

Adaptive Parameterization





N Control points

2N variable

Problem

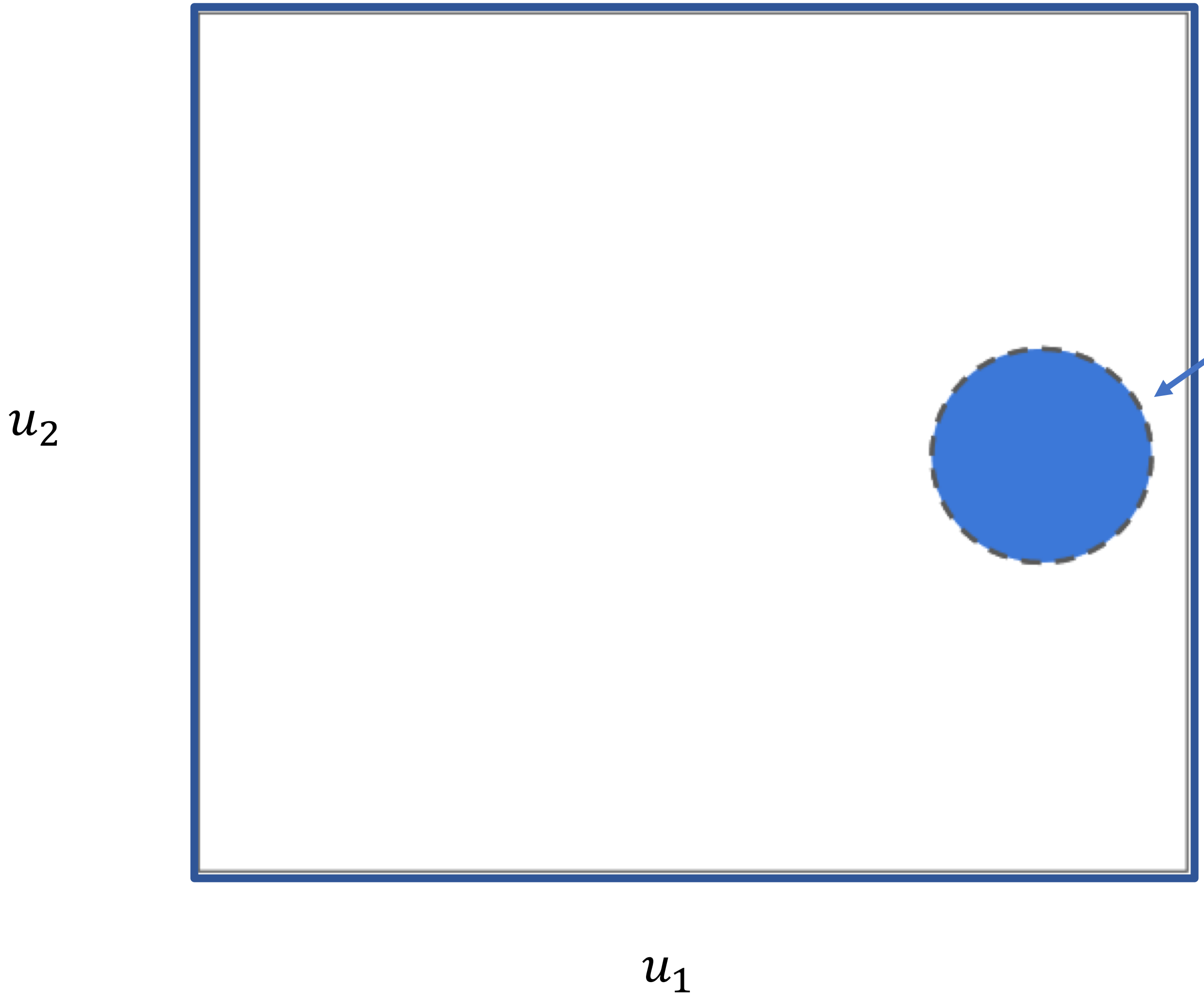
Find the trajectories which falsify the requirements:

$$B = \{\boldsymbol{u} \in \mathrm{U} \mid \rho, (M(\boldsymbol{u}), \phi) < 0\} \subseteq U$$

- **Training Set**: K= $\{\boldsymbol{u}_{\mathrm{i}}, \rho(M(\boldsymbol{u}_i), \phi))\}_{i \leq n}$ (the partial knowledge after n iterations)
- **Gaussian Process**: $\rho_K(\boldsymbol{u}) \sim GP(m_K(\boldsymbol{u}), \sigma_K(\boldsymbol{u}))$ (the partial model)

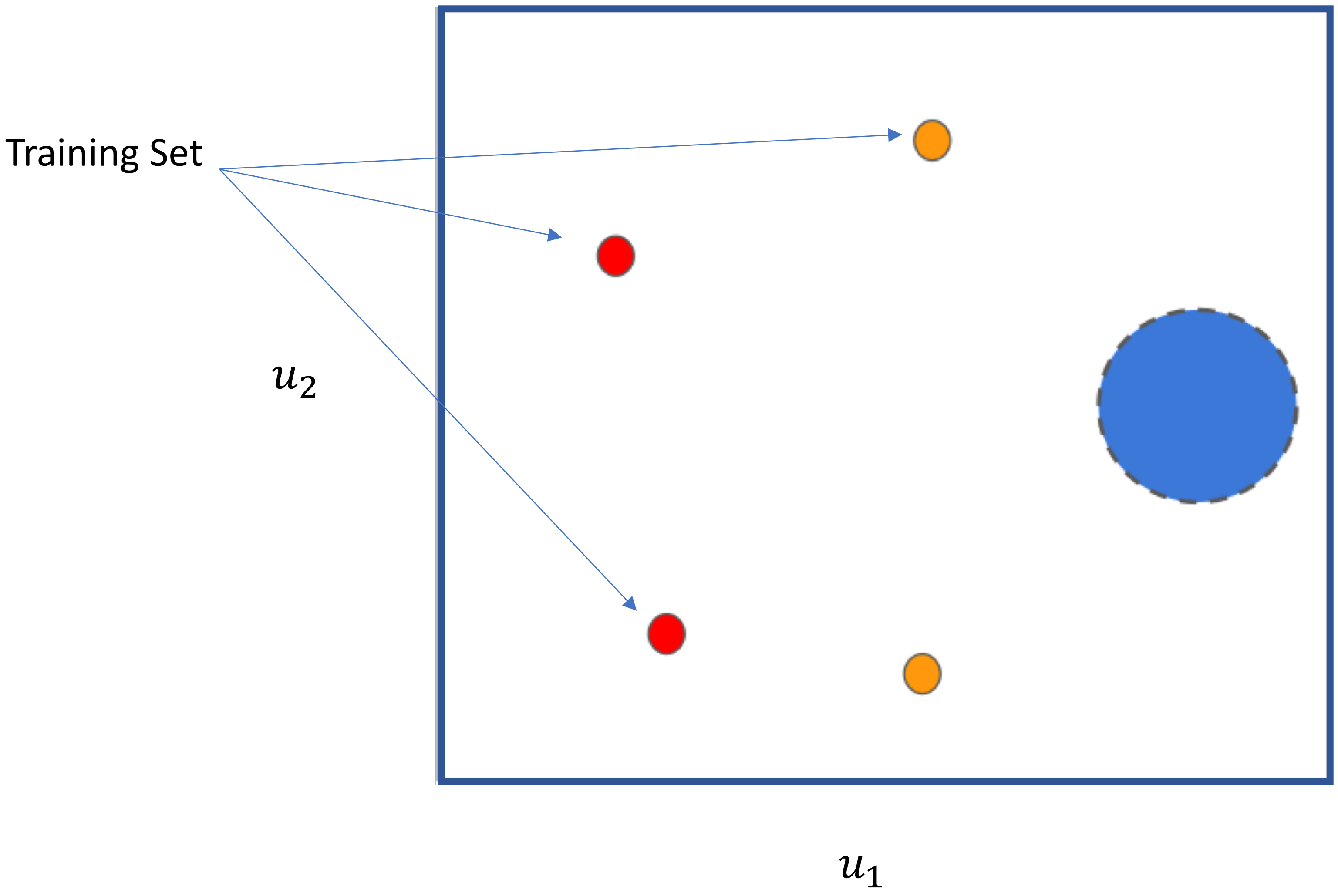
$$P(\rho_K(\boldsymbol{u}) < 0) = CDF\left(\frac{0 - m_K(\boldsymbol{u})}{\sigma_K(\boldsymbol{u})}\right)$$

Domain Estimation Algorithm (DEA)

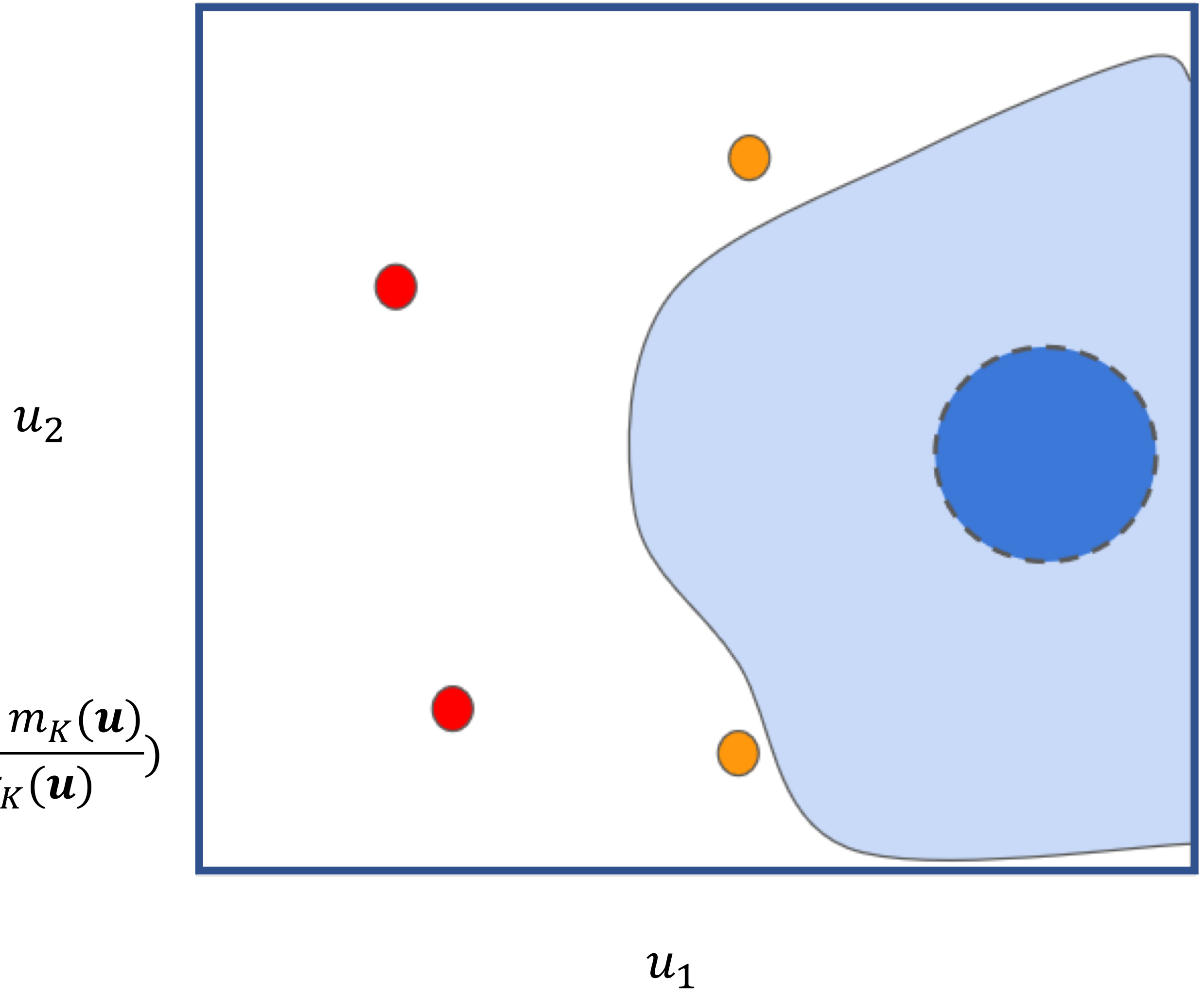


B= $\{\boldsymbol{u} \in \mathrm{U} \mid \rho(\, M(\boldsymbol{u}), \phi) < 0\}$

Domain Estimation Algorithm (DEA)



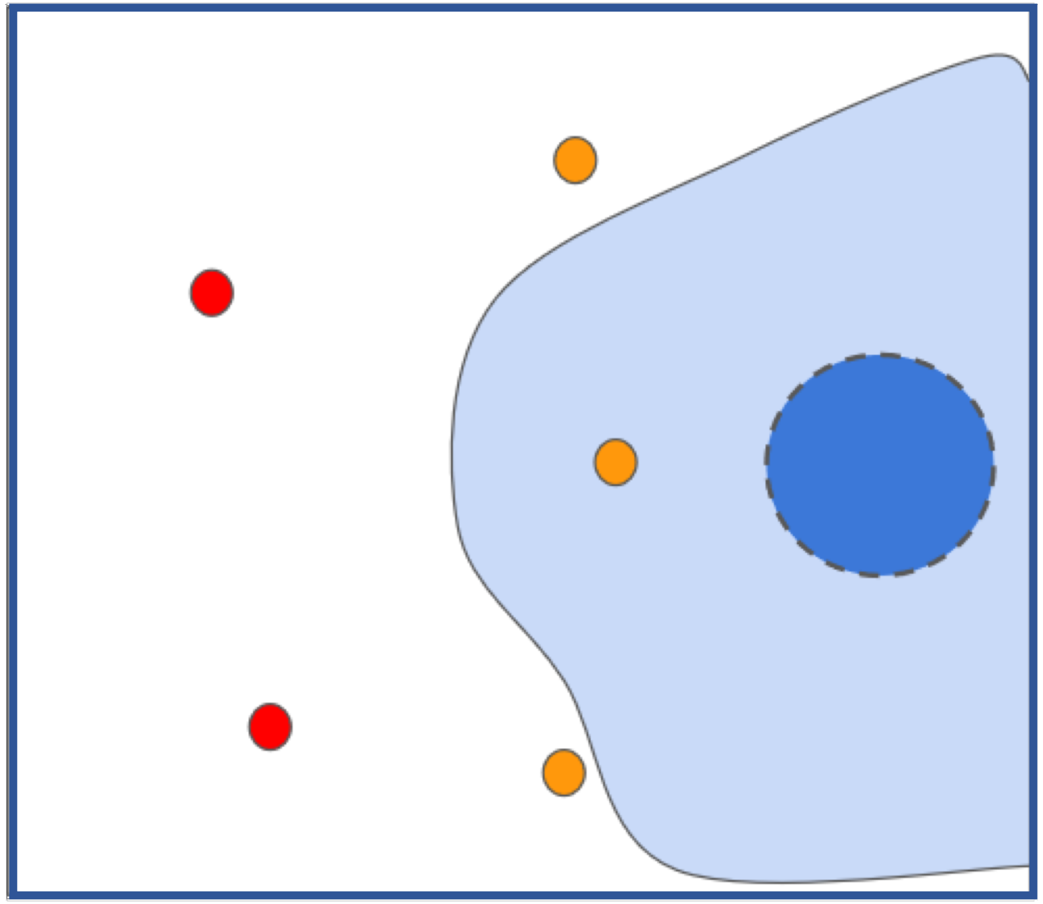
Domain Estimation Algorithm (DEA)



Sample a new point accordingly to:

$$P(\rho_K(\boldsymbol{u}) < 0) = CDF\left(\frac{0 - m_K(\boldsymbol{u})}{\sigma_K(\boldsymbol{u})}\right)$$

Domain Estimation Algorithm (DEA)

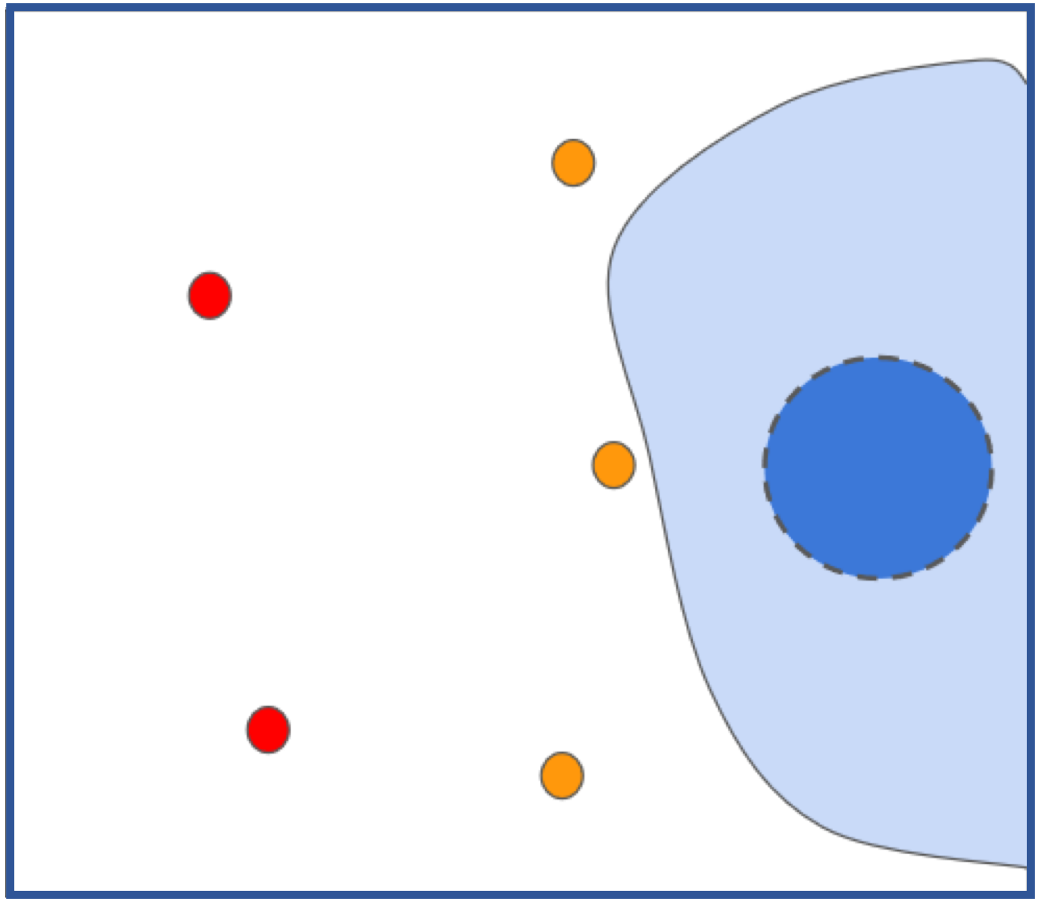


Sample a new point accordingly to:

$$P(\rho_K(\boldsymbol{u}) < 0) = CDF\left(\frac{0 - m_K(\boldsymbol{u})}{\sigma_K(\boldsymbol{u})}\right)$$

Domain Estimation Algorithm (DEA)

u_2

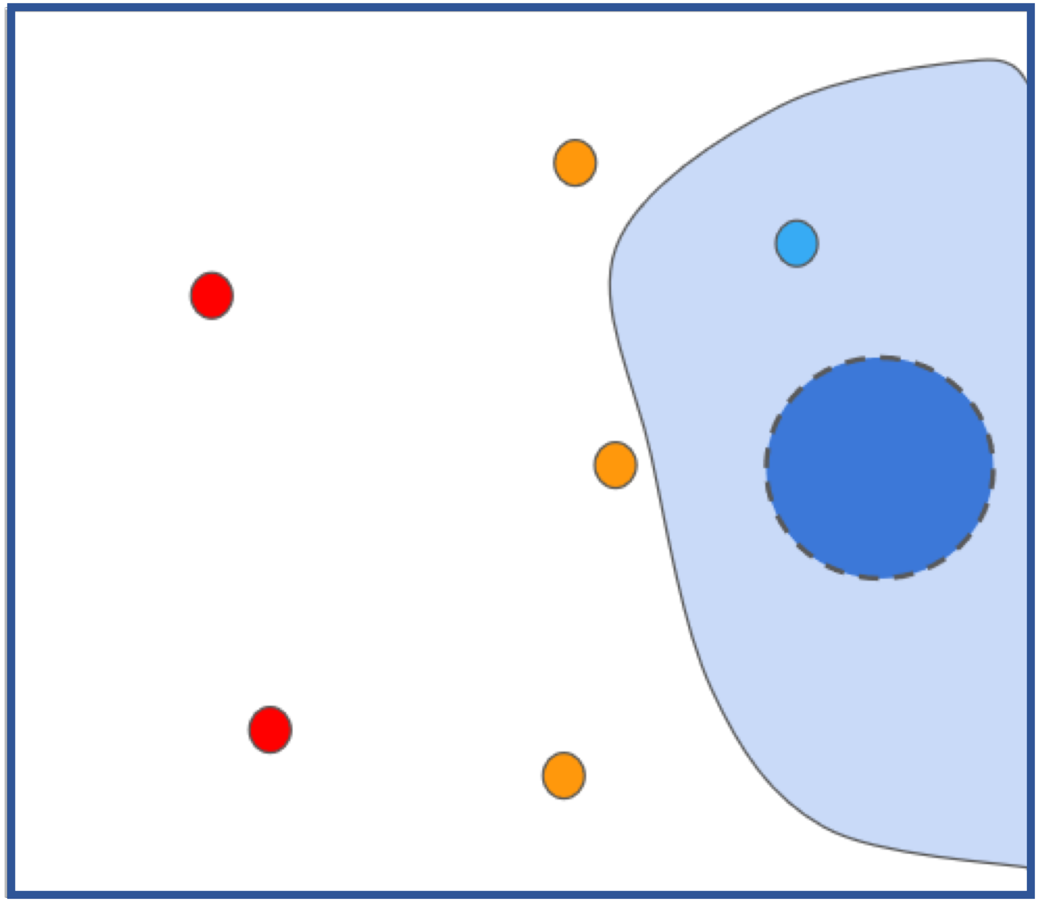


u_1

Sample a new point accordingly to:

$$P(\rho_K(\boldsymbol{u}) < 0) = CDF(\frac{0 - m_K(\boldsymbol{u})}{\sigma_K(\boldsymbol{u})})$$

Domain Estimation Algorithm (DEA)



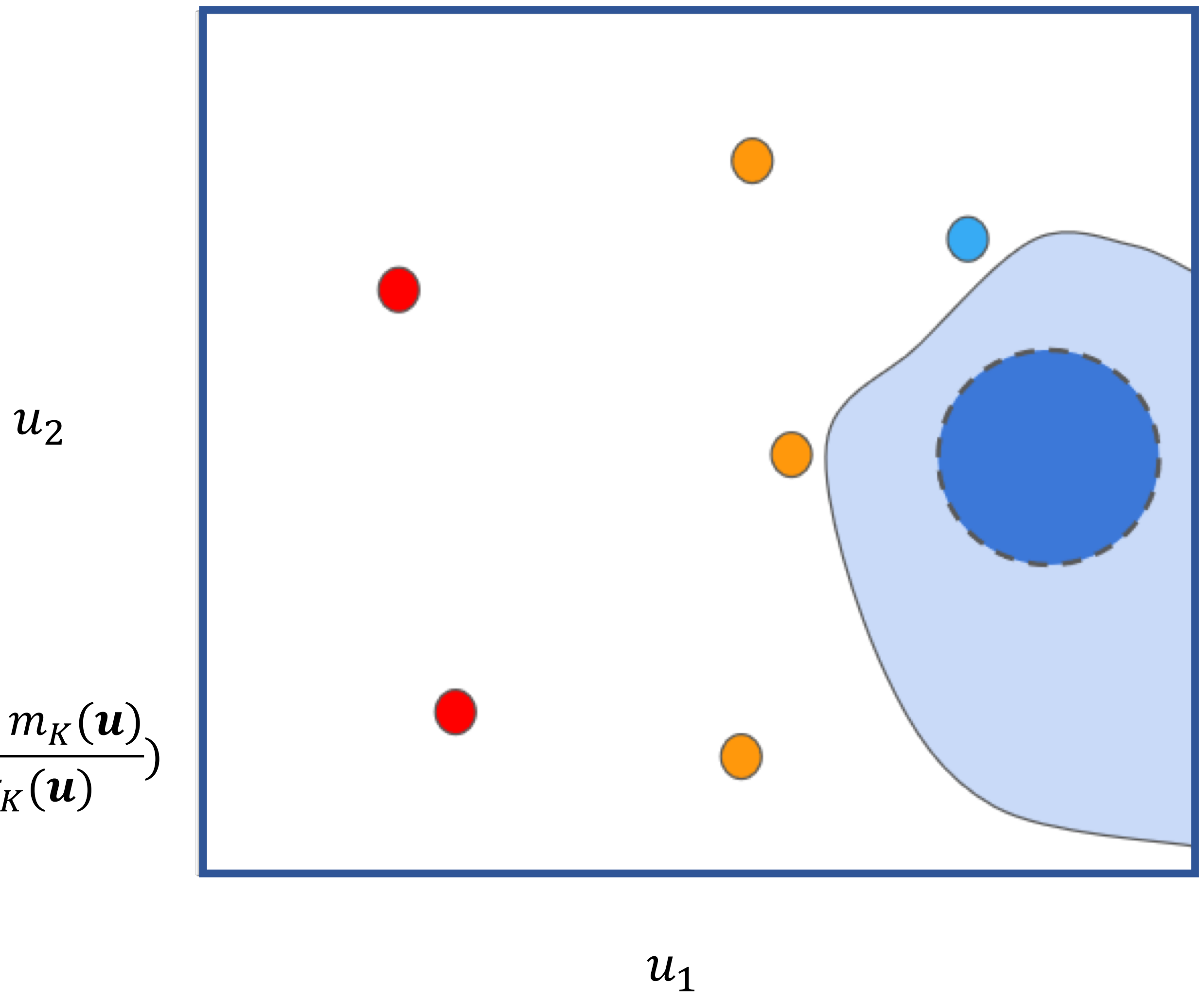
u_2

u_1

Sample a new point accordingly to:

$$P(\rho_K(\boldsymbol{u}) < 0) = CDF\left(\frac{0 - m_K(\boldsymbol{u})}{\sigma_K(\boldsymbol{u})}\right)$$

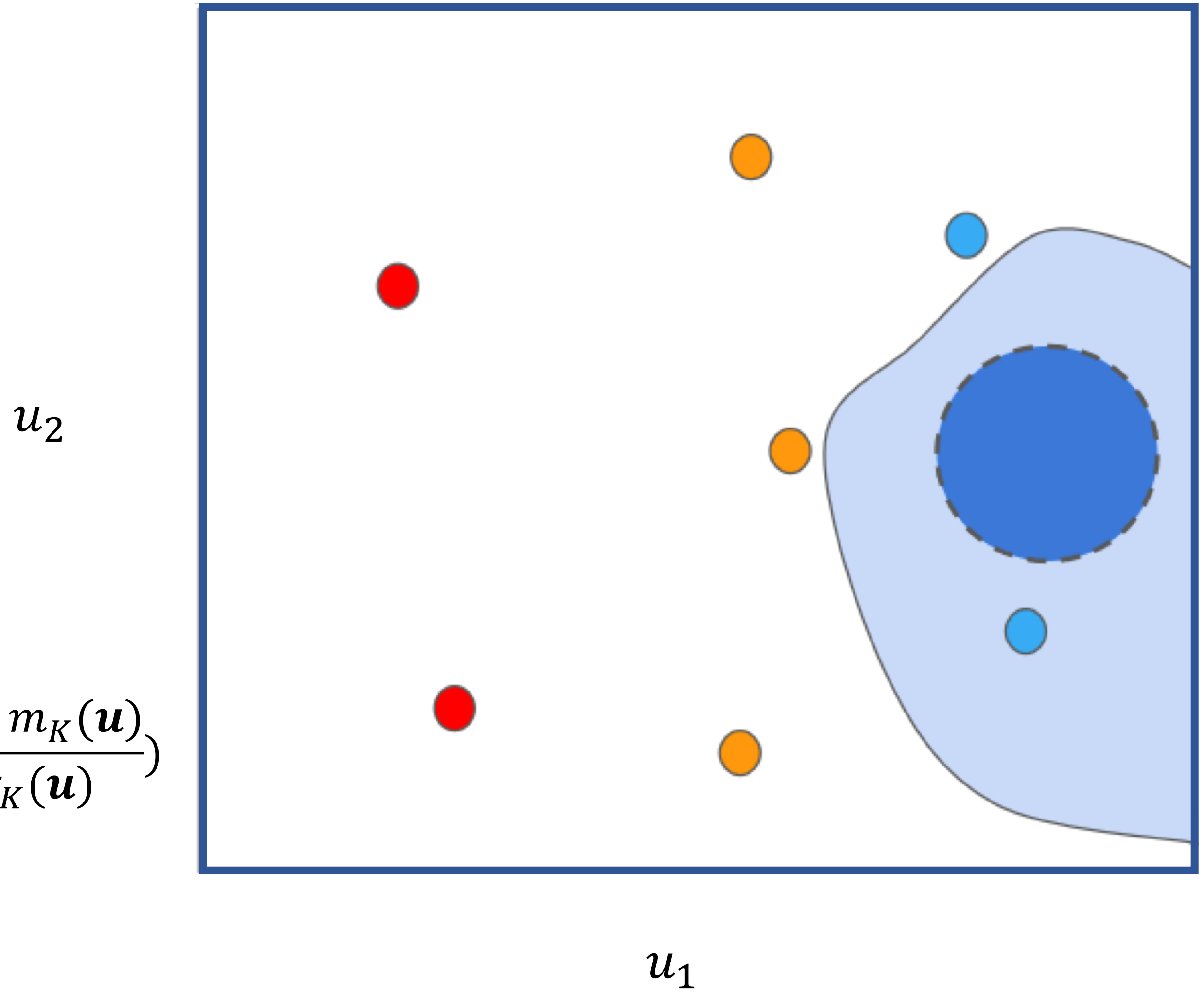
Domain Estimation Algorithm (DEA)



Sample a new point accordingly to:

$$P(\rho_K(\boldsymbol{u}) < 0) = CDF\left(\frac{0 - m_K(\boldsymbol{u})}{\sigma_K(\boldsymbol{u})}\right)$$

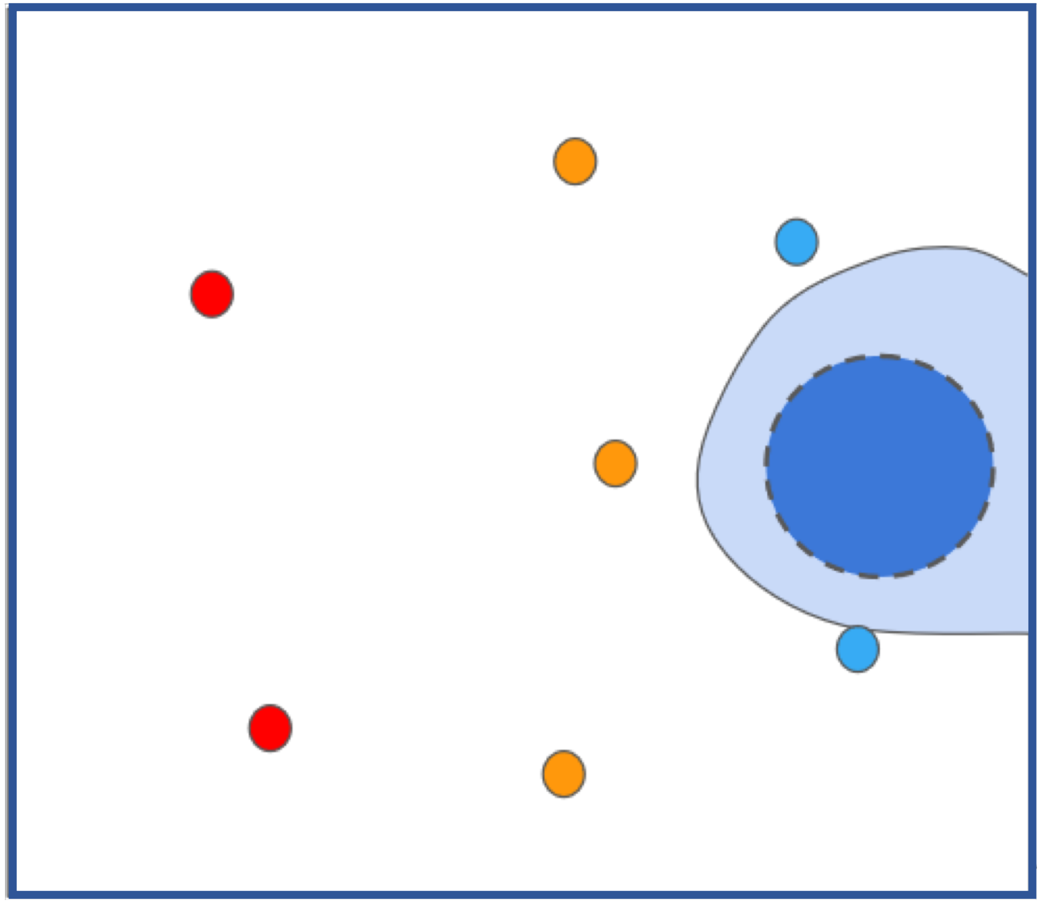
Domain Estimation Algorithm (DEA)



Sample a new point accordingly to:

$$P(\rho_K(\boldsymbol{u}) < 0) = CDF\left(\frac{0 - m_K(\boldsymbol{u})}{\sigma_K(\boldsymbol{u})}\right)$$

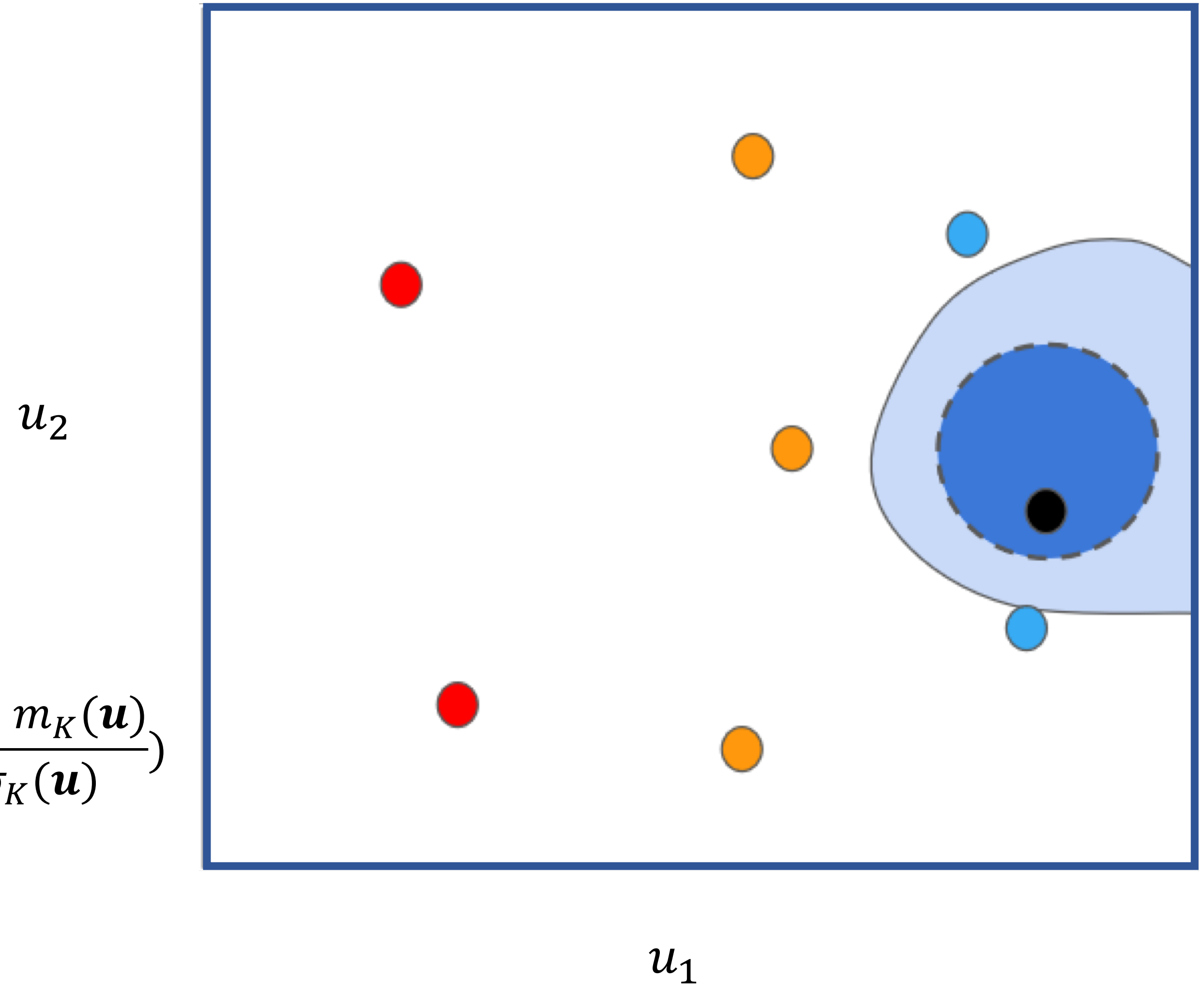
Domain Estimation Algorithm (DEA)



Sample a new point accordingly to:

$$P(\rho_K(\boldsymbol{u}) < 0) = CDF(\frac{0 - m_K(\boldsymbol{u})}{\sigma_K(\boldsymbol{u})})$$

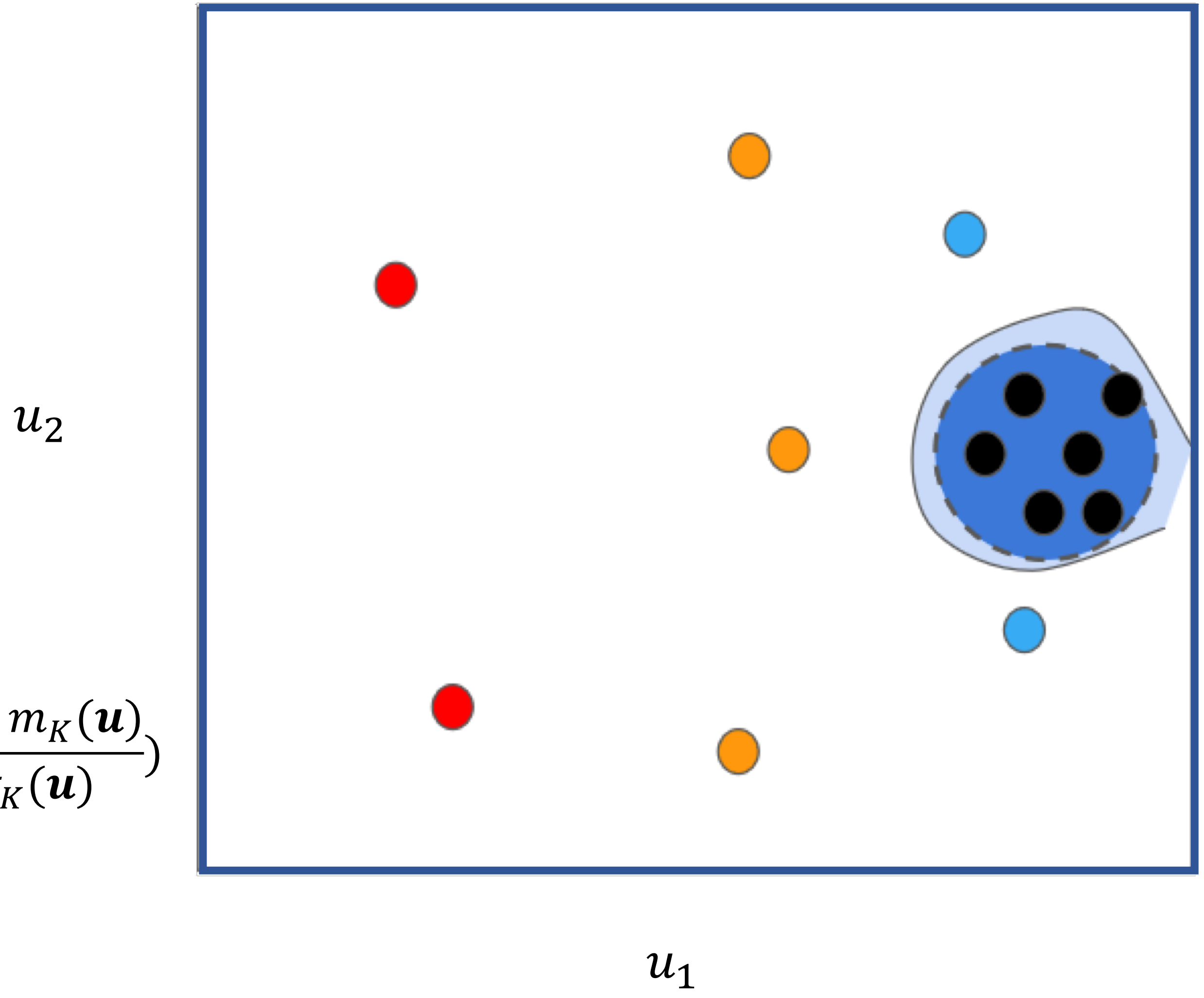
Domain Estimation Algorithm (DEA)



Sample a new point accordingly to:

$$P(\rho_K(\boldsymbol{u}) < 0) = CDF\left(\frac{0 - m_K(\boldsymbol{u})}{\sigma_K(\boldsymbol{u})}\right)$$

Domain Estimation Algorithm (DEA)

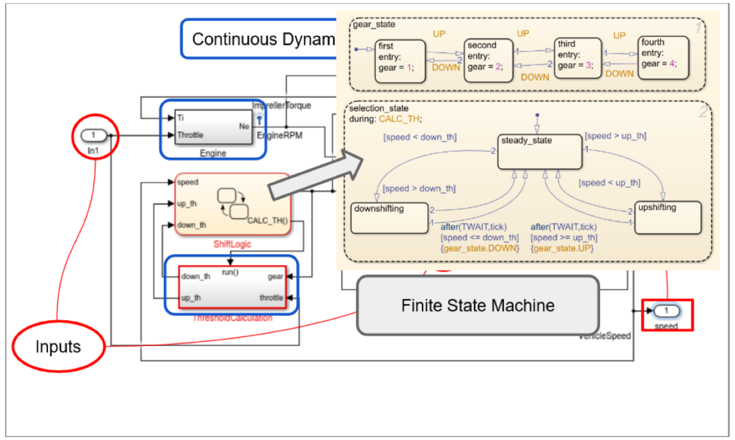


Sample a new point accordingly to:

$$P(\rho_K(\boldsymbol{u}) < 0) = CDF(\frac{0 - m_K(\boldsymbol{u})}{\sigma_K(\boldsymbol{u})})$$

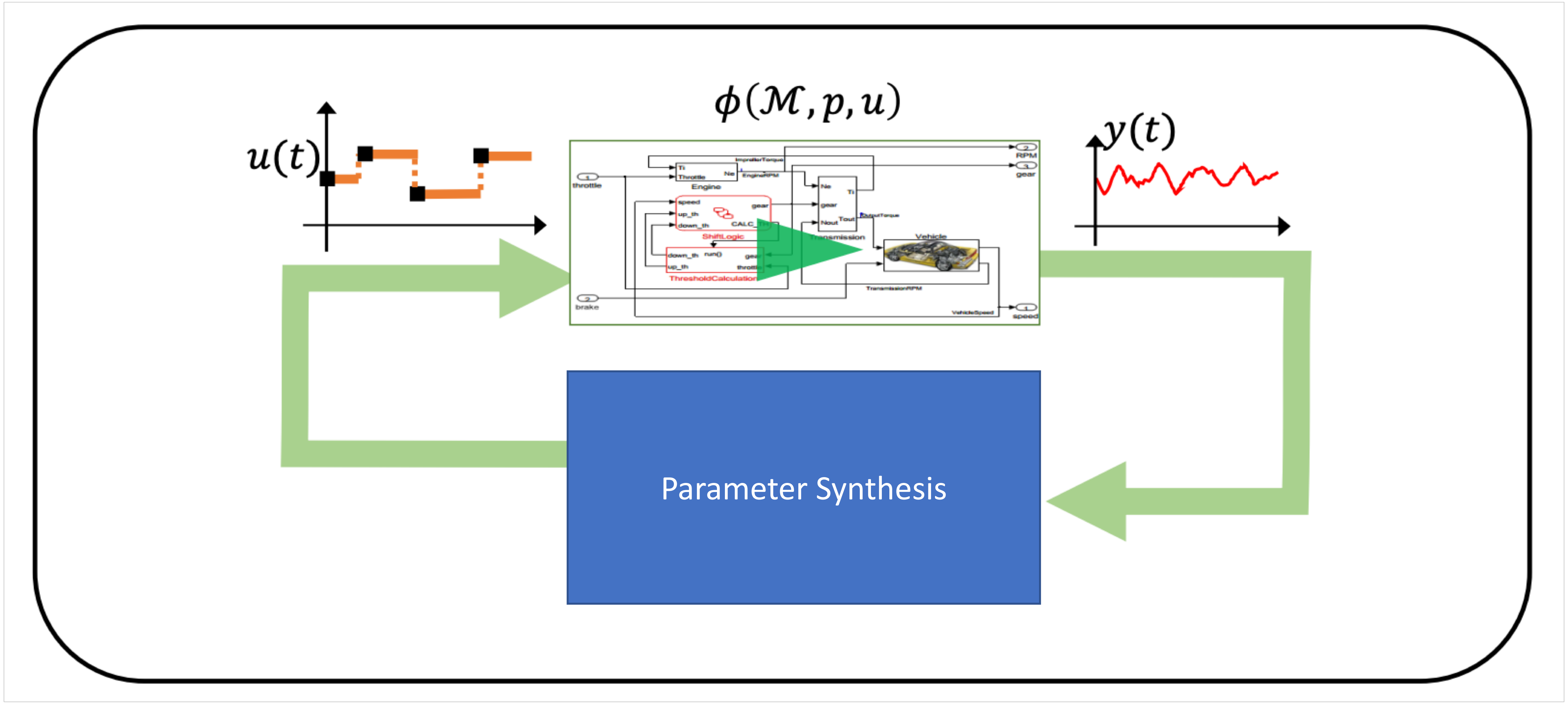
Tests Case & Results

- $\phi_1(\bar{v},\bar{\omega}) = \mathbf{G}_{[0,30]}(v \leq \bar{v} \wedge \omega \leq \bar{\omega})$ (in the next 30 seconds the engine and vehicle speed never reach $\bar{\omega}$ rpm and $\bar{v}$ km/h, respectively)
- $\phi_2(\bar{v},\bar{\omega}) = \mathbf{G}_{[0,30]}(\omega \leq \bar{\omega}) \rightarrow \mathbf{G}_{[0,10]}(v \leq \bar{v})$ (if the engine speed is always less than $\bar{\omega}$ rpm, then the vehicle speed can not exceed $\bar{v}$ km/h in less than 10 sec)
- $\phi_3(\bar{v},\bar{\omega}) = \mathbf{F}_{[0,10]}(v \geq \bar{v}) \rightarrow \mathbf{G}_{[0,30]}(\omega \leq \bar{\omega})$ (the vehicle speed is above $\bar{v}$ km/h than from that point on the engine speed is always less than $\bar{\omega}$ rpm)



	Adaptive DEA		Adaptive GP-UCB		S-TaLiRo		
Req	nval	times	nval	times	nval	times	Alg
ϕ_1	$\mathbf{4.42 \pm 0.53}$	2.16 ± 0.61	$\mathbf{4.16 \pm 2.40}$	0.55 ± 0.30	5.16 ± 4.32	0.57 ± 0.48	UR
ϕ_1	$\mathbf{6.90 \pm 2.22}$	5.78 ± 3.88	8.7 ± 1.78	1.52 ± 0.40	39.64 ± 44.49	4.46 ± 4.99	SA
ϕ_2	$\mathbf{3.24 \pm 1.98}$	1.57 ± 1.91	7.94 ± 3.90	1.55 ± 1.23	12.78 ± 11.27	1.46 ± 1.28	CE
ϕ_2	$\mathbf{10.14 \pm 2.95}$	12.39 ± 6.96	23.9 ± 7.39	9.86 ± 4.54	59 ± 42	6.83 ± 4.93	SA
ϕ_2	$\mathbf{8.52 \pm 2.90}$	9.13 ± 5.90	13.6 ± 3.48	4.12 ± 1.67	43.1 ± 39.23	4.89 ± 4.43	SA
ϕ_3	$\mathbf{5.02 \pm 0.97}$	2.91 ± 1.20	5.44 ± 3.14	0.91 ± 0.67	10.04 ± 7.30	1.15 ± 0.84	CE
ϕ_3	$\mathbf{7.70 \pm 2.36}$	7.07 ± 3.87	10.52 ± 1.76	2.43 ± 0.92	11 ± 9.10	1.25 ± 1.03	UR

Parameter Synthesis



Parameter Synthesis

Problem

Given a model, depending on a set of parameters $\theta \in \Theta$, and a specification ϕ (STL formula), find the parameter combination θ s.t. the system satisfies ϕ as more as possible

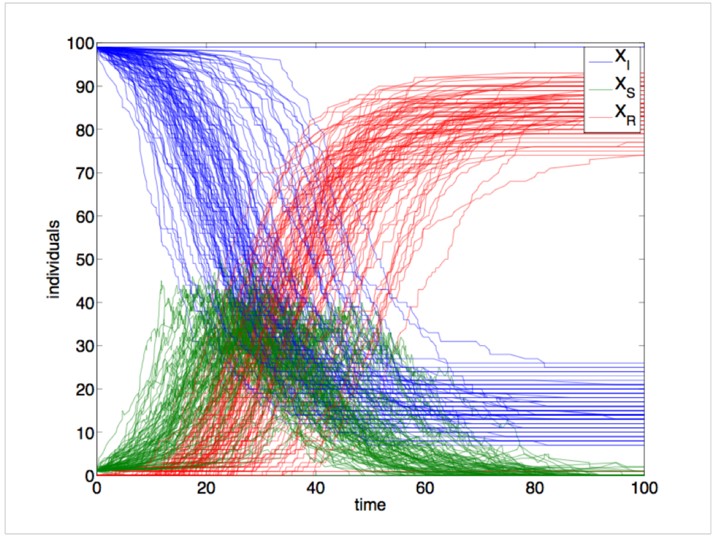
Solution Strategy

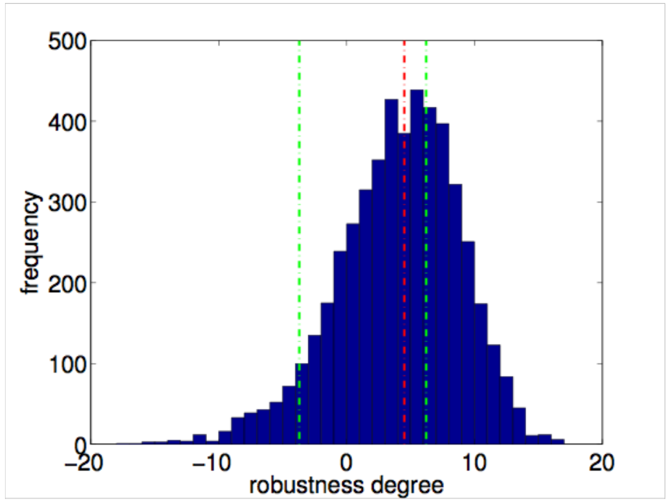
- **rephrase** it as a optimisation problem (maximizing ρ)
- **evaluate** the function to optimise
- **solve** the optimisation problem using

Parameter Synthesis via Robustness Maximisation

Robustness Distribution

$$\mathbb{P}\left(R_\varphi(\mathbf{X}) \in [a,b]\right) = \mathbb{P}\left(\mathbf{X} \in \{\mathbf{x} \in \mathcal{D} \mid \rho(\varphi, \mathbf{x}, 0) \in [a,b]\}\right)$$





Indicators

$\mathbb{E}(R_\varphi)$ (the average robustness degree)

$\mathbb{E}(R_\varphi \mid R_\varphi > 0)$ and $\mathbb{E}(R_\varphi \mid R_\varphi < 0)$ (the conditional averages)

Parameter Synthesis

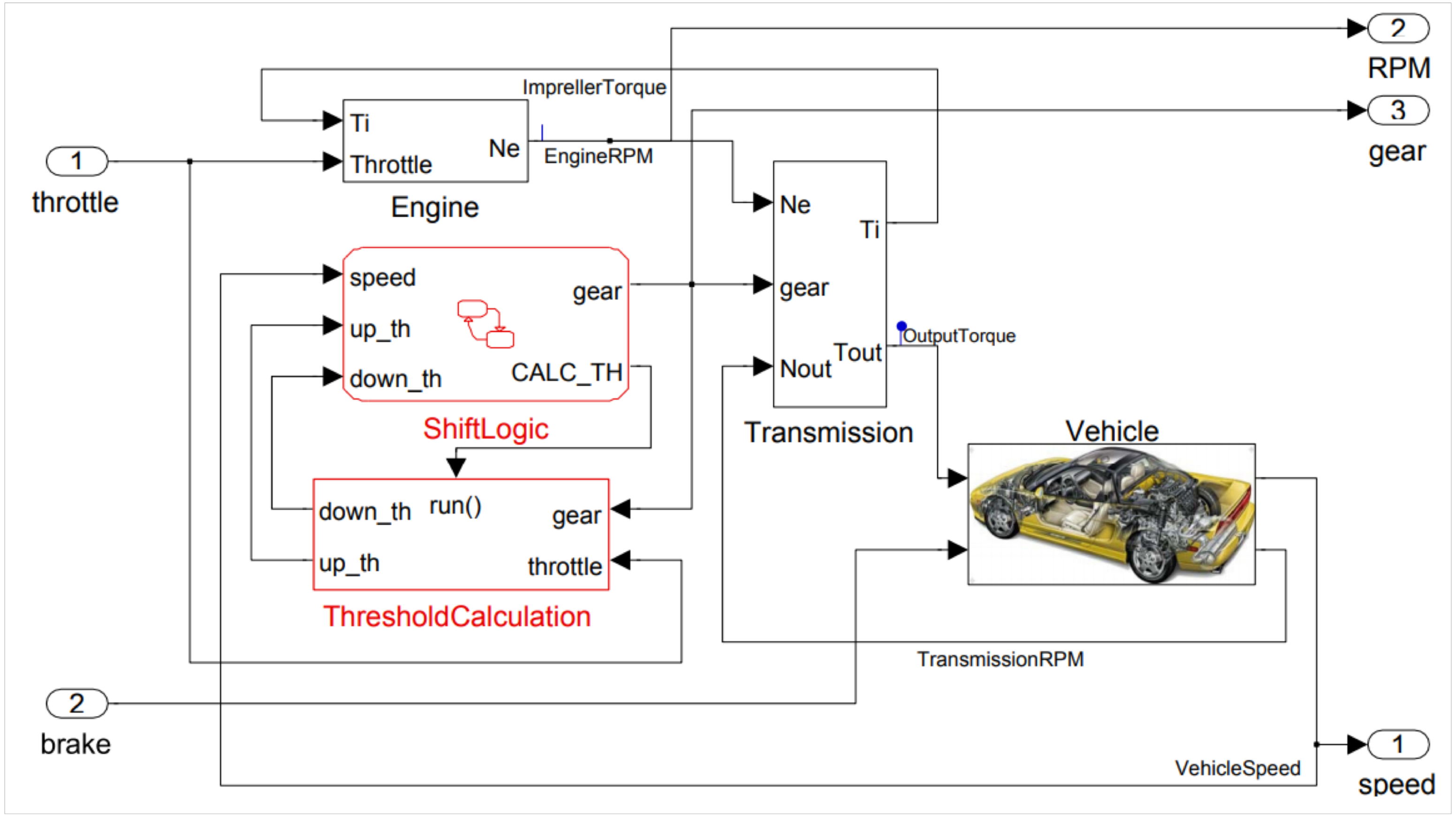
Problem

Find the parameter configuration that maximizes $E[R_\phi](\theta)$, of which we have few costly and noisy evaluations.

Methodology

1. Sample $\{(\theta_{(i)}, y_{(i)}), i = 1,\dots,n\}$
2. Emulate (**GP Regression**): $E[R_\phi] \sim GP(\mu,k)$
3. Optimize the emulation via **GP-UCB algorithm**, new $\theta_{(n+1)}$

Specification Mining



- What is the maximum speed that the vehicle can reach ?
- What is the minimum d well time in a given gear ?

Specification Mining

- Specification Mining: Try to find values of parameters of a PSTL formula from a given model

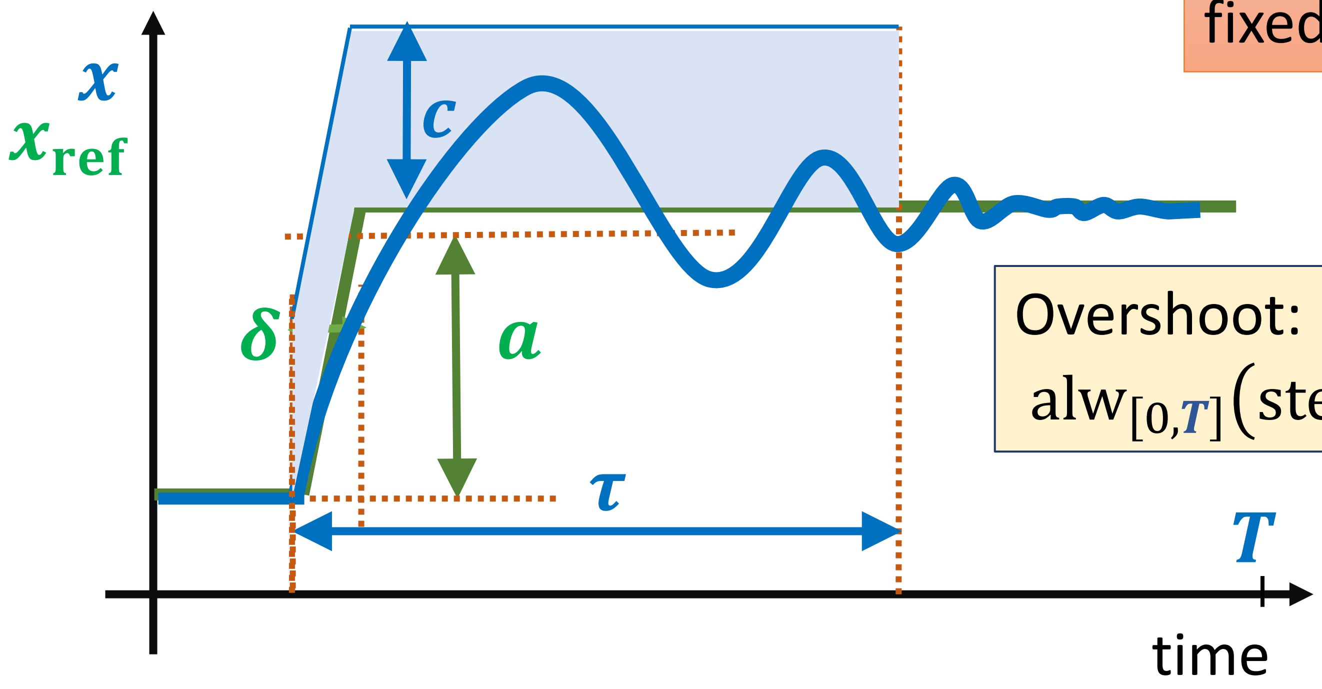
- Why?
  - Good to know “as-is” properties of the model
  - Finds worst-case behaviors of the model
  - Discriminates between regular and anomalous behaviours

Specification Templates using PSTL

- If we convert all values in an STL formula to parameters, then we get a specification template, or a Parametric STL (PSTL) formula

PSTL = Parametric Signal Temporal Logic

In previous lecture, a, c, T, τ, δ were some fixed values, here they represent parameters



Step:

$$\mathrm{step}(y) \coloneqq y(t+\delta) - y(t) > a$$

Overshoot:

$$\mathrm{alw}_{[0,T]}\left(\mathrm{step}(x_{\mathrm{ref}}) \Rightarrow \mathrm{alw}_{[0,\tau]}(x(t) - x_{\mathrm{ref}}(t) < c)\right)$$

Parametric Signal Temporal Logic

Definition (PSTL syntax)

$$\phi := (x_i \bowtie \pi) \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \, \mathcal{U}_{[\tau_1, \tau_2]} \, \varphi_2$$

with $\bowtie \in \{>, \leq\}$

- π is **threshold** parameter
- τ_1, τ_2 are **temporal** parameters

- $\mathbb{K} = (\mathcal{T} \times \mathcal{C})$ be the **parameter space**
- $\boldsymbol{\theta} \in \mathbb{K}$ is a **parameter configuration**

e.g., $\phi = \mathcal{F}_{[a,b]}(x_i > k), \boldsymbol{\theta} = (0, 2, 3.5)$ then $\phi_{\boldsymbol{\theta}} = \mathcal{F}_{[0,2]}(x_i > 3.5)$.

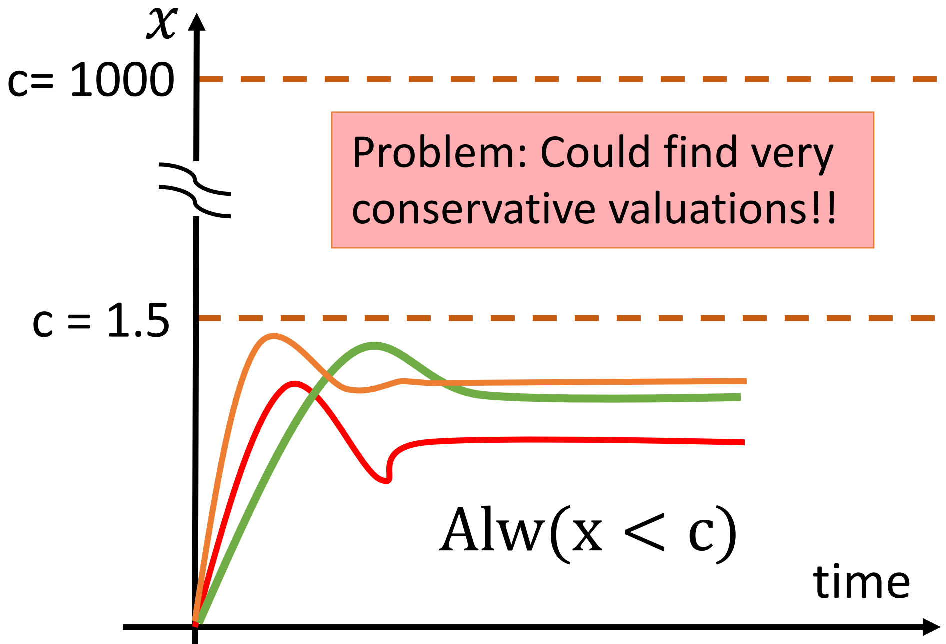
Parameter inference for PSTL

- Given:
  - PSTL formula $\varphi(\mathbf{p})$, $[\mathbf{p} = (p_1, p_2, \ldots, p_m)]$
  - Traces $x_1, \ldots, x_n$
- Find:
  - ~~Valuation~~ $\nu(\mathbf{p})$ such that: $\forall i : x_i \vDash \varphi\big(\nu(\mathbf{p})\big)$

    δ-tight valuation

  - and $\exists i: x_i \nvDash \varphi(\nu(\mathbf{p}) \pm \delta)$:

    i.e. small perturbation in $\nu(\mathbf{p})$ makes some trace not satisfy formula



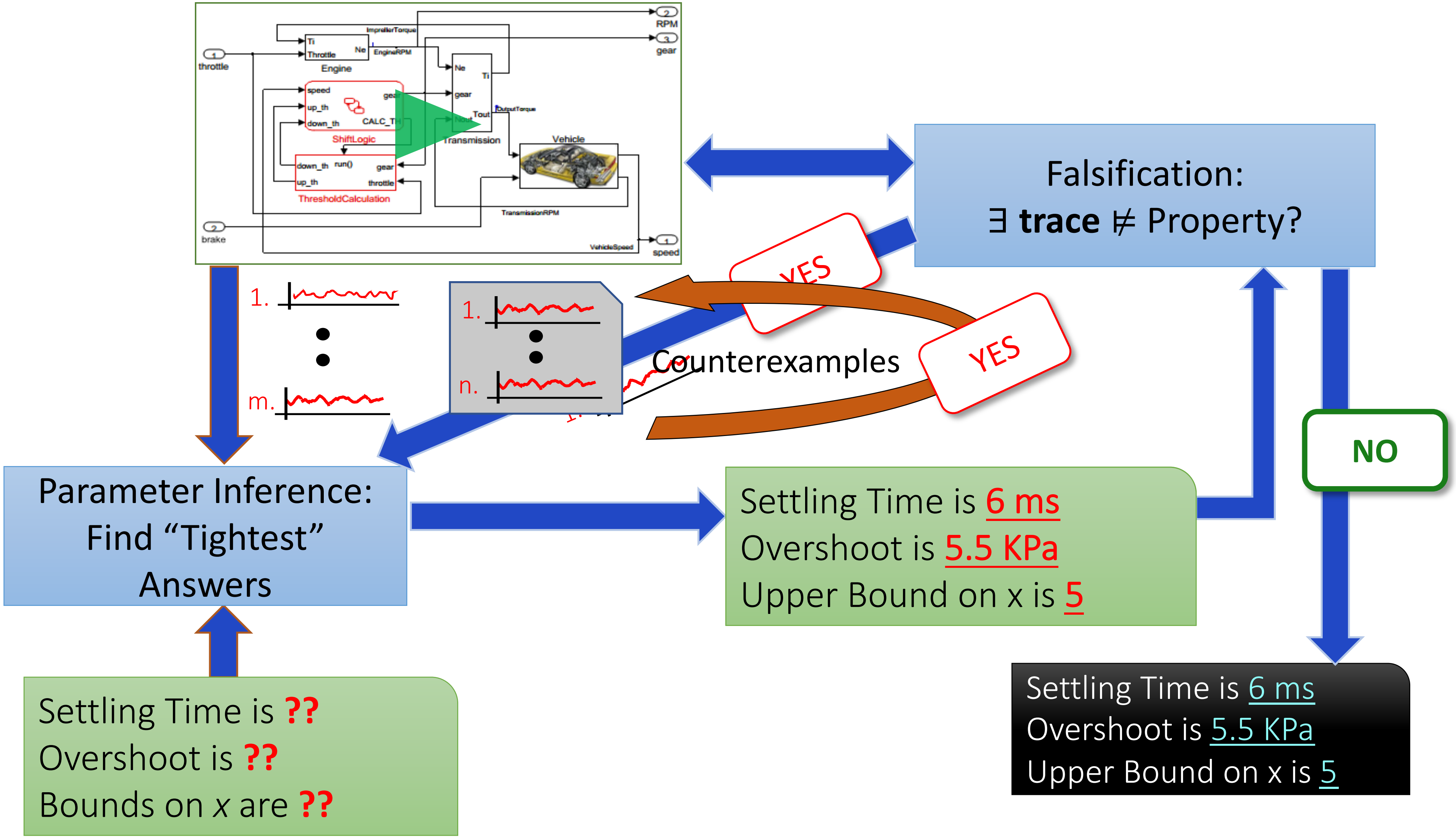
Problem: Could find very conservative valuations!!

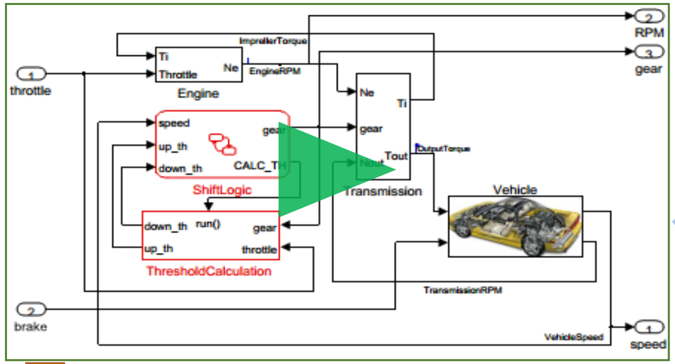
formula sat for given valuation ⇒ ∀ greater (or lesser) valuations sat

Finding δ-tight valuations hard in general, but **efficient** for **Monotonic PSTL**

Binary search on parameter space

Specification Mining





1.

1.

Specification Mining

Falsification:
∃ **trace** ⊭ Property?

YES

NO

Param…
Find "Tightest" Answers

Secret Sauce:

- Infer parameters for a given PSTL formula from traces
- Falsify given STL formula

Overshoot is 5.5 KPa
Upper Bound on x is 5

Settling Time is **??**
Overshoot is **??**
Bounds on *x* are **??**

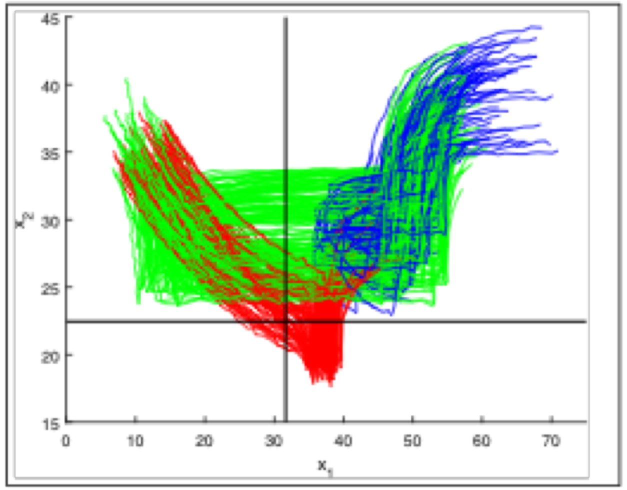
Settling Time is 6 ms
Overshoot is 5.5 KPa
Upper Bound on x is 5

Learning STL classifiers

Goal: Given sets of good and bad trajectories (or generative models), learn MTL properties that can separate the two behaviours (a MTL classifier)

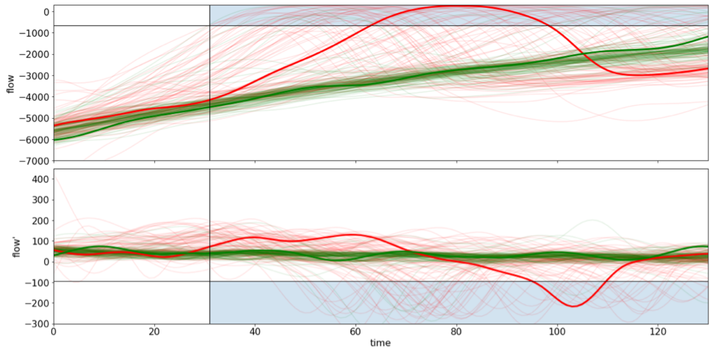
Idea: for a fixed template formula, learn optimally separating parameters by **Bayesian Optimisation**.

$$\varphi = ((x_2 > 22.46)\,\mathcal{U}_{[49,287]}\,(x_1 \le 31.65))$$



Maritime surveillance

$$\phi = \mathcal{F}_{[31,130]}((flow \ge -670) \vee (flow' \le -94))$$



Light entrainment of biological oscillator

Idea: explore formula structure by **genetic programming** on syntactic trees

Problem Formulation

A supervised two-class classification problem

Given a training set of D_p(good) and D_n(bad) find the best ϕ that better separates the two sets.

Discrimination Function

$$G(\phi) = \frac{\mathbb{E}(R_\phi|\vec{X}_p) - \mathbb{E}(R_\phi|\vec{X}_n)}{\sigma(R_\phi|\vec{X}_p) + \sigma(R_\phi|\vec{X}_n)}.$$

Observation: only statistical and noisy evaluations of $G(\phi)$

Goal: maximize $G(\phi)$

ROGE – RObustness GEnetic Algorithm

It is a bi-level optimization algorithm. A GEnetic algorithm to learn the structure and a Bayesian optimization algorithm to learn the parameters.

Require: $\mathcal{D}_p, \mathcal{D}_n, \mathbb{K}, Ne, Ng, \alpha, s$

$gen \leftarrow \textsc{GenerateInitialFormulae}(Ne, s)$

$gen_\Theta \leftarrow \textsc{LearningParameters}(gen, G, \mathbb{K})$

for $i = 1 \ldots Ng$ **do**

  $subg_\Theta \leftarrow \textsc{Sample}(gen_\Theta, F)$

  $newg \leftarrow \textsc{Evolve}(subg_\Theta, \alpha)$

  $newg_\Theta \leftarrow \textsc{LearningParameters}(newg, G, \mathbb{K})$

  $gen_\Theta \leftarrow \textsc{Sample}(newg_\Theta \cup gen_\Theta, F)$

end for

return gen_Θ

$$\phi_{best} = \mathrm{argmax}_{\phi_\theta \in gen_\Theta}(G(\phi_\theta))$$

Learning the Parameters

Problem

Given a PSTL formula ϕ, a parameter space $\mathbb{K}$, find Θ^* that maximises the discrimination function $G(\phi_\Theta)$.

Methodology

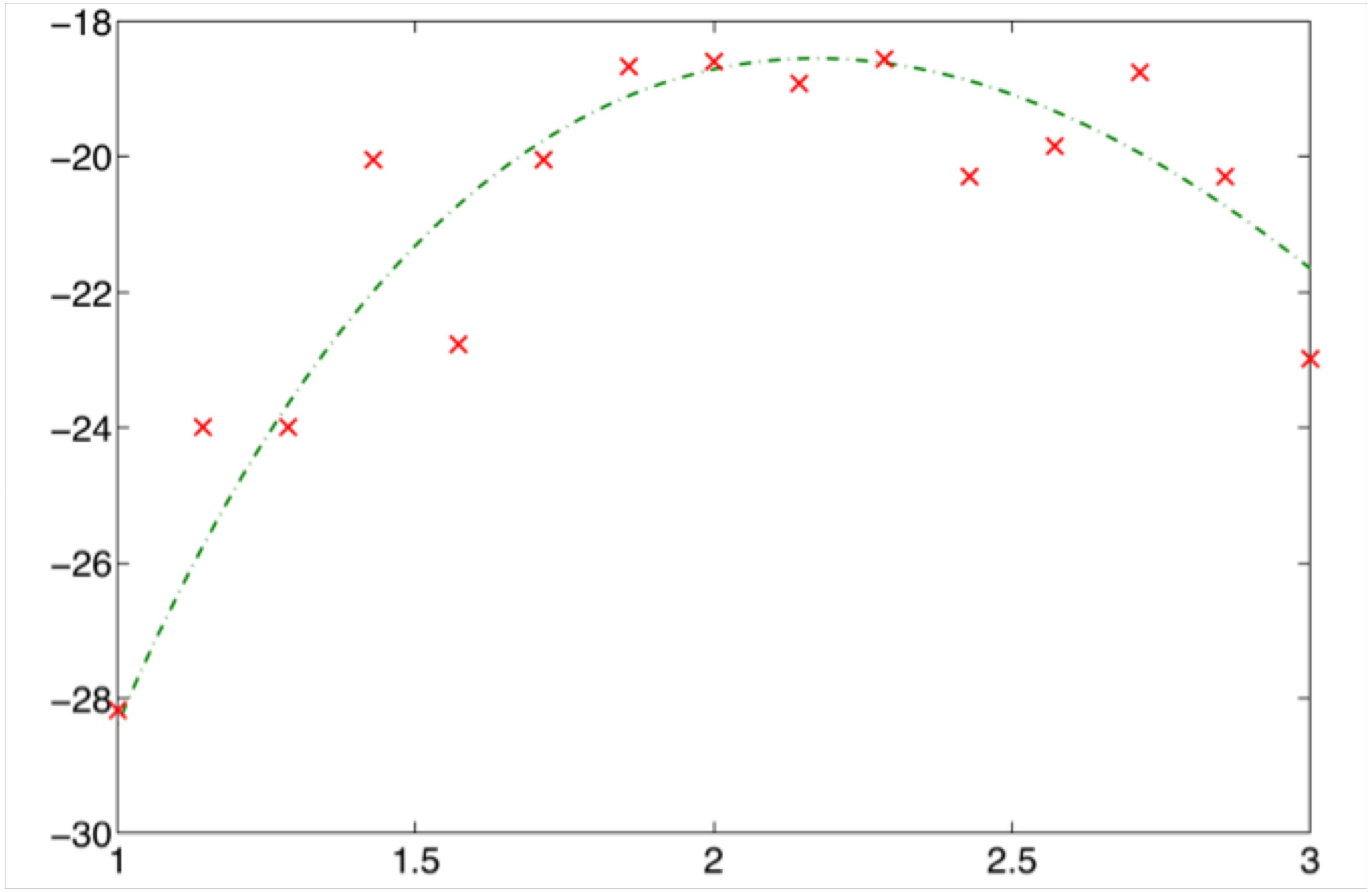
1. Sample $\{(\theta_{(i)}, y_{(i)}), i = 1,\ldots,n\}$
2. Emulate (**GP Regression**): $G[R_\phi] \sim GP(\mu,k)$
3. Optimize the emulation via **GP-UCB algorithm**, new $\theta_{(n+1)}$

$\exists\delta$ s.t. $\mathbb{E}(R_{\phi_{\Theta^*}}|\vec{X}_p) > \delta$ and $\mathbb{E}(R_{\phi_{\Theta^*}}|\vec{X}_n) \leq \delta$

Translation. $(\vec{x} - \delta) \Rightarrow \mathbb{E}(R_{\phi_{\Theta^*}}|\vec{X}_p) > 0$ and $\mathbb{E}(R_{\phi_{\Theta^*}}|\vec{X}_n) \leq 0$

(1) The G($\phi_\circ$) Computation

Collection of the training set $\{(\theta^{(i)}, y^{(i)}), i = 1,\ldots,m\}$ for parameters values θ.

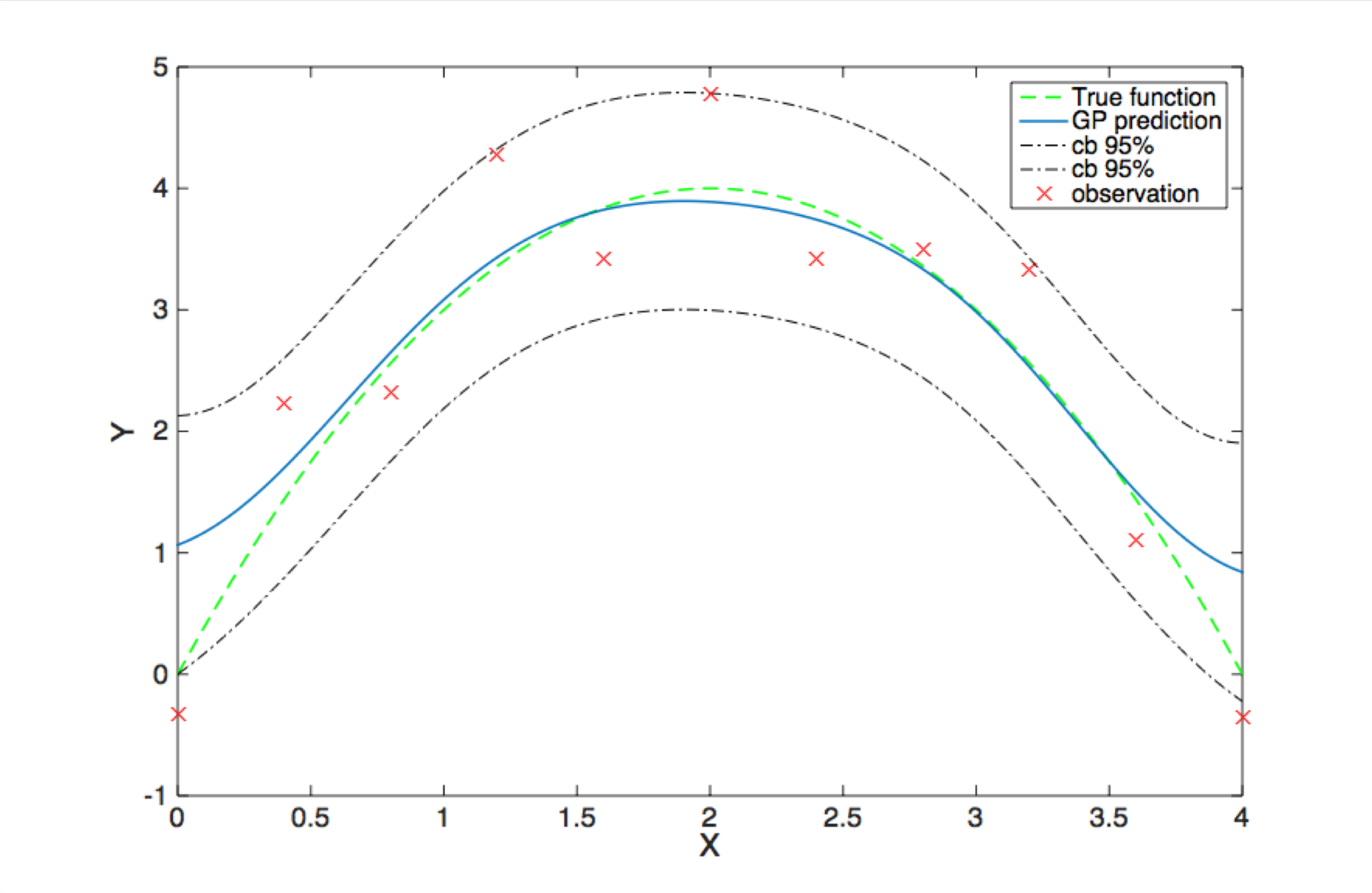


(2) The GP Regression

We have noisy observations y of the function value distributed around an unknown true value f (θ) with spherical Gaussian noise

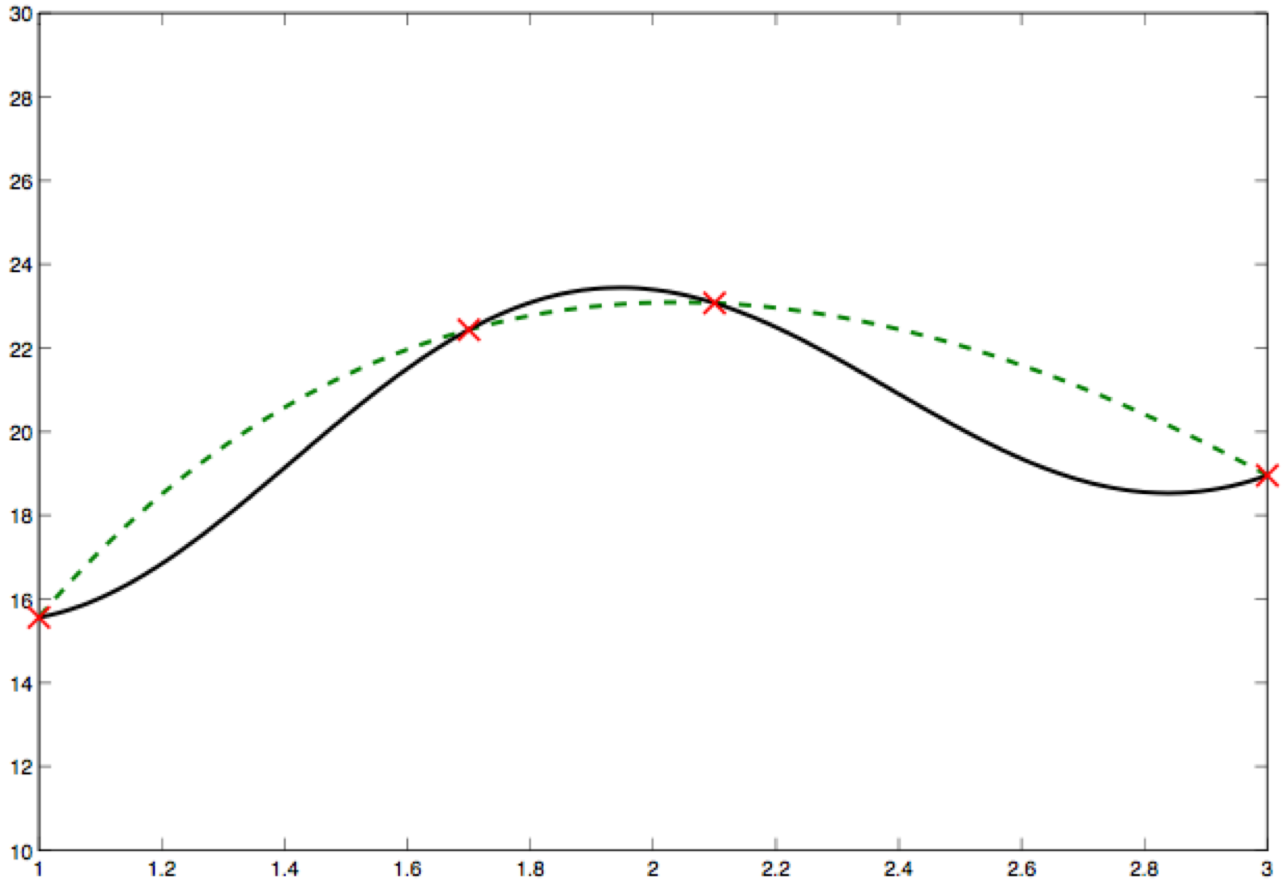
(2) The GP Regression

We have noisy observations y of the function value distributed around an unknown true value f (θ) with spherical Gaussian noise



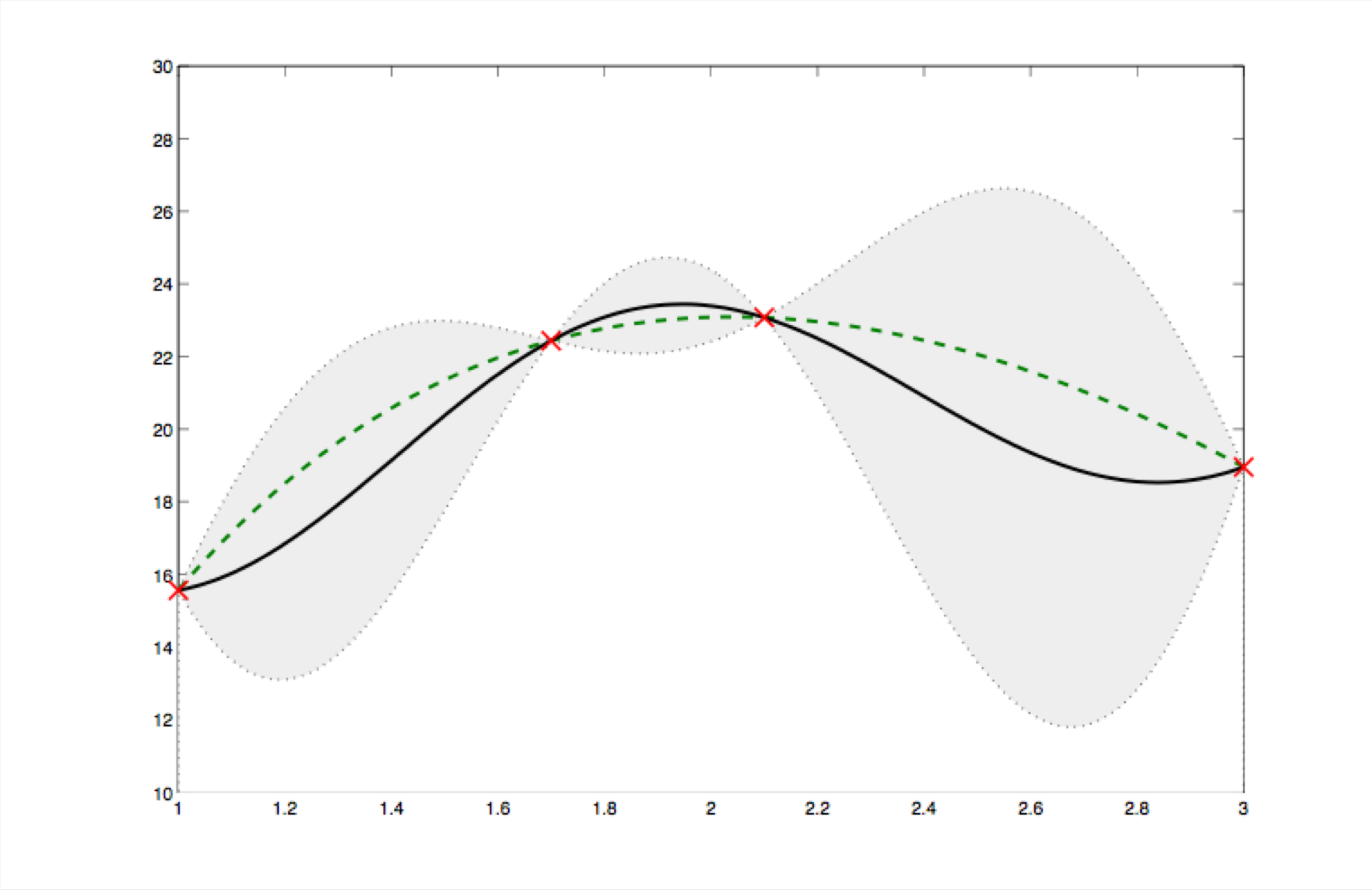
(3) The GP-UCB Algorithm

Balance Exploration and Exploitation: we maximise the **95% upper quantile of the distribution**: $\theta_{t+1} = argmax_{\theta}[\mu^*(\theta) + \beta_t \sqrt{k^*(\theta,\theta)}]$



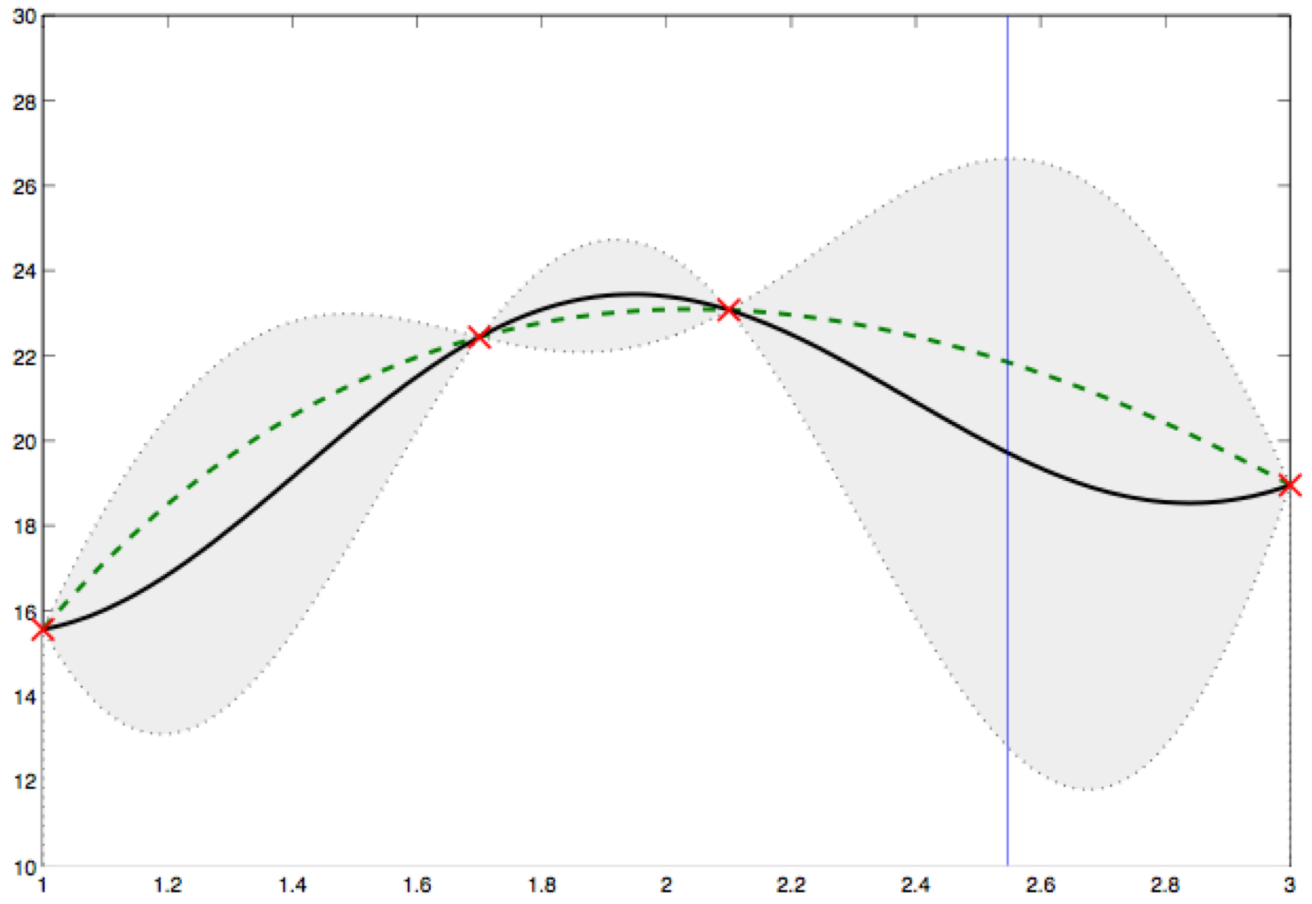
(3) The GP-UCB Algorithm

Balance Exploration and Exploitation: we maximise the **95% upper quantile of the distribution**: $\theta_{t+1} = argmax_{\theta}[\mu^*(\theta) + \beta_t\sqrt{k^*(\theta,\theta)}]$



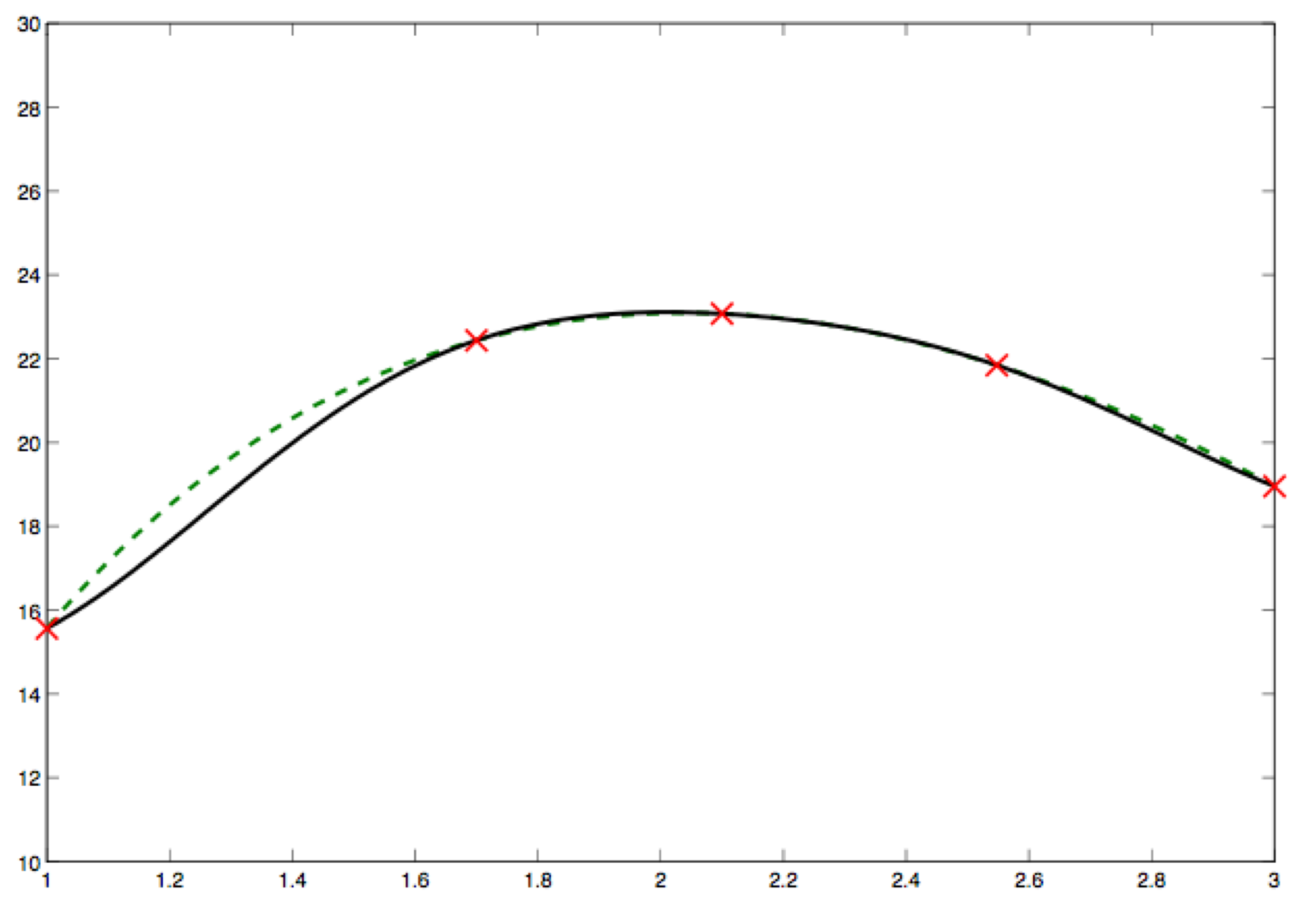
(3) The GP-UCB Algorithm

Balance Exploration and Exploitation: we maximise the **95% upper quantile of the distribution**: $\theta_{t+1} = argmax_{\theta}[\mu^*(\theta) + \beta_t\sqrt{k^*(\theta,\theta)}]$



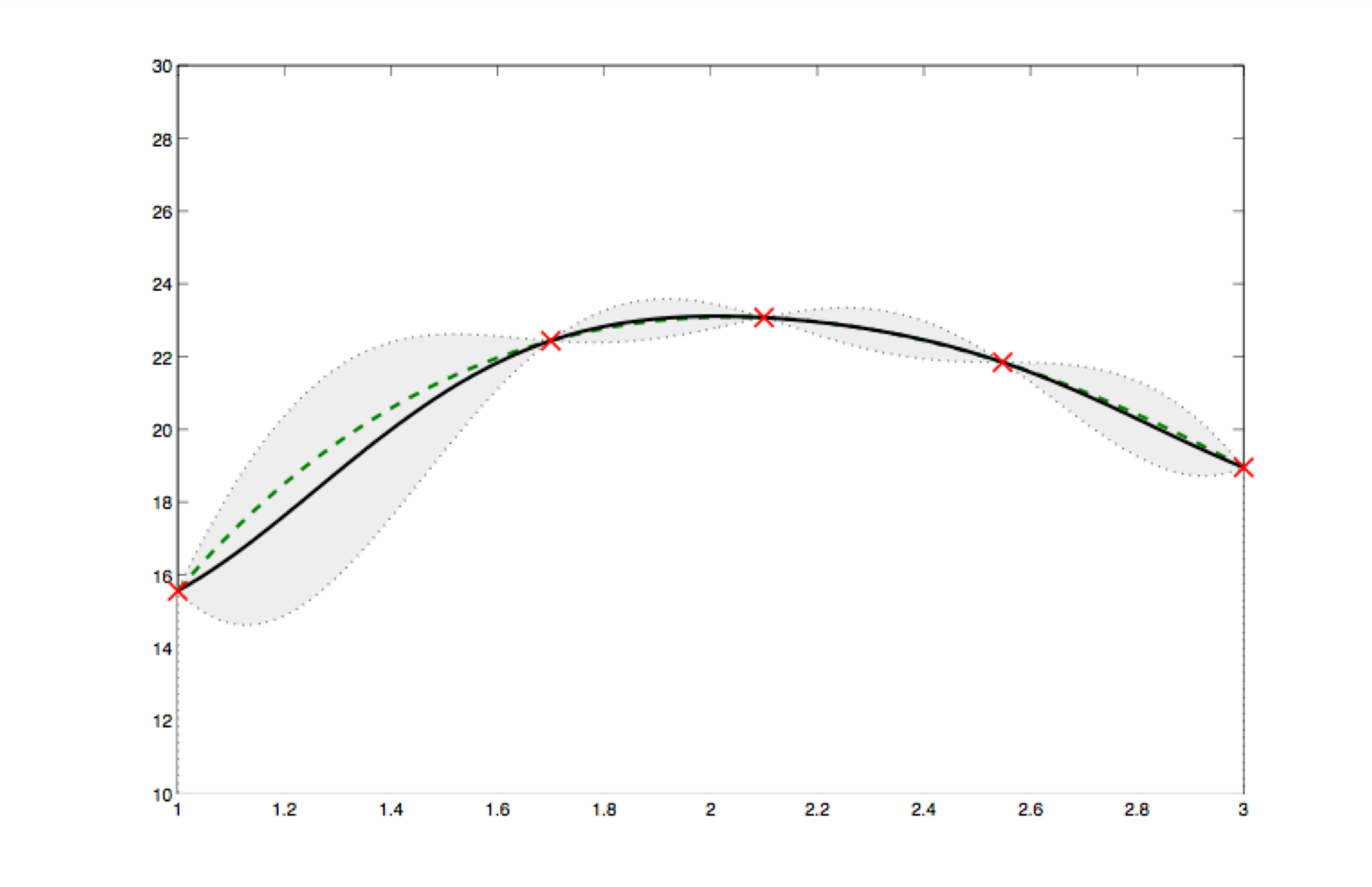
(3) The GP-UCB Algorithm

Balance Exploration and Exploitation: we maximise the **95% upper quantile of the distribution**: $\theta_{t+1} = argmax_{\theta}[\mu^*(\theta) + \beta_t\sqrt{k^*(\theta,\theta)}]$



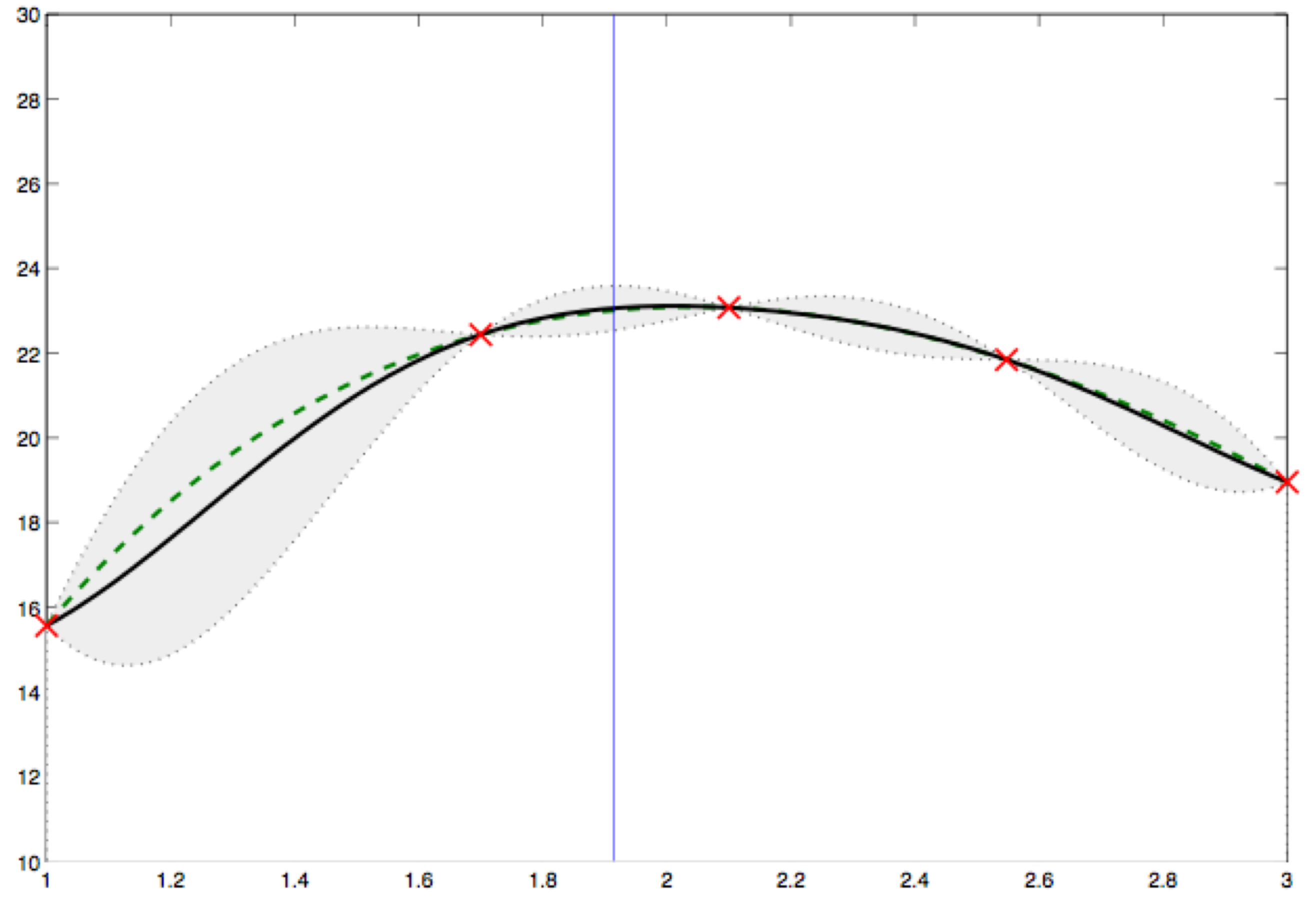
(3) The GP-UCB Algorithm

Balance Exploration and Exploitation: we maximise the **95% upper quantile of the distribution**: $\theta_{t+1} = argmax_{\theta}[\mu^*(\theta) + \beta_t\sqrt{k^*(\theta,\theta)}]$



(3) The GP-UCB Algorithm

Balance Exploration and Exploitation: we maximise the **95% upper quantile of the distribution**: $\theta_{t+1} = argmax_\theta[\mu^*(\theta) + \beta_t\sqrt{k^*(\theta,\theta)}]$



Learning the Structure

Problem

Given a set of PSTL formulas *gen*, find the best ϕ such that ϕ_Θ maximises the discrimination function $G(\phi_\Theta)$.

Methodology

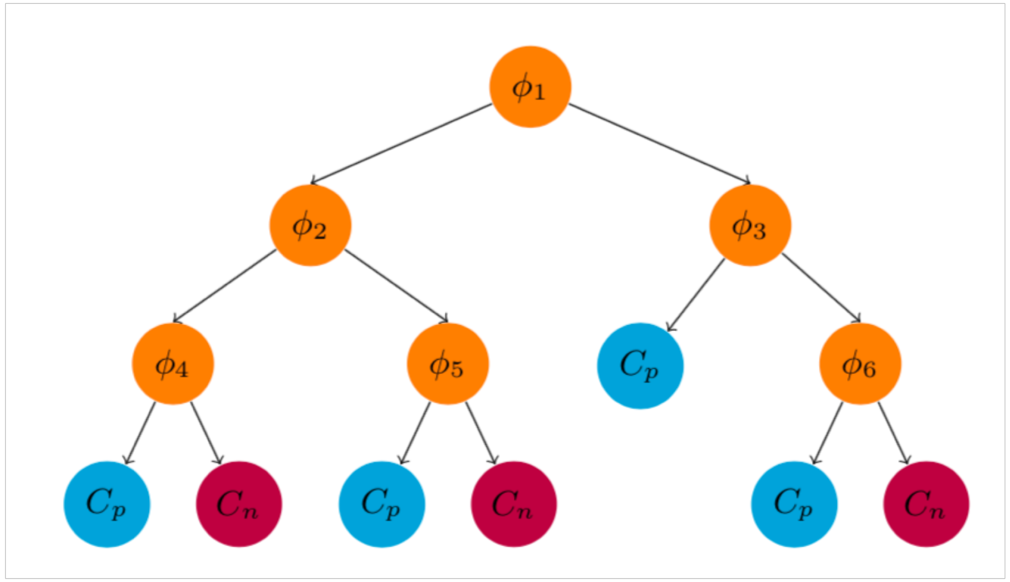
1. **GENERATEINITIALFORMULAE**: $gen=\{\phi_1,\cdots,\phi_{N_e}\}$
2. **SAMPLE**$(gen_\Theta, F)=subg_\Theta$, $N_e/2$ formulae, $F(\phi)=G(\phi)-S(\phi)$
3. **EVOLVE**$(subg_\Theta, \alpha) = newg_\Theta$, based on two genetic operators, a **recombination** and a **mutation** operator.

Regularization

Formula size penalty $S(\phi)$ and complexity of initial population.

Comparison with

DTL4STL [2], that uses a decision tree algorithm for the structure of the formula and an heuristic impurity measure for parameter synthesis

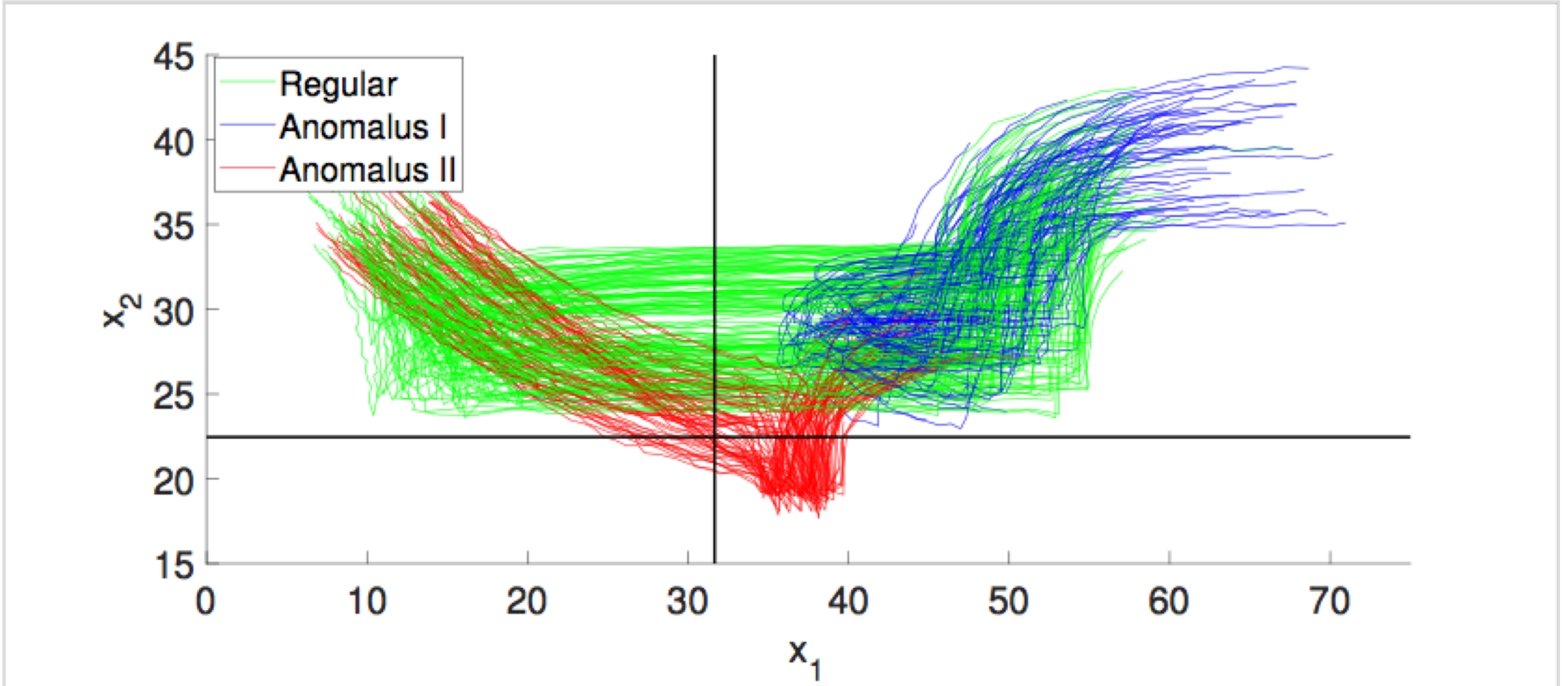


$$\phi_{Tree} = (\phi_1 \wedge ((\phi_2 \wedge \phi_4) \vee (\neg\phi_2 \wedge \phi_5))) \vee (\neg\phi_1 \wedge (\phi_3 \vee (\neg\phi_3 \wedge \phi_6)))$$

[2] Bombara, G et all, A Decision Tree Approach to Data Classification Using Signal Temporal Logic. In: Proc. of HSCC, 2016.

Maritime Surveillance

Synthetic dataset of naval surveillance of 2-dimensional coordinates traces of vessels behaviours.



$$\phi_{ROGE} = ((x_2 > 22.46)\, \mathcal{U}_{[49,287]}\, (x_1 \leq 31.65))$$

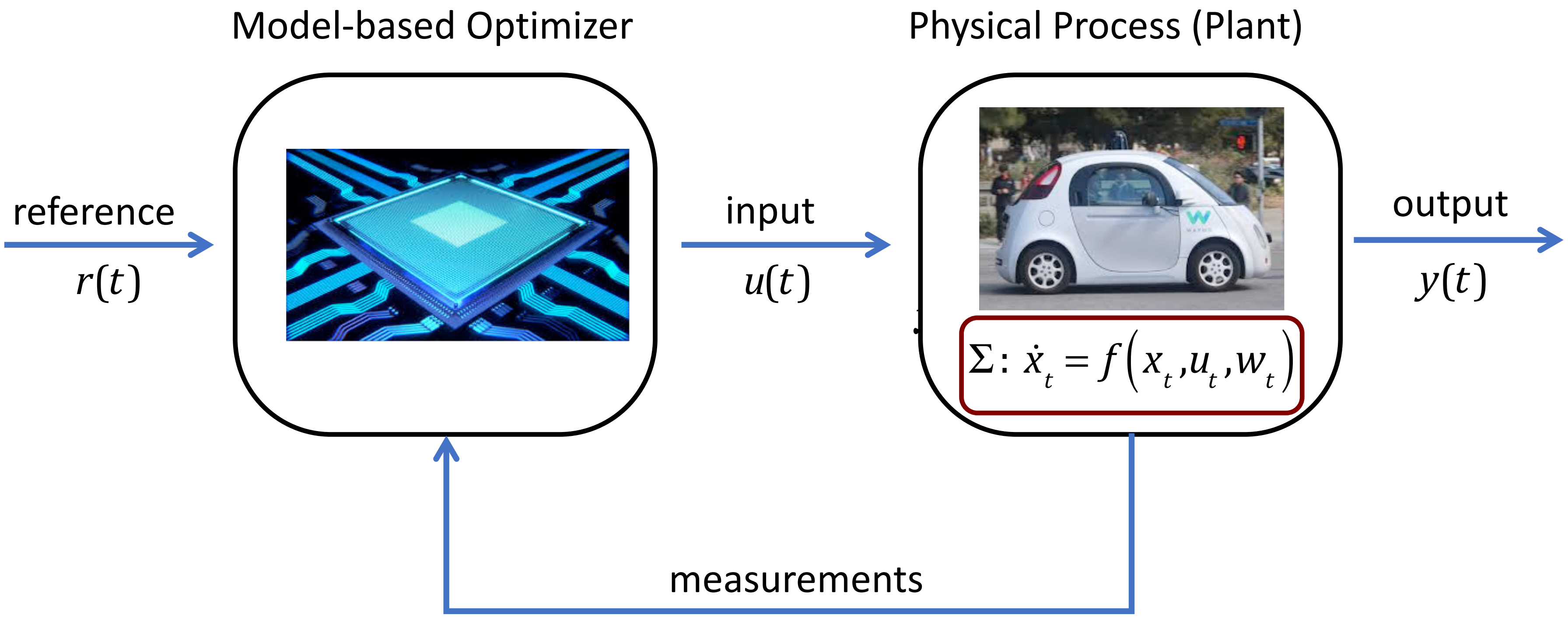
Maritime Surveillance

$$\phi_{ROGE} = ((x_2 > 22.46)\,\mathcal{U}_{[49,287]}\,(x_1 \leq 31.65))$$

$$\begin{aligned}\psi_{DTL4STL} = (((\mathcal{G}_{[187,196)}x_1 < 19.8) \wedge (\mathcal{F}_{[55.3,298)}x_1 > 40.8)) \vee ((\mathcal{F}_{[187,196)}x_1 > 19.8) \wedge \\ ((\mathcal{G}_{[94.9,296)}x_2 < 32.2) \vee ((\mathcal{F}_{[94.9,296)}x_2 > 32.2) \wedge (((\mathcal{G}_{[50.2,274)}x_2 > 29.6) \wedge \\ (\mathcal{G}_{[125,222)}x_1 < 47)) \vee ((\mathcal{F}_{[50.2,274)}x_2 < 29.6) \wedge (\mathcal{G}_{[206,233)}x_1 < 16.7))))))\end{aligned}$$

	ROGE	DTL4STL	DTL4STL$_p$
Mis. Rate	0	0.01 ± 0.013	0.007 ± 0.008
Comp. Time (sec.)	73 ± 18	144 ± 24	-

Control Synthesis with STL



The idea is to use the dynamical model of the process to predict its future evolution and optimize consequently the control input signal

Bibliography

- **IFM17**] Silvetti S., Policriti A., Bortolussi L. (2017) *An Active Learning Approach to the Falsification of Black Box Cyber-Physical Systems.* IFM 2017. LNCS, vol 10510. Springer, Cham.
- [**TACAS18**] Bortolussi L., Silvetti S. (2018*) Bayesian Statistical Parameter Synthesis for Linear Temporal Properties of Stochastic Models.* TACAS 2018. LNCS, vol 10806. Springer, Cham
- [**QEST18**] Nenzi L., Silvetti S., Bartocci E., Bortolussi L. (2018) *A Robust Genetic Algorithm for Learning Temporal Specifications from Data.* QEST 2018. LNCS, vol 11024. Springer, Cham.

1. Several excellent papers on the first development of falsification technology can be found on the web-site of S-TaLiRo : https://sites.google.com/a/asu.edu/s-taliro/references
2. Breach : https://link.springer.com/chapter/10.1007/978-3-642-14295-6_17
3. Jyotirmoy Deshmukh, Marko Horvat, Xiaoqing Jin, Rupak Majumdar, and Vinayak S. Prabhu. 2017. Testing Cyber-Physical Systems through Bayesian Optimization. *ACM Trans. Embed. Comput. Syst.* 16, 5s, Article 170 (September 2017)
4. Deshmukh, Jyotirmoy, Xiaoqing Jin, James Kapinski, and Oded Maler. Stochastic Local Search for Falsification of Hybrid Systems. In *International Symposium on Automated Technology for Verification and Analysis*, pp. 500-517.
5. Jin, Deshmukh et al. Mining Requirements from Closed-loop Control Models (HSCC '13, IEEE Trans. On Computer Aided Design '15)