Cyber-Physical Systems

Laura Nenzi

Università degli Studi di Trieste
I Semestre 2024

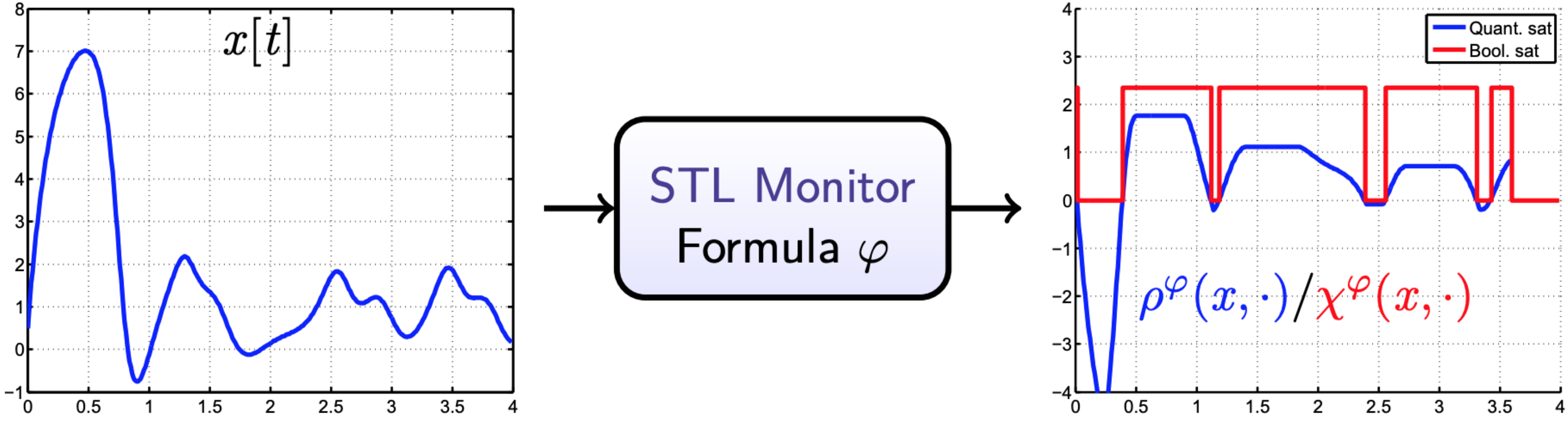
Lecture 14-15: STL applications: testing and falsification, parameter synthesis

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

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**: $M \models \phi \iff \forall \mathbf{x} \in trace(M) \quad s(\varphi, \mathbf{x}, 0) = 1$
- **Monitoring**: computing s for a single trace $\mathbf{x} \in trace(M)$
- **Statistical Model Checking**: "doing statistics" on s$(\varphi, \mathbf{x}, 0)$ for a finite-subset of $trace(M)$

STL Monitor



An STL monitor is a transducer that transforms x into Boolean or a quantitative signal

Parametric Chemical Reaction Network (PCRN)

Population CTMC models, i.e. CTMC models in the biochemical reactions style.

SET OF SPECIES

$\mathbb{S} = \{S_1, \ldots, S_n\}$, i.e. the different agent states.

STATE SPACE

The state space is described by a vector of n variables

$$\mathbf{X} = (X_{S_1}, \ldots, X_{S_n}) \in \mathbb{N},$$

each counting the number of agents (jobs, molecules, ...) of a given kind.

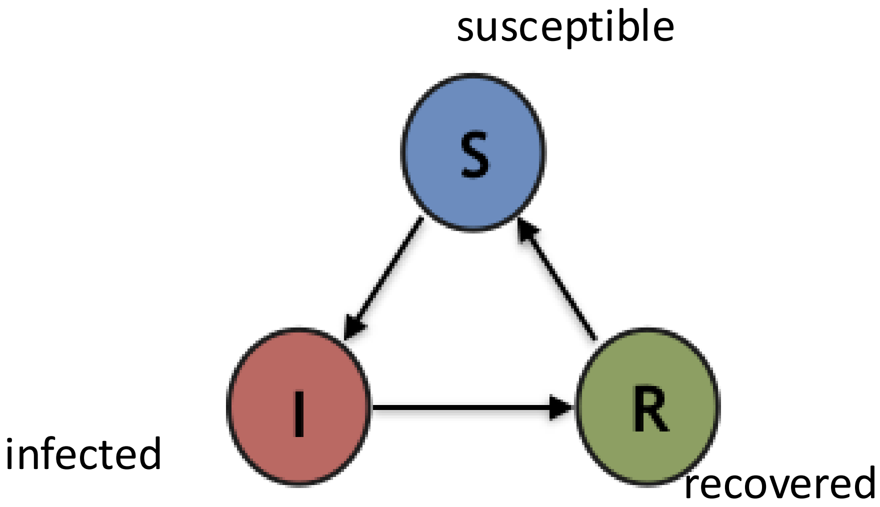
TRANSITIONS

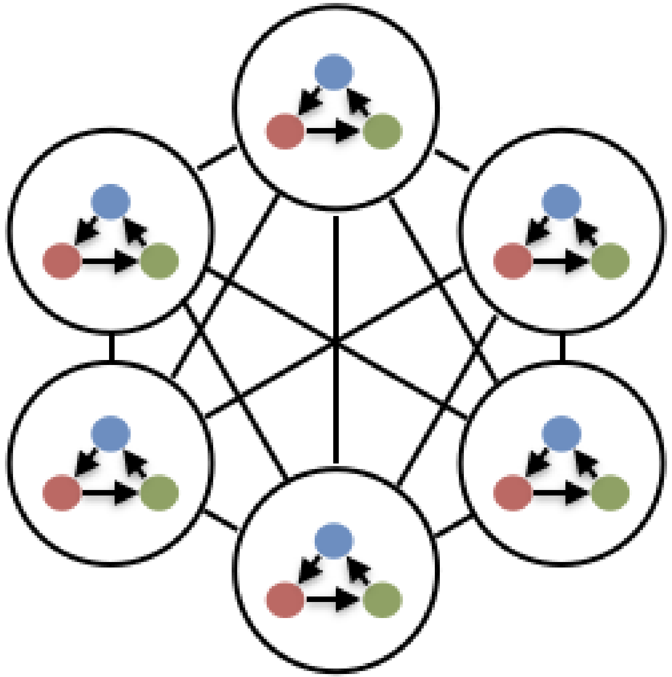
The dynamics is given by a set of chemical reactions:

$$m_1 S_1 + \ldots + m_n S_n \to r_1 S_1 + \ldots + r_n S_n,$$

with a rate given by a function $f(\boldsymbol{X}, \boldsymbol{\theta})$.

Example: SIR epidemic model





infection: $S + I \to 2I$ $\quad f_i(\boldsymbol{X}, \boldsymbol{\theta}) = k_i X_S X_I$

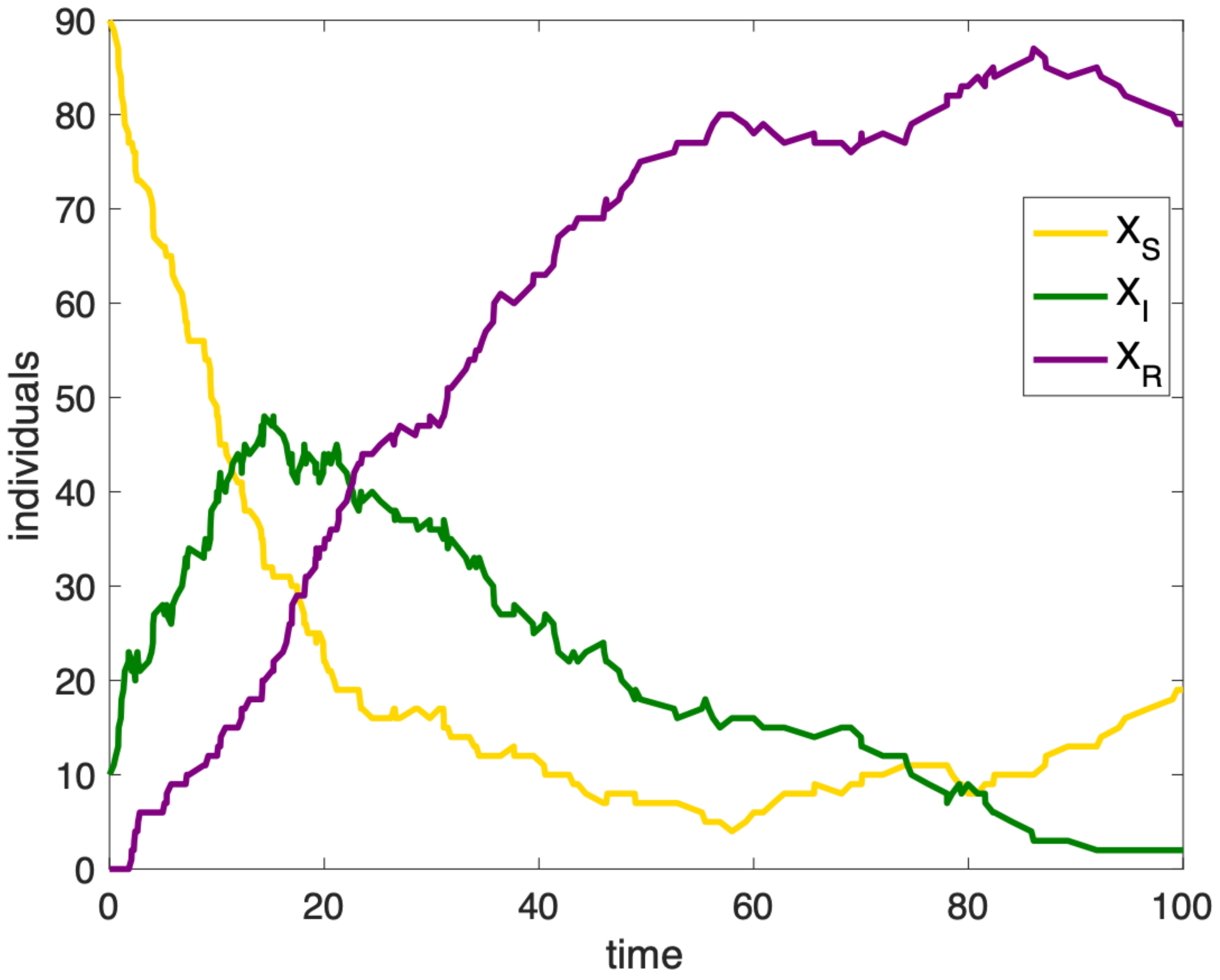
recover: $I \to R$ $\quad f_r(\boldsymbol{X}, \boldsymbol{\theta}) = k_r X_I$

loss of immunity: $R \to S$ $\quad f_l(\boldsymbol{X}, \boldsymbol{\theta}) = k_l X_R$

$$\mathcal{M}_{\boldsymbol{\theta}}$$

State vector: $\mathbf{X} = (X_S, X_I, X_R)$
Vector of parameters: $\boldsymbol{\theta} = (k_i, k_r, k_l)$

Example: SIRS epidemic model



Stochastic Semantics

SATISFACTION PROBABILITY (Boolean Semantics)

$$P(\varphi) = \mathbb{P}\{I_{\varphi}(X) = 1\} := P\{\vec{x} \in Path^{\mathcal{M}} | \mathcal{X}(\vec{x}, 0, \varphi) = 1\}$$

where $I_{\varphi}(X)$ is a Bernoulli random variable

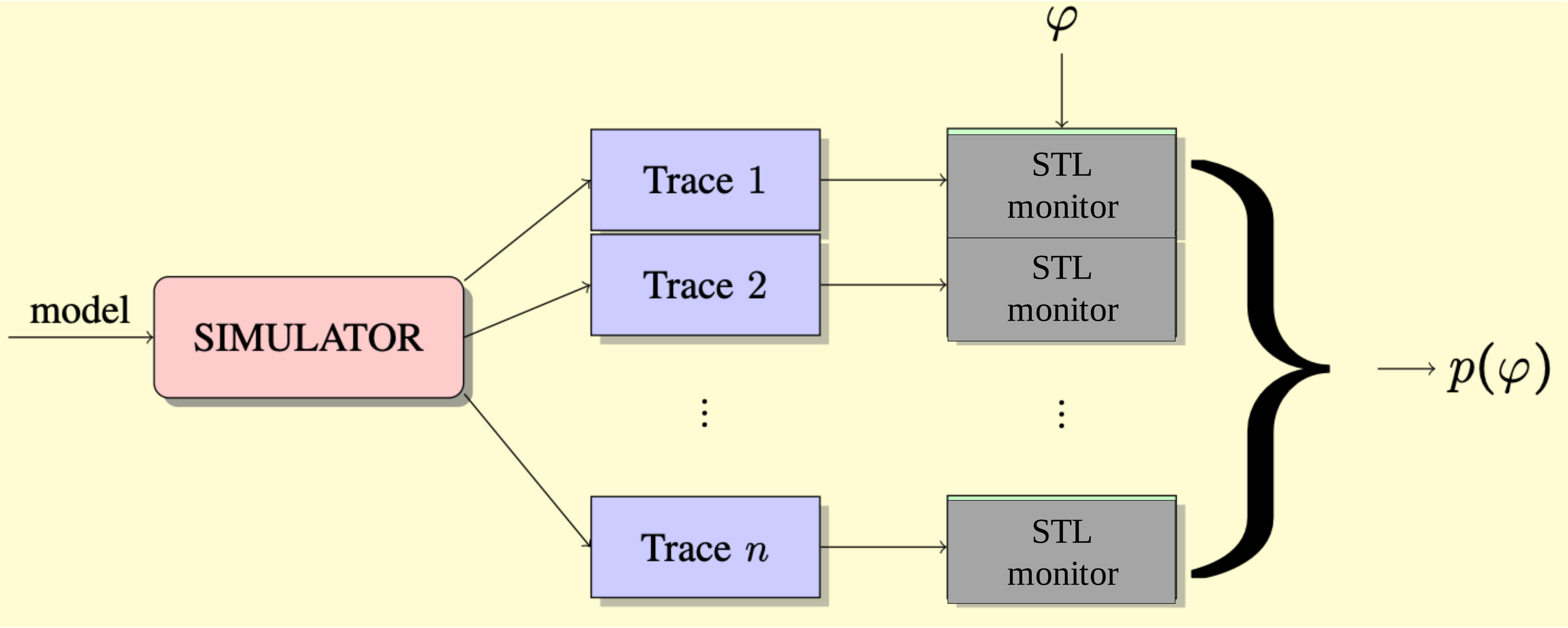
AVERAGE ROBUSTNESS (Quantitative Semantics)

$$\mathbb{P}\{R_{\varphi}(X) \in [a, b]\} := P\{\vec{x} \in Path^{\mathcal{M}} | \rho(\vec{x}, 0, \varphi) \in [a, b]\}$$

where $R_{\varphi}(X)$ in a measurable function

Statistical Model Checking (SMC)

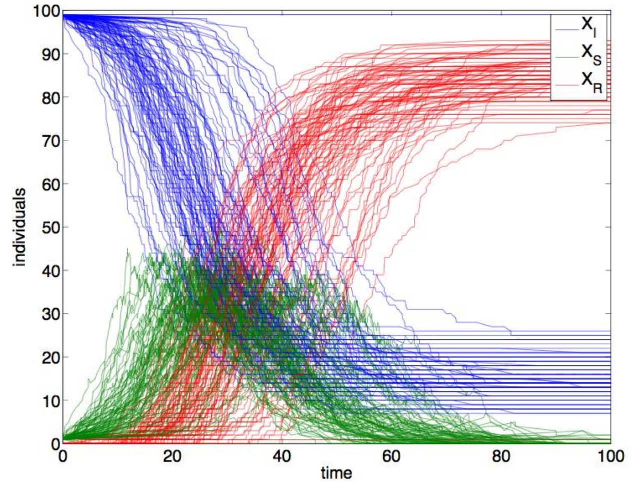
The probability satisfaction can be estimated as an average of the truth values T_i of the formula φ over many sample trajectories.

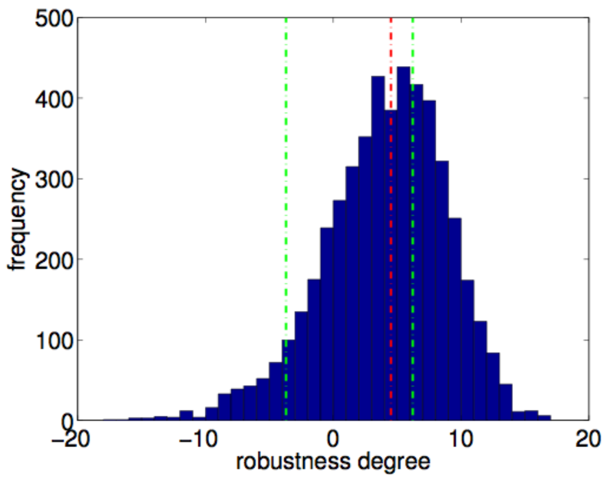


Average robustness degree

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)

Specification	**Natural Language**
Safety ($\square_{[0,\theta]}\phi$)	ϕ should always hold from time 0 to θ.
Liveness ($\Diamond_{[0,\theta]}\phi$)	ϕ should hold at some point from 0 to θ (or now).
Coverage ($\Diamond\phi_1 \wedge \Diamond\phi_2 \ldots \wedge \Diamond\phi_n$)	ϕ_1 through ϕ_n should hold at some point in the future (or now), not necessarily in order or at the same time.
Stabilization ($\Diamond\square\phi$)	At some point in the future (or now), ϕ should always hold.
Recurrence ($\square\Diamond\phi$)	At every point in time, ϕ should hold at some point in the future (or now).
Reactive Response ($\square(\phi \rightarrow \psi)$)	At every point in time, if ϕ holds then ψ should hold.

The many uses of STL and its quantitative semantics

- Requirement-based testing for closed-loop control models
- Falsification Analysis
- Parameter Synthesis
- Mining Specifications/Requirements from Models
- Online Monitoring
- …

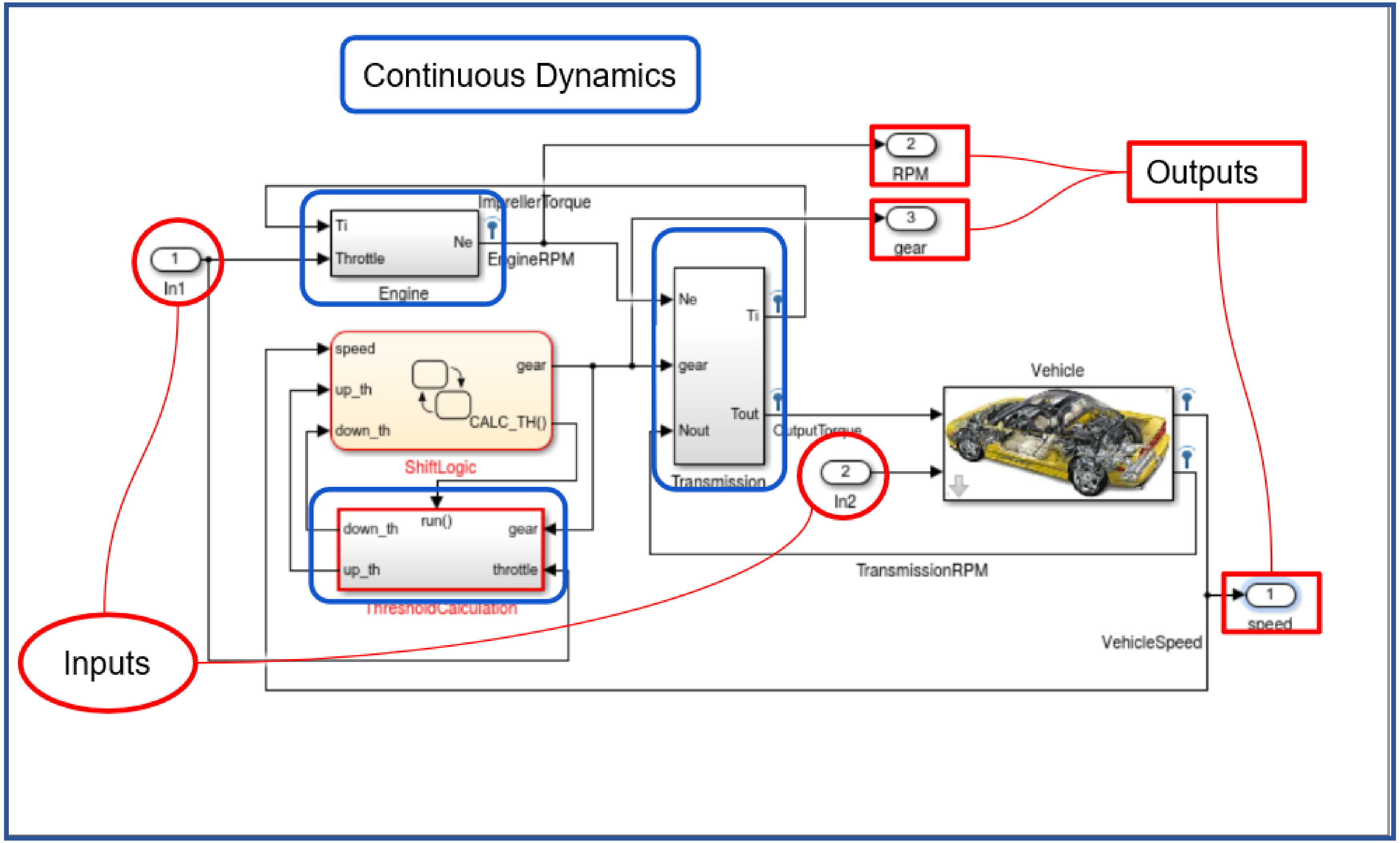
- Testing and falsification

Example

Inputs:

Throttle

Brake



Outputs:

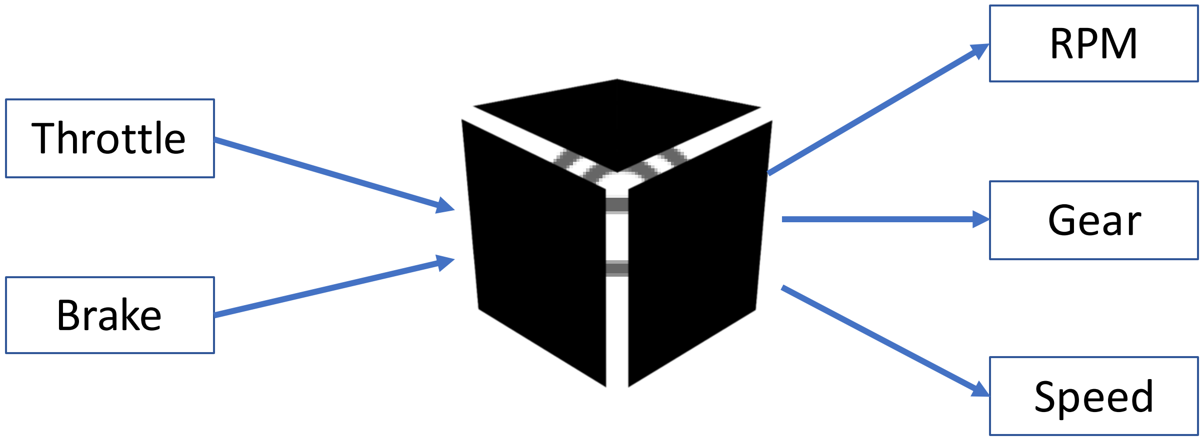
RPM

Gear

Speed

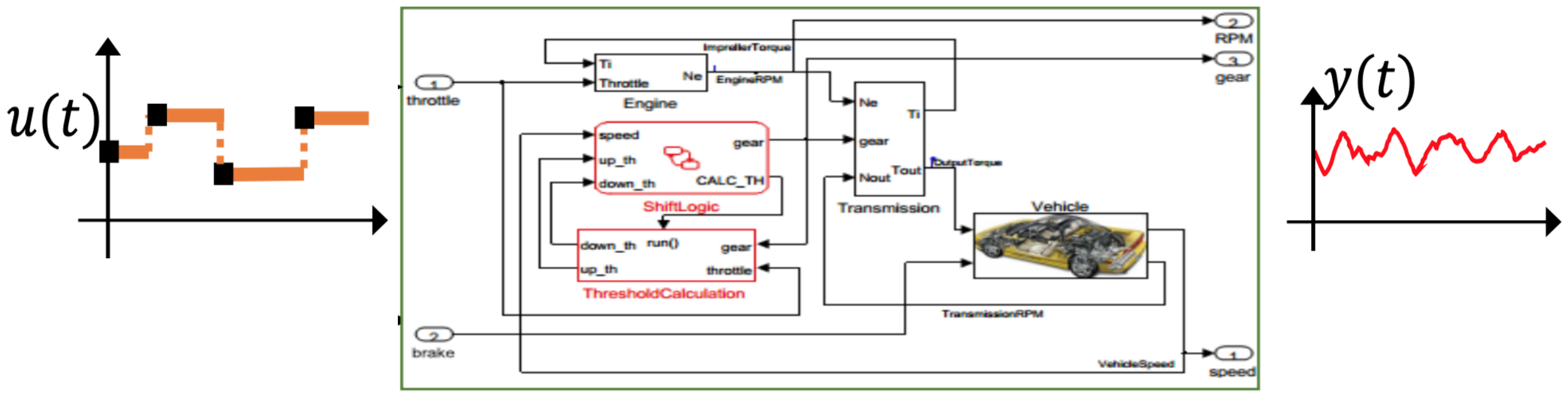
Simulink model of a Car Automatic Gear Transmission Systems

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 $\mathbf{y}(t)$



Automatic Transmission		
	Natural Language	**MTL**
ϕ_1^{AT}	The engine speed never reaches $\bar{\omega}$.	$\Box(\omega < \bar{\omega})$
ϕ_2^{AT}	The engine and the vehicle speed never reach $\bar{\omega}$ and $\bar{v}$, resp.	$\Box((\omega < \bar{\omega}) \wedge (v < \bar{v}))$
ϕ_3^{AT}	There should be no transition from gear two to gear one and back to gear two in less than 2.5 sec.	$\Box((g_2 \wedge X g_1) \rightarrow \Box_{(0,2.5]} \neg g_2)$
ϕ_4^{AT}	After shifting into gear one, there should be no shift from gear one to any other gear within 2.5 sec.	$\Box((\neg g_1 \wedge X g_1) \rightarrow \Box_{(0,2.5]} g_1)$
ϕ_5^{AT}	When shifting into any gear, there should be no shift from that gear to any other gear within 2.5sec.	$\wedge_{i=1}^{4} \Box((\neg g_i \wedge X g_i) \rightarrow \Box_{(0,2.5]} g_i)$
ϕ_6^{AT}	If engine speed is always less than $\bar{\omega}$, then vehicle speed can not exceed $\bar{v}$ in less than T sec.	$\neg(\Diamond_{[0,T]}(v > \bar{v}) \wedge \Box(\omega < \bar{\omega}))$
ϕ_7^{AT}	Within T sec the vehicle speed is above $\bar{v}$ and from that point on the engine speed is always less than $\bar{\omega}$.	$\Diamond_{[0,T]}((v \geq \bar{v}) \wedge \Box(\omega < \bar{\omega}))$
ϕ_8^{AT}	A gear increase from first to fourth in under 10secs, ending in an RPM above $\bar{\omega}$ within 2 seconds of that, should result in a vehicle speed above $\bar{v}$.	$((g_1 \,\mathcal{U}\, g_2 \,\mathcal{U}\, g_3 \,\mathcal{U}\, g_4) \wedge \Diamond_{[0,10]}(g_4 \wedge \Diamond_{[0,2]}(\omega \geq \bar{\omega}))) \rightarrow \Diamond_{[0,10]}(g_4 \rightarrow X(g_4 \,\mathcal{U}_{[0,1]}\, (v \geq \bar{v})))$

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

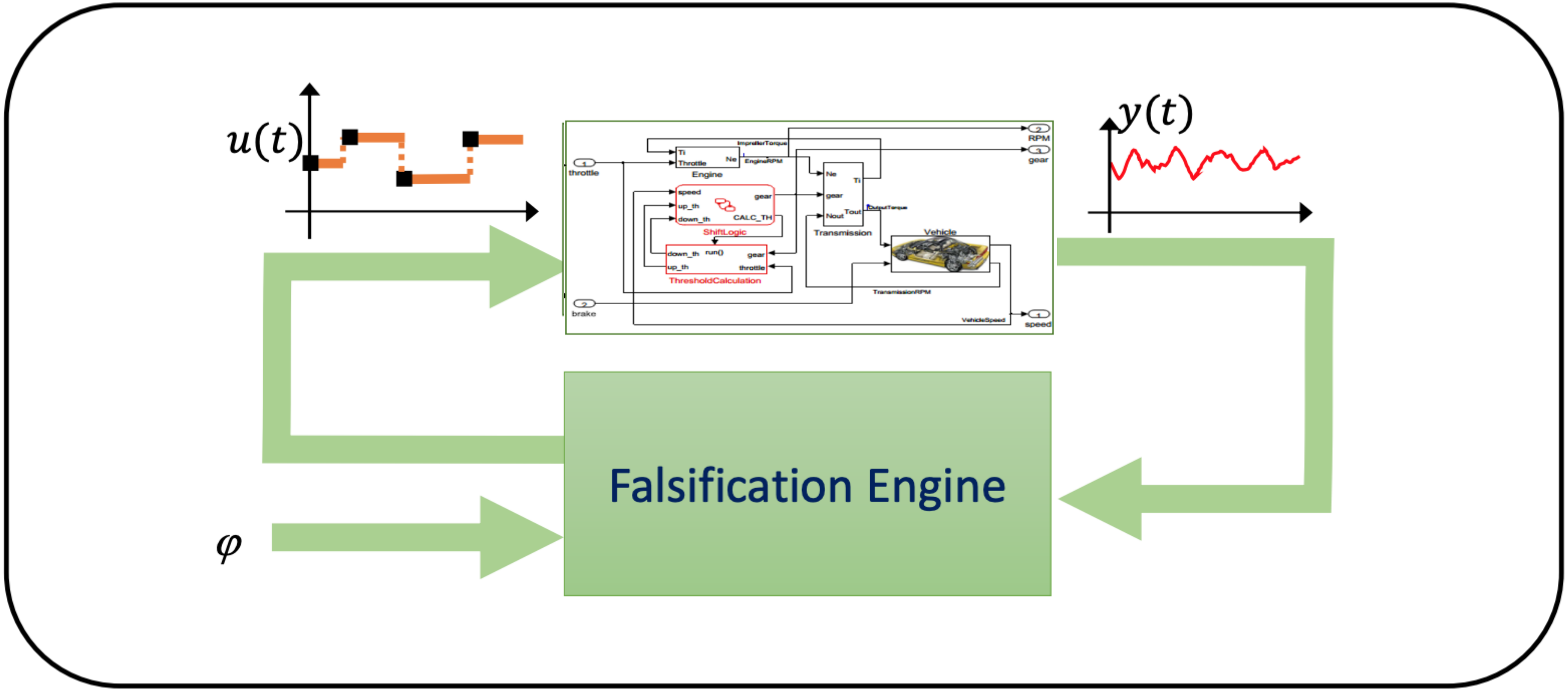
Verification vs. Testing

- 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:

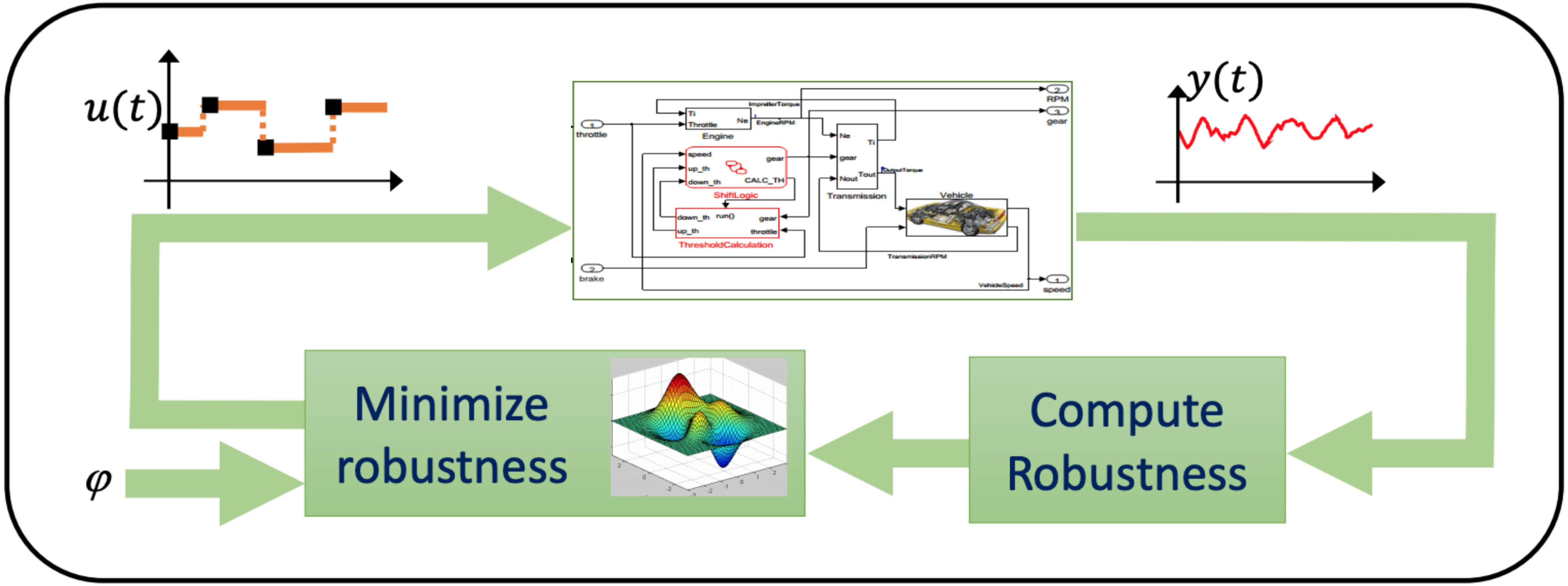
  Prove the following: $\forall \mathbf{u} \in U : \left(\mathbf{y} = M(\mathbf{u})\right) \vDash \varphi(\mathbf{u}, \mathbf{y})$
- Falsification/Testing Problem:

  Find a witness to the query: $\exists \mathbf{u} \in U : \left(\mathbf{y} = M(\mathbf{u})\right) \nvDash \varphi(\mathbf{u}, \mathbf{y})$
- 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.,

Falsification/Testing



Falsification by optimization



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

Falsification/Testing

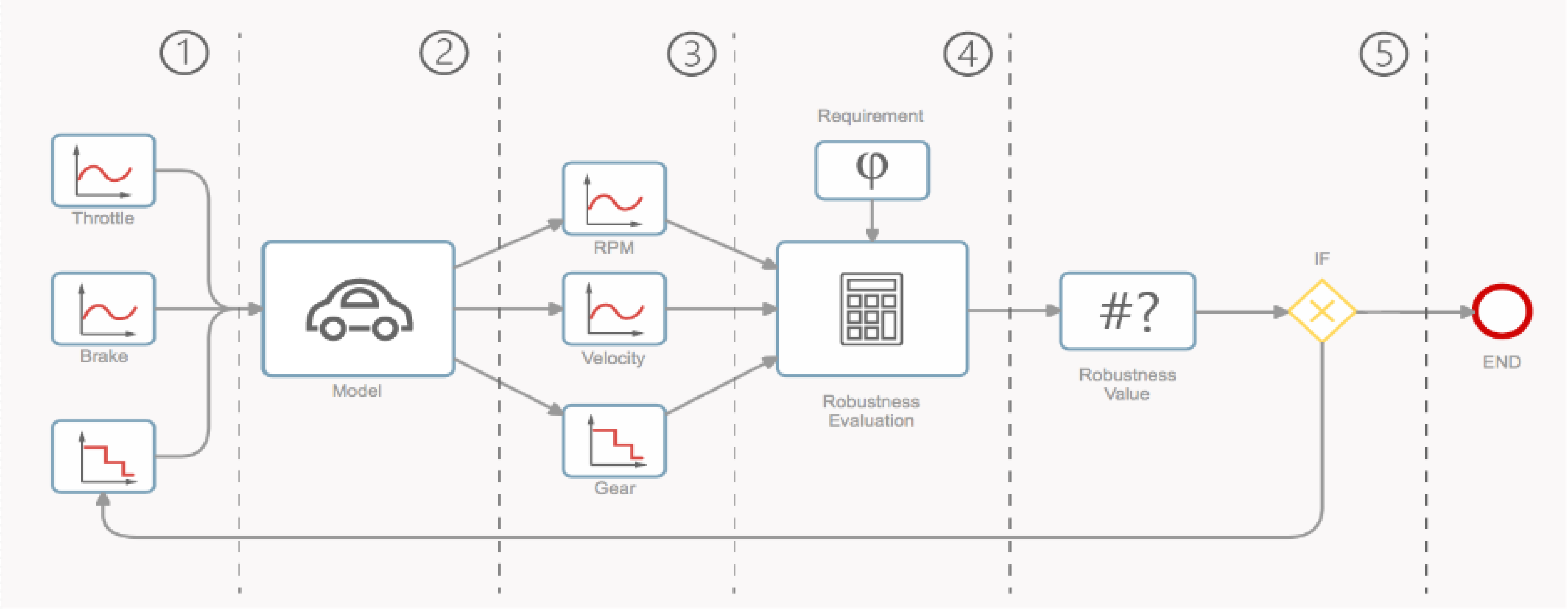
- Falsification or testing attempts to find one or more $\mathbf{u}$ signals such that $\neg\varphi(\mathbf{u}, M(\mathbf{u}))$ is true.

- 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 $\mathbf{u}$: $\mathbb{T}$ -> $D_1 \times \cdots \times D_m$, where $\mathbb{T} \subseteq [0, T]$
In simple words: input signals range over bounded intervals and over a bounded time horizon

Falsification 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

Falsification re-framed

Given:

- Set of all such input signals : U
- Input signal $\mathbf{u} : \mathbb{T} \rightarrow D_1 \times \cdots \times D_m,$ where $\mathbb{T} \subseteq [0, T], D_i \subset \mathbb{R}$ compact set
- Model M s.t. $M(\mathbf{u}) = \mathbf{y}, \quad \mathbf{y}: \mathbb{T} \rightarrow \mathbb{R}^n$
  M 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: $\exists \mathbf{u} \in U : \left(\mathbf{y} = M(\mathbf{u})\right) \vDash \neg\varphi(\mathbf{u}, \mathbf{y})$

Input/Output Properties for Closed-loop Models

- Properties/Specifications/Requirements are rarely monolithic formulas $\varphi(\mathbf{u}, \mathbf{y})$
- Typically specified as a pair: a pre-condition φ_I on the inputs, and a post-condition φ_O on the outputs
- Verification problem then stated as:

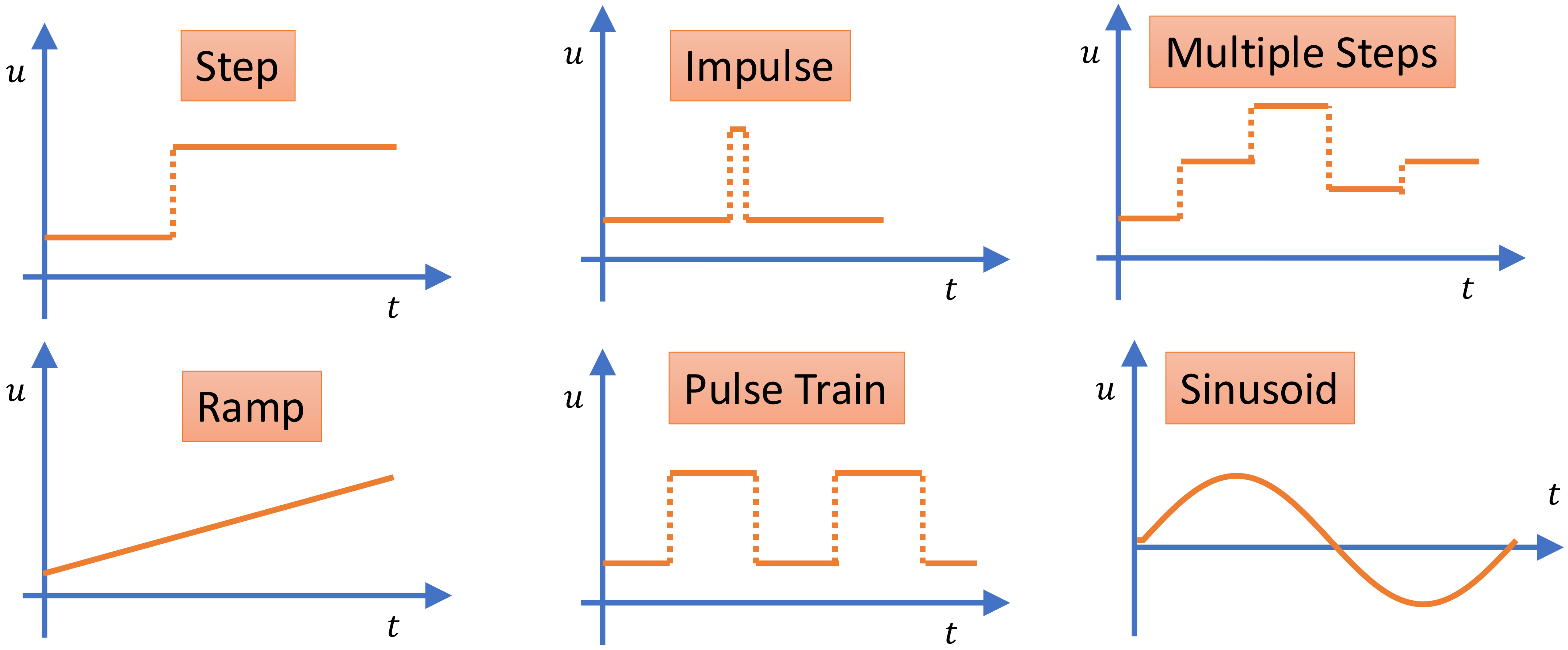
  Prove that: $\forall \mathbf{u} \in U: (\mathbf{u} \vDash \varphi_I) \wedge \left(\mathbf{y} = M(\mathbf{u})\right) \Rightarrow (\mathbf{y} \vDash \varphi_O)$
- Testing problem stated as:

  Find u such that $(\mathbf{u} \vDash \varphi_I) \wedge \left(\mathbf{y} = M(\mathbf{u})\right) \wedge (\mathbf{y} \nvDash \varphi_O)$

Input Properties/Pre-conditions

- Common practice in control theory to excite closed-loop models with input signals of certain special shapes
- Motivation comes from theory of linear systems, where a *step-response* or *impulse-response* are enough to characterize all behaviors of the system
- Such special shapes do not provide comprehensive information for nonlinear closed-loop systems, yet, it is still common to excite these systems with a few common patterns
- Frequently, input signal patterns come from engineering insights or application-specific domain expertise

Common input patterns used for testing



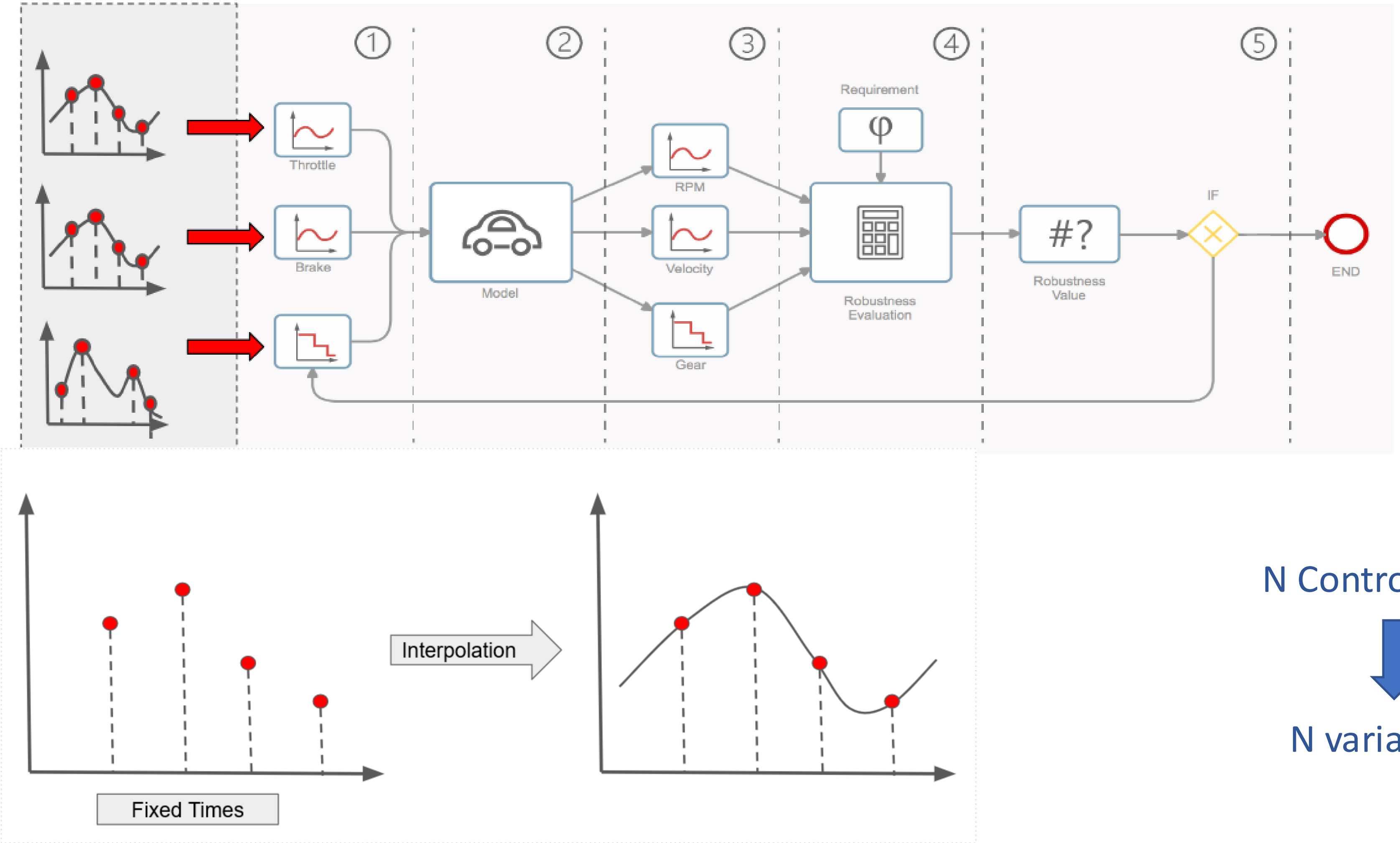
Testing in practice

- Each time-point in a signal is an independent dimension, i.e. the signal can change arbitrarily at each time-point in the signal
- Number of independent domains is infinite (e.g. consider a signal defined over rational time-points)
- Typical testing approach is to find a *test-suite:* This is a **finite** number of test input signals (satisfying φ_I) and then obtain output behaviors using these signals as test inputs.
- If each corresponding output signal satisfies the output property φ_0, then testing concludes, indicating that the model is correct for the given test-suite (i.e. no output in the test-suite satisfies φ_0).

Signal Generation

- Find a *signal generator* for the property φ_I
  - Function that uses random-ness to generate an input signal that satisfies φ_I (hopefully, an input signal different from previously generated ones!)
- Signal generation usually relies on defining a *finite parameterization* for the input signal
  - For the chosen class of signals, find parameters that define the shape
  - Define acceptable ranges for the parameters
  - Define a generation function that takes the *parameter values* as inputs and generates an input signal

Finite Parameterization



N Control points

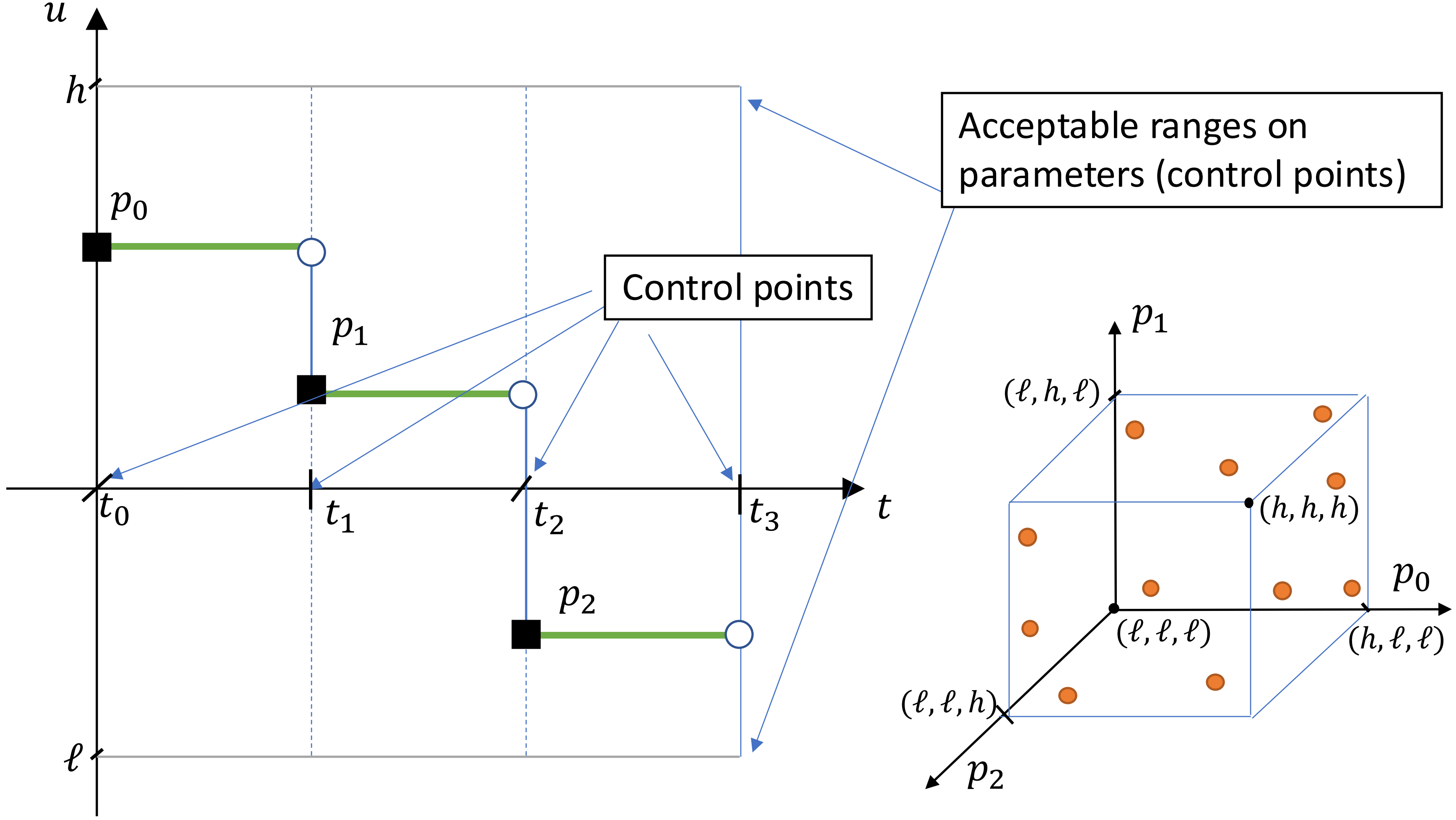
N variable

Finite parameterization using control points

Finite Parameterization of $u(t)$:

$$u(t) = \begin{cases} p_0 \text{ if } \mathrm{t}_0 \le t < t_1 \\ p_1 \text{ if } \mathrm{t}_1 \le t < t_2 \\ p_2 \text{ if } \mathrm{t}_2 \le t < t_3 \end{cases}$$

We can view this as values of u are picked for (fixed) time points (determined *a priori*), and then $u(t)$ is generated using constant interpolation

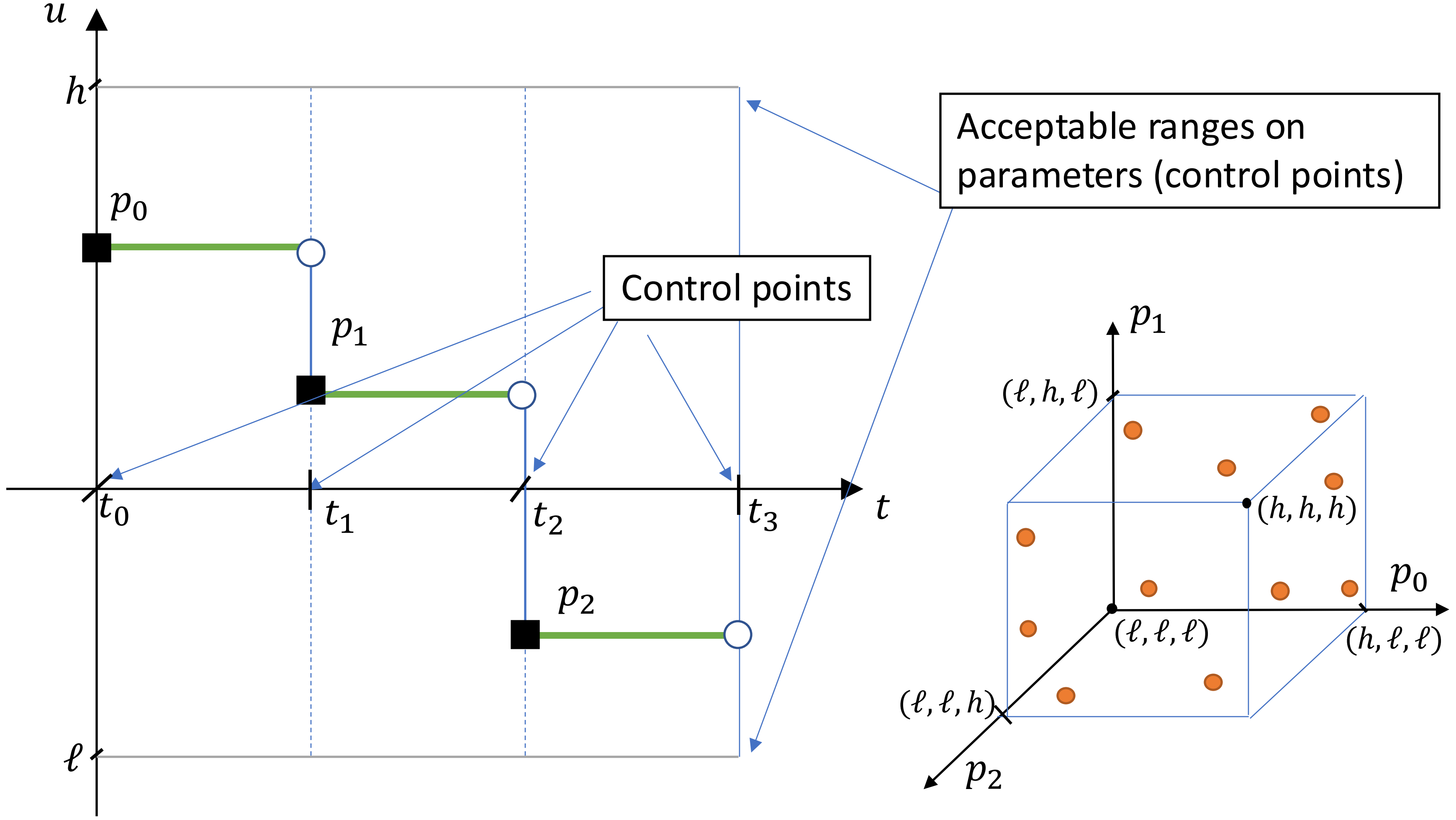


Finite parameterization using control points

Finite Parameterization of $u(t)$:

$$u(t) = \begin{cases} p_0 \text{ if } \mathrm{t}_0 \leq t < t_1 \\ p_1 \text{ if } \mathrm{t}_1 \leq t < t_2 \\ p_2 \text{ if } \mathrm{t}_2 \leq t < t_3 \end{cases}$$

We can view this as values of u are picked for (fixed) time points (determined *a priori*), and then $u(t)$ is generated using constant interpolation

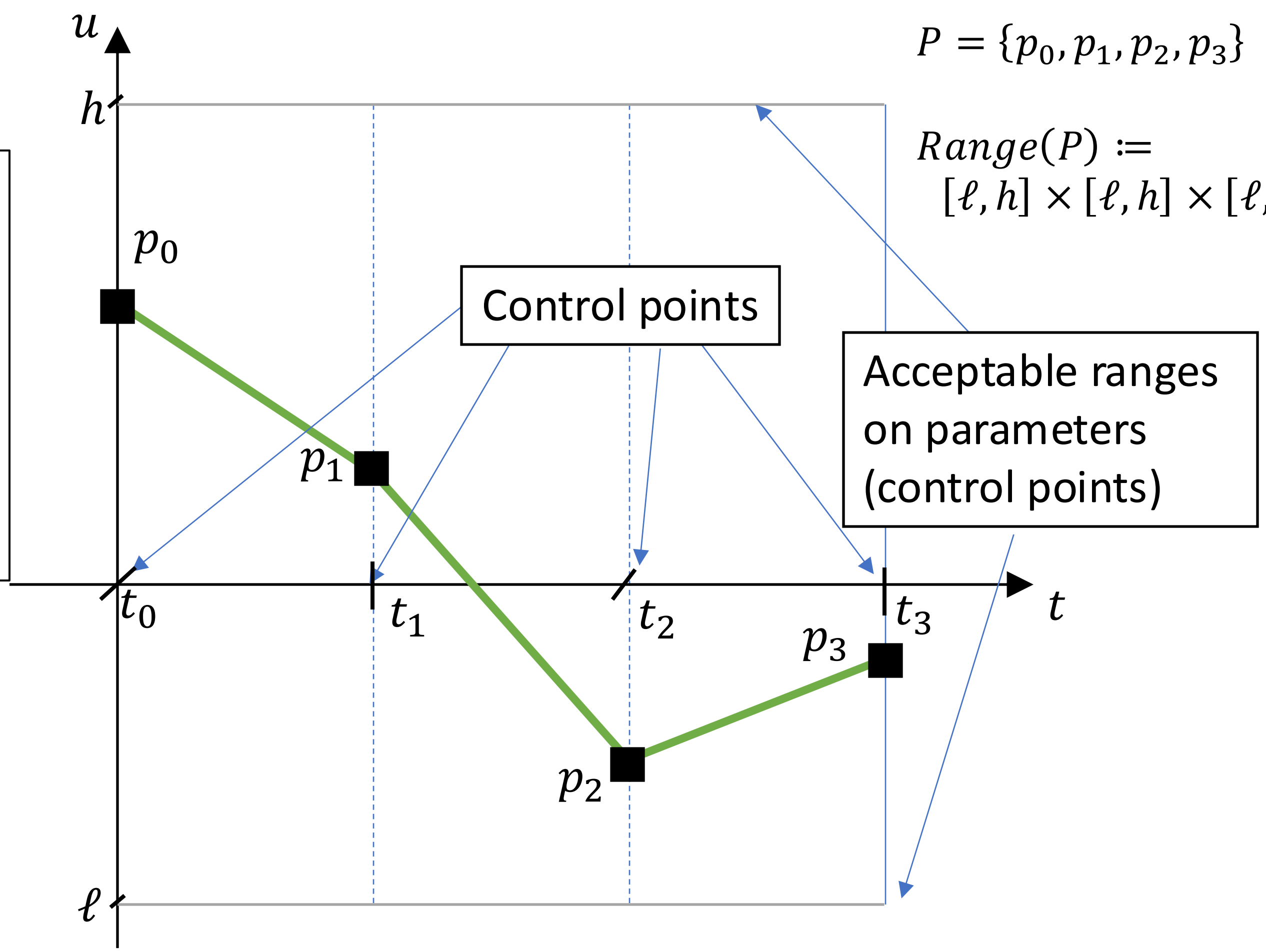


Finite parameterization using linear interpolation

Finite Parameterization of $u(t)$:

$$u(t) = \begin{cases} p_0 + (t - t_0) \cdot \dfrac{p_1 - p_0}{t_1 - t_0} & \text{if } \mathrm{t}_0 \leq t < t_1 \\ p_1 + (t - t_1) \cdot \dfrac{p_2 - p_1}{t_2 - t_1} & \text{if } \mathrm{t}_1 \leq t < t_2 \\ p_2 + (t - t_2) \cdot \dfrac{p_3 - p_2}{t_3 - t_2} & \text{if } \mathrm{t}_2 \leq t < t_3 \end{cases}$$

We can view this as values of u are picked for (fixed) time points (determined *a priori*), and then $u(t)$ is generated using linear interpolation



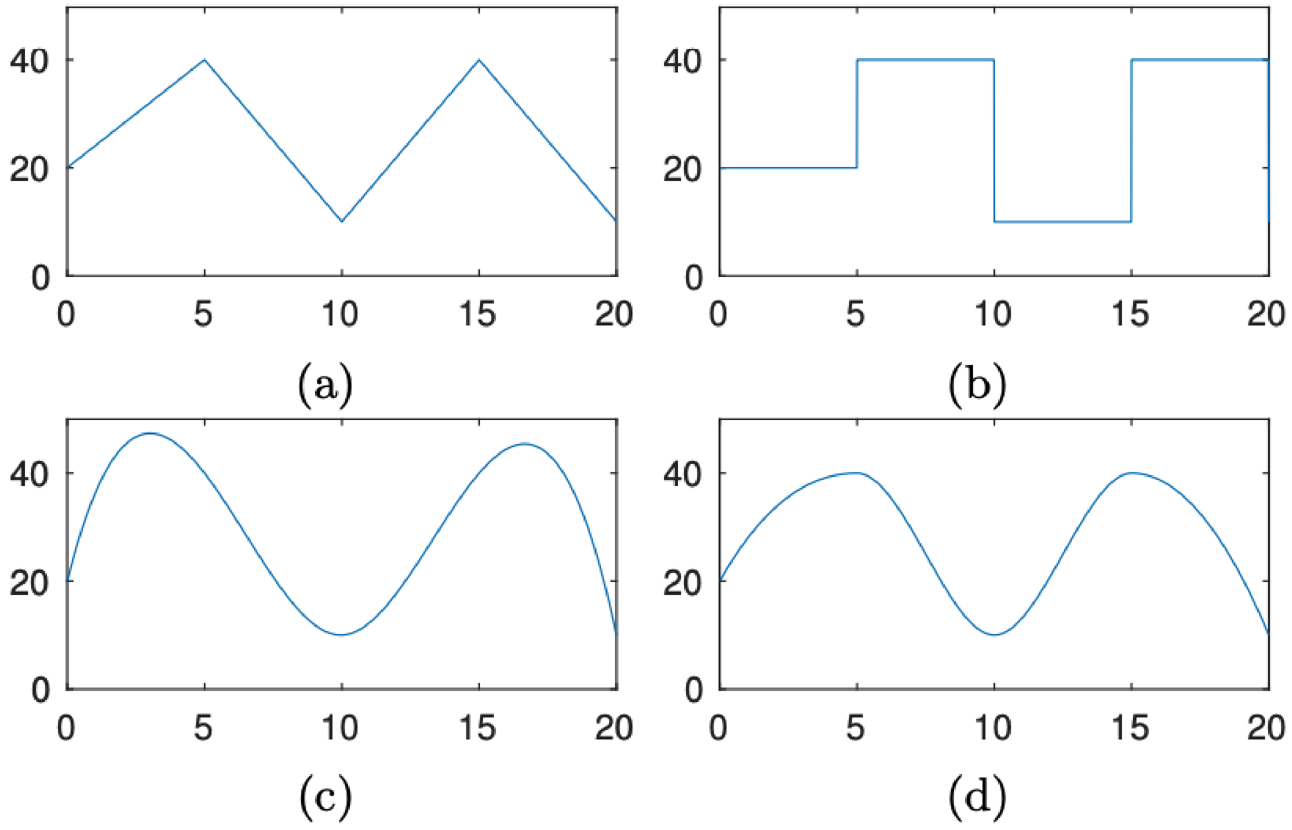
$P = \{p_0, p_1, p_2, p_3\}$

$Range(P) \coloneqq [\ell, h] \times [\ell, h] \times [\ell, h] \times [\ell, h]$

Finite parameterization using interpolation

Linear

Piecewise constant



Spline

Piecewise cubic interpolation

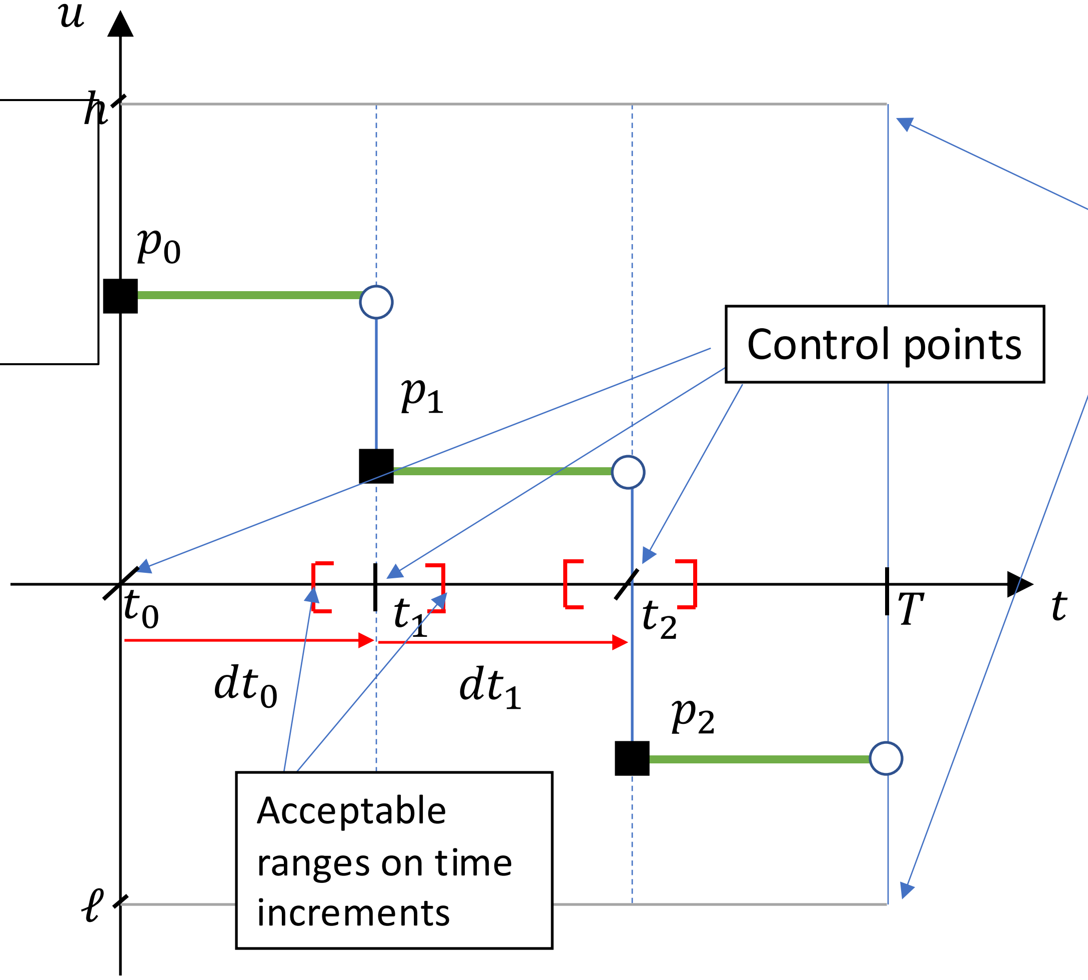
$\lambda = [20, 40, 10, 40, 10]$ $t = [0, 5, 10, 15, 20]$

Finite parameterization variable control point times

Finite Parameterization of $u(t)$:

$$u(t) = \begin{cases} p_0 \text{ if } t_0 \leq t < t_0 + dt_0 \\ p_1 \text{ if } t_1 \leq t < t_1 + dt_1 \\ p_2 \text{ if } t_2 \leq t < T \end{cases}$$

We can view this as values of u and time increments in u are both picked, and then $u(t)$ is generated using constant interpolation

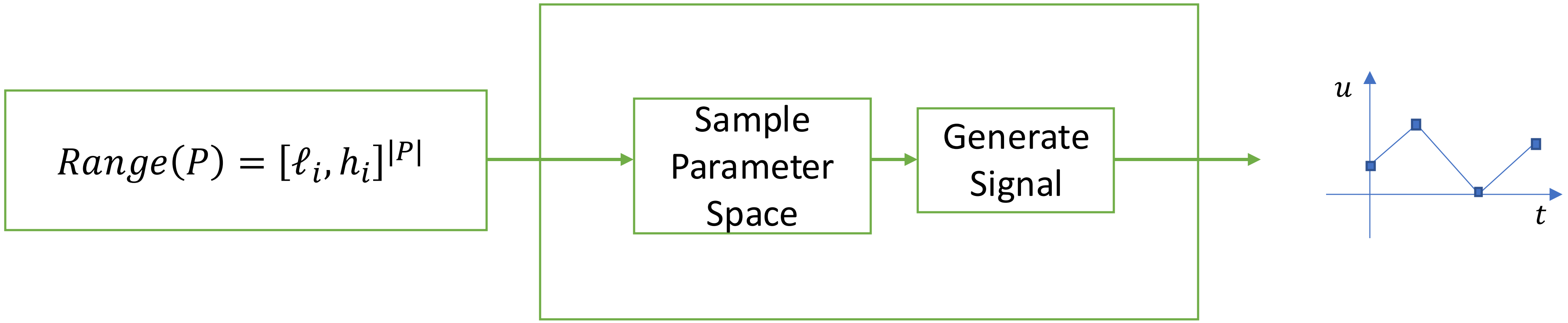


Acceptable ranges on parameter values

$P = \{p_0, p_1, p_2, dt_0, dt_1\}$

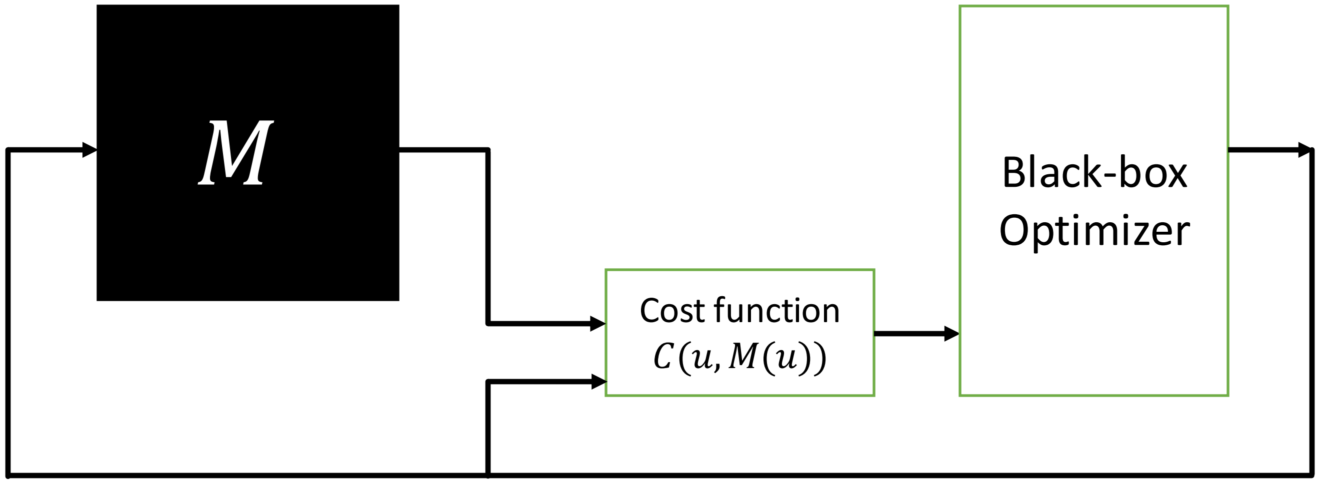
$$Range(P) \coloneqq [\ell, h] \times [\ell, h] \times [\ell, h] \times [\tau_\ell, \tau_h] \times [\tau_\ell, \tau_h]$$

Signal Generator



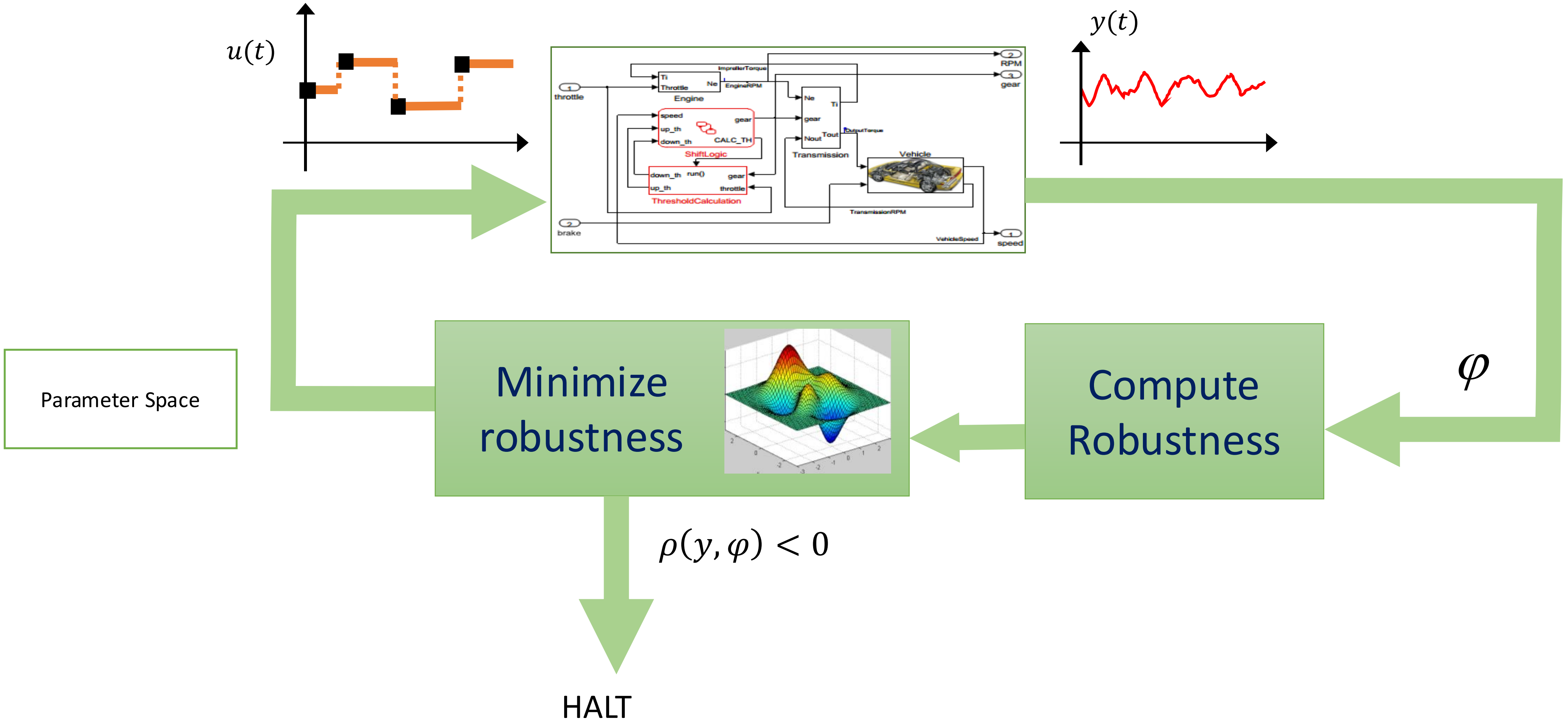
- Signal Generation controlled by the testing algorithm
  - Parameter space could be sampled all at once
  - Parameter space could be sampled in a sequential fashion, e.g. using a method such as Markov Chain Monte Carlo
  - Sampling scheme could be application-specific: uniform random, quasi-random (more evenly spread out), truncated normal, grid-based sampling (points from a fixed grid), etc.

Black-box Optimization



- Given:
  - Function $M: U \rightarrow Y$ with unknown symbolic representation
  - Ability to query the value of M at any given u; query will return some y
  - Cost function $C: X \times Y \rightarrow \mathbb{R}$
- Objective of black-box optimizer
  - Let $x^* = \min_{x \in X} C(x, f(x))$
  - Find $\hat{x}$ such that $||\hat{x} - x^*||$ is small
- Let $\widehat{x_i}$ be the best answer found by optimizer in its i^{th} iteration
- Ideally, $\lim_{i \to \infty} || \widehat{x_i} - x^* || = 0$

Falsification using Optimization



Step-by-step of how falsification works

- Given: a finite parameterization for input signals, a model that can be simulated and an STL property
- While the number of allowed iterations is not exhausted do:
  - pick values for the signal parameters
  - generate an input signal
  - run simulation with generated input signal to get output signal
  - compute robustness value of given property w.r.t. the input/output signals
  - if robustness value is negative, **HALT**
  - pick a new set of values for the signal parameters based on certain heuristics

Picking new parameter values to explore

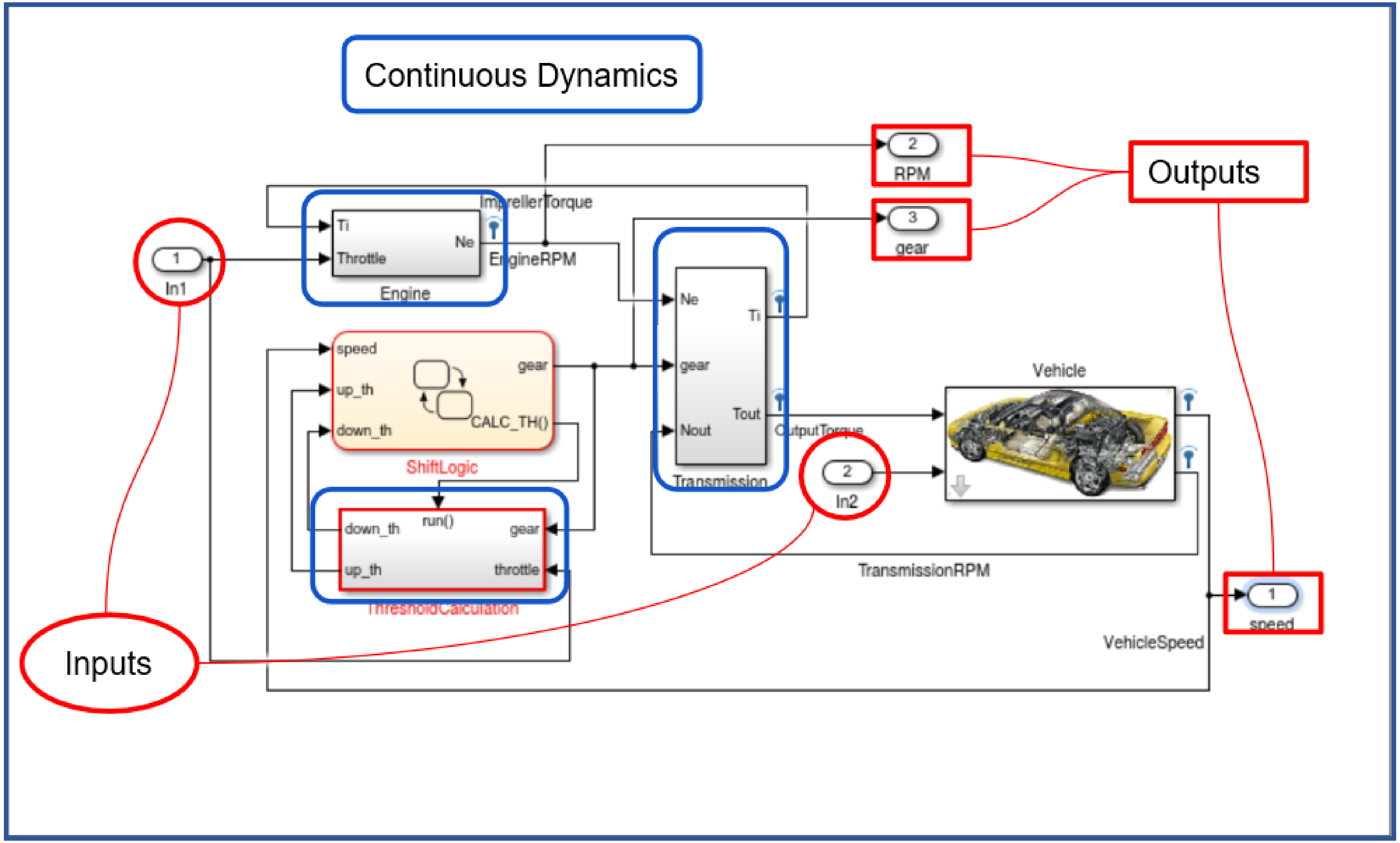
- Pick random sampling as a (not very good) strategy!
- Basic method: locally approximate the gradient of the function ρ locally, and chose the direction of steepest descent (greedy heuristic to take you quickly close to a local optimum)
- Challenge 1: cost surface may not be convex, thus you could have many local optima
- Challenge 2: cost surface may be highly nonlinear and even discontinuous, using just gradient-based methods may not work well
- Heuristics rely on:
  - combining gradient-based methods with perturbing the search strategy (e.g. simulated annealing, stochastic local search with random restarts)
  - evolutionary strategies: Covariance Matrix Adaptation Evolution Strategy (CMA-ES), genetic algorithms etc.
  - probabilistic techniques: Ant Colony Optimization, Cross-Entropy optimization, Bayesian optimization

Model

Inputs:

Throttle

Brake



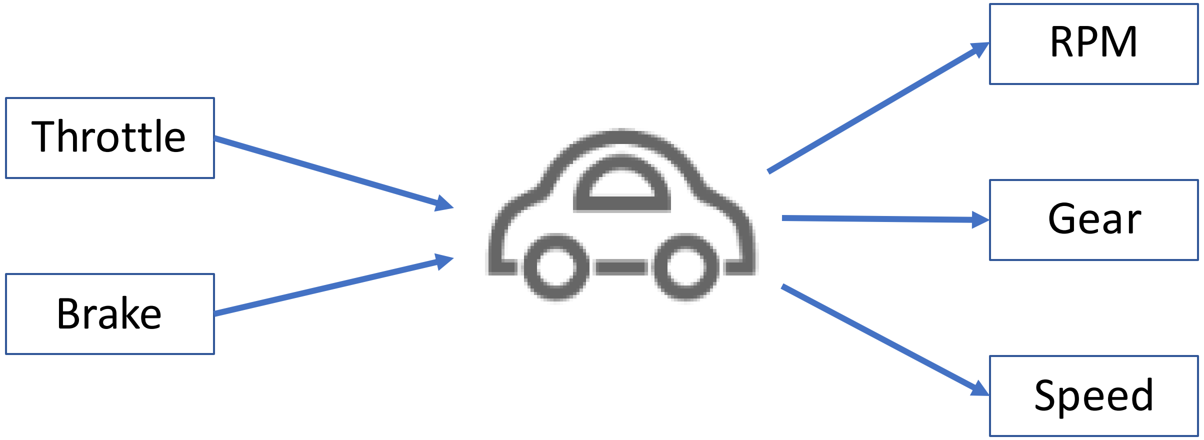
Outputs:

RPM

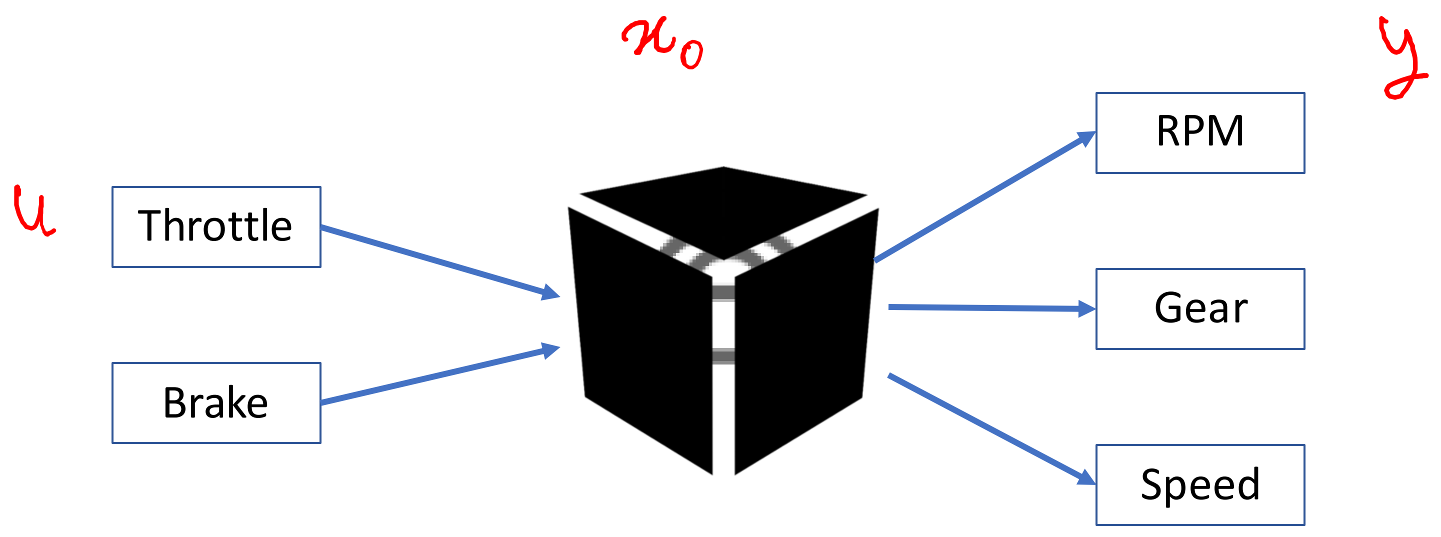
Gear

Speed

Model

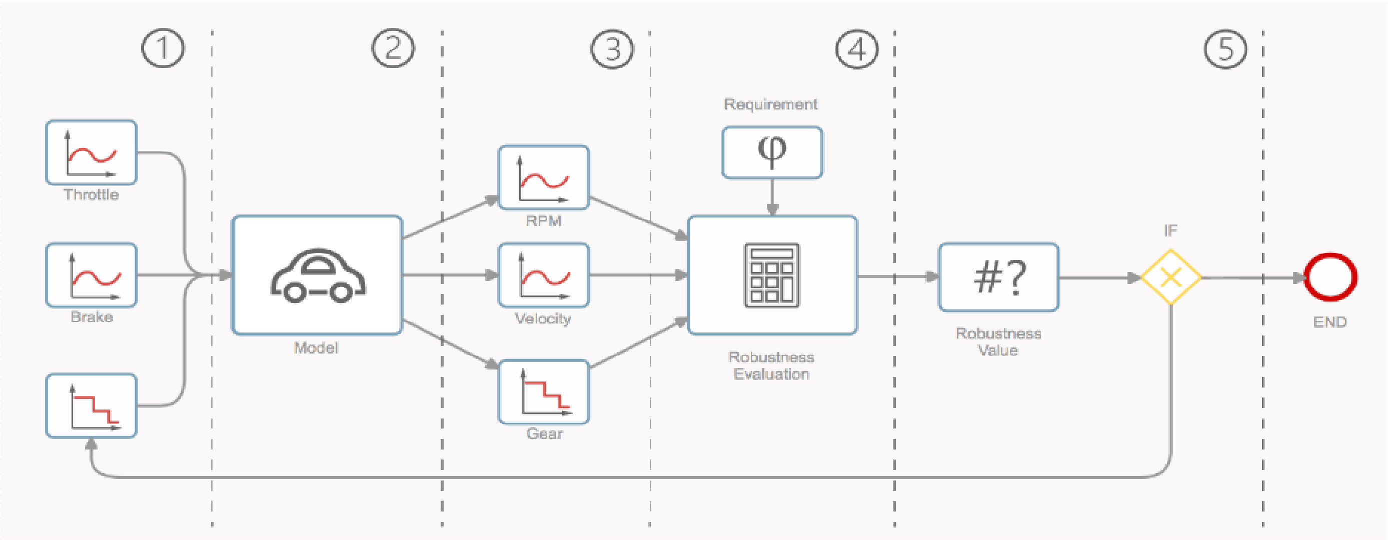


Black Box Assumption



- Less information
- A more general Approach (interesting for industries)

Falsification of 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

Gaussian Processes

Definition

$$f \sim GP(m,k) \iff (f(t_1), f(t_2), \dots, f(t_n)) \sim N(m,K)$$

where $m = (m(t_1), m(t_2), \dots, m(t_n))$ is the vector mean

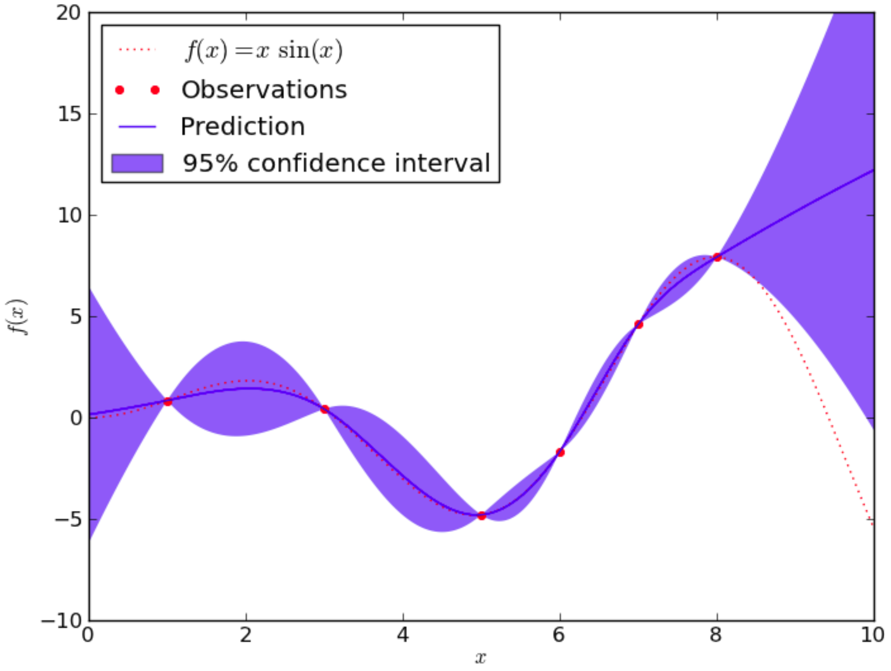
$K \in \mathbb{R}^{n \times n}$ is the covariance matrix, such that $K_{ij} = k(f(t_i), f(t_j))$

Prediction

$$\{\underbrace{f(\theta_1), \dots, f(\theta_n)}_{\mathbf{f}}, f(\theta')\} \sim \mathcal{N}(\mathbf{m}', K')$$

$$\mathbb{E}(f(\theta')) = \underbrace{(k(\theta_1, \theta'), \dots, k(\theta_N, \theta',))}_{\mathbf{k}} \cdot K^{-1} \cdot \mathbf{f}$$

$$var(f(\theta')) = k(\theta', \theta') - \mathbf{k} \cdot K^{-1} \cdot \mathbf{k}^T$$



Gaussian Process Regression

Gaussian Processes can be used for Bayesian prediction and classification tasks.

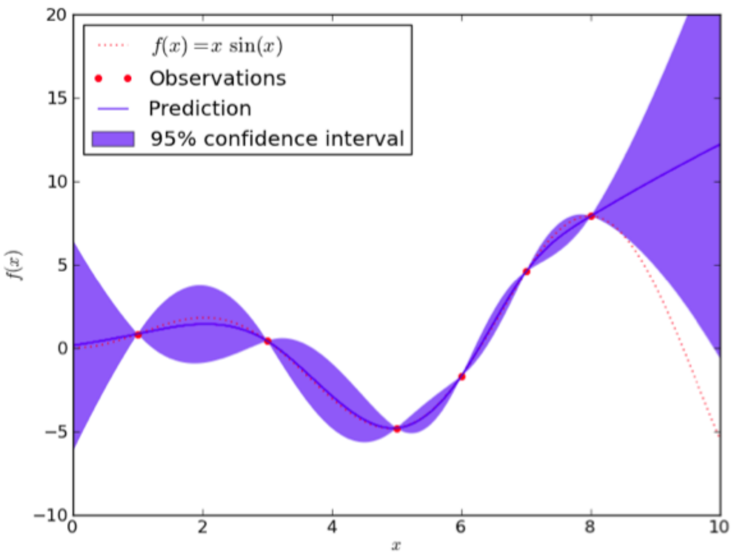
Idea: put a **GP prior** on functions; condition on **observed data (training set)** (x_i, y_i); we compute a **posterior** distribution on functions; make **predictions**.

Latent function: f , GP ; **Noise model**: $p(y_i | f(x_i))$

Prediction (latent function f^* at x^*)

$$p(f^*|\mathbf{y}) \propto \int df(\mathbf{x}) p(f^*, f(\mathbf{x})) p(\mathbf{y}|f(\mathbf{x}))$$

Under Gaussian noise $y(\mathbf{x}) = f(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2)$
predictions have an analytic expression.



$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right)$$

$$\begin{aligned} \mathbf{f}_*|X,\mathbf{y},X_* &\sim \mathcal{N}(\bar{\mathbf{f}}_*, \operatorname{cov}(\mathbf{f}_*)), \quad \text{where} \\ \bar{\mathbf{f}}_* &\triangleq \mathbb{E}[\mathbf{f}_*|X,\mathbf{y},X_*] = K(X_*,X)[K(X,X) + \sigma_n^2 I]^{-1}\mathbf{y}, \\ \operatorname{cov}(\mathbf{f}_*) &= K(X_*,X_*) - K(X_*,X)[K(X,X) + \sigma_n^2 I]^{-1} K(X,X_*) \end{aligned}$$

Domain Estimation Problem

Finding the trajectories which falsify the requirements, finding $\boldsymbol{u} \in B$

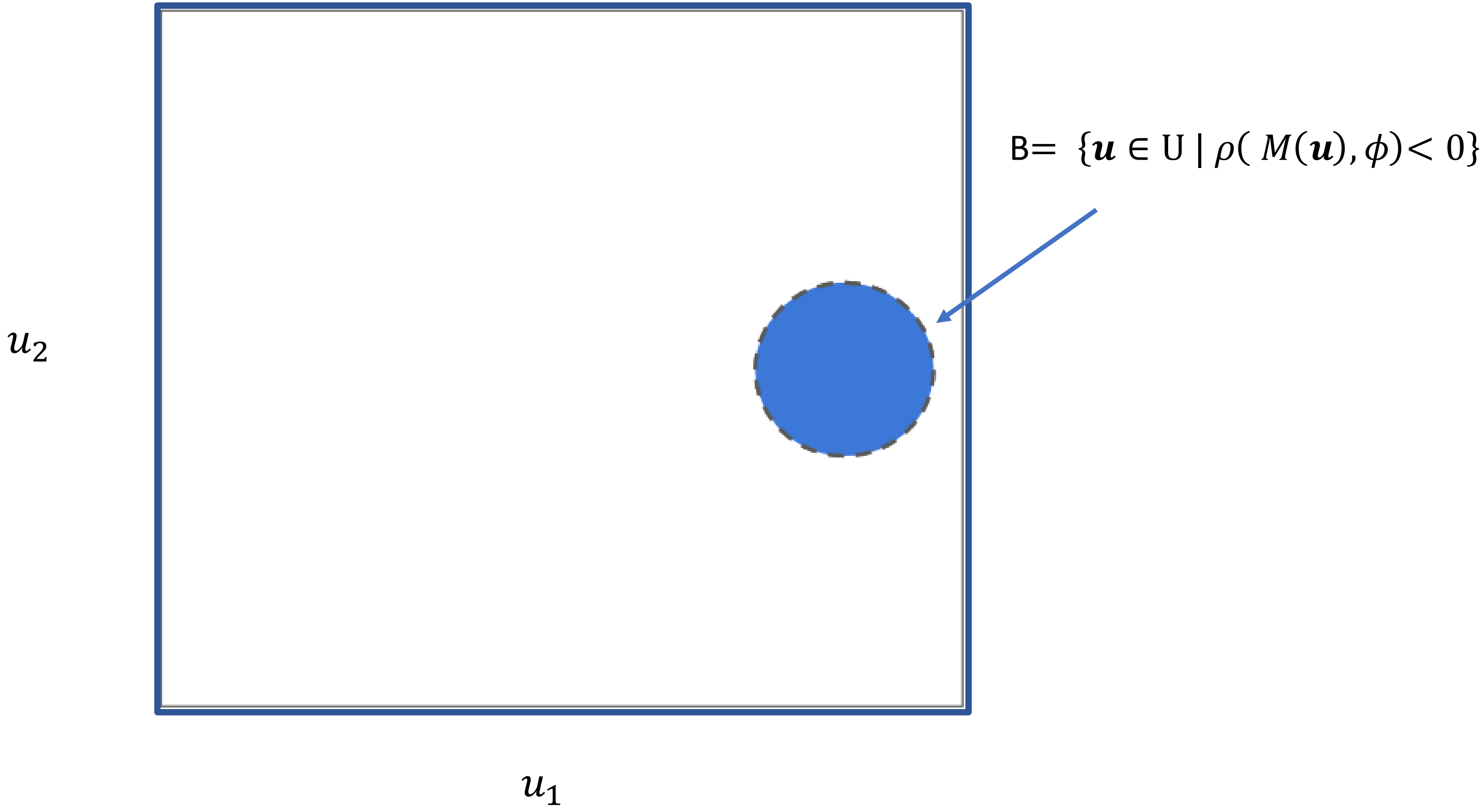
$$\mathrm{B} = \{\boldsymbol{u} \in \mathrm{U} \mid \rho(\phi, \boldsymbol{u}, 0) < 0\} \subseteq U$$

- **Training Set**: $\mathrm{K} = \{\boldsymbol{u}_{\mathrm{i}}, \rho(\phi, \boldsymbol{u}_i, 0))\}_{i \leq n}$ (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)

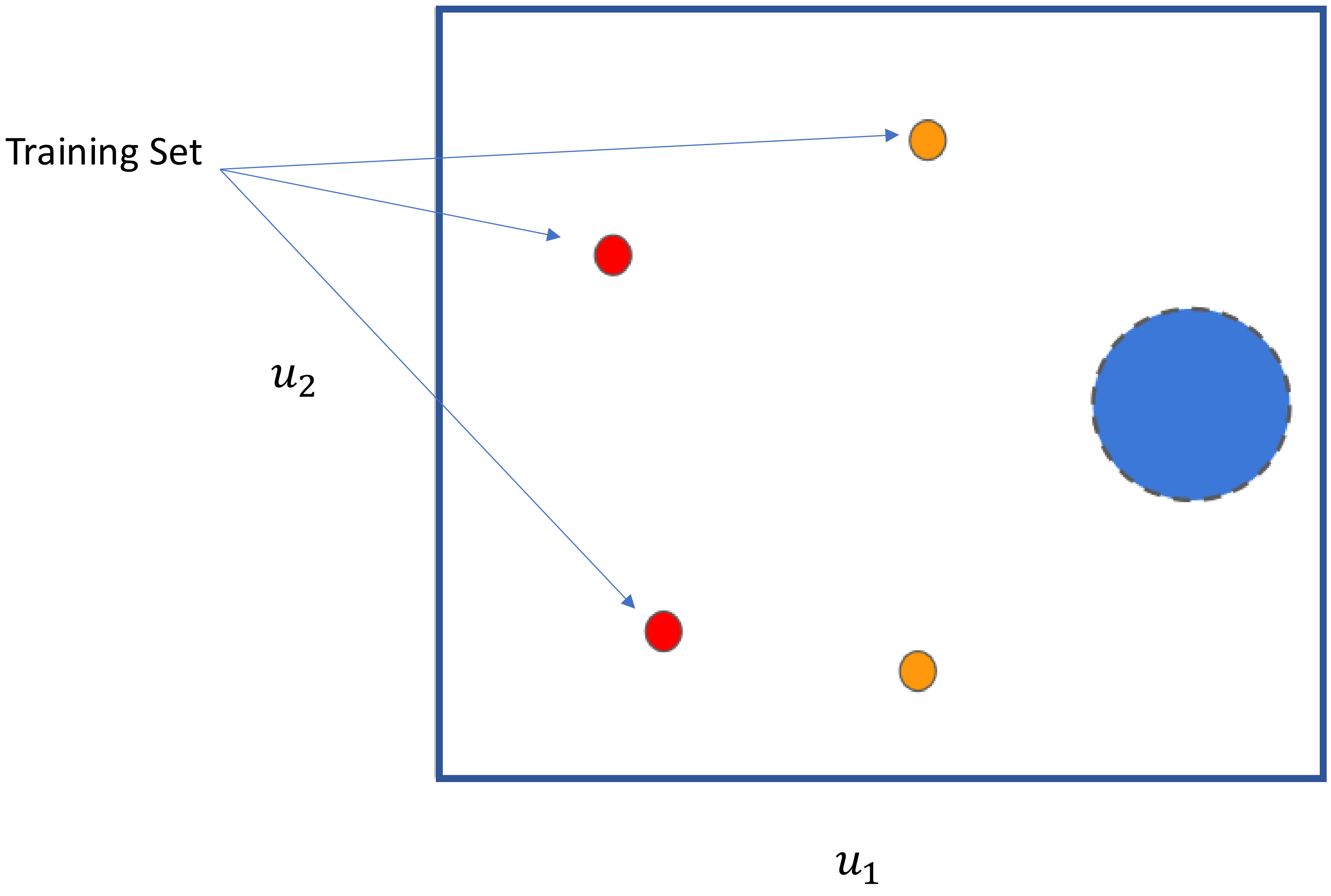
$$P(\rho_K(\boldsymbol{u}) < 0) = CDF\left(\frac{0 - m_K(\boldsymbol{u})}{\sigma_K(\boldsymbol{u})}\right)$$

Idea: implementing an iterative sample strategy in order to increase the probability to sample a point in B, as the number of iterations increases.

Domain Estimation Algorithm (DEA)

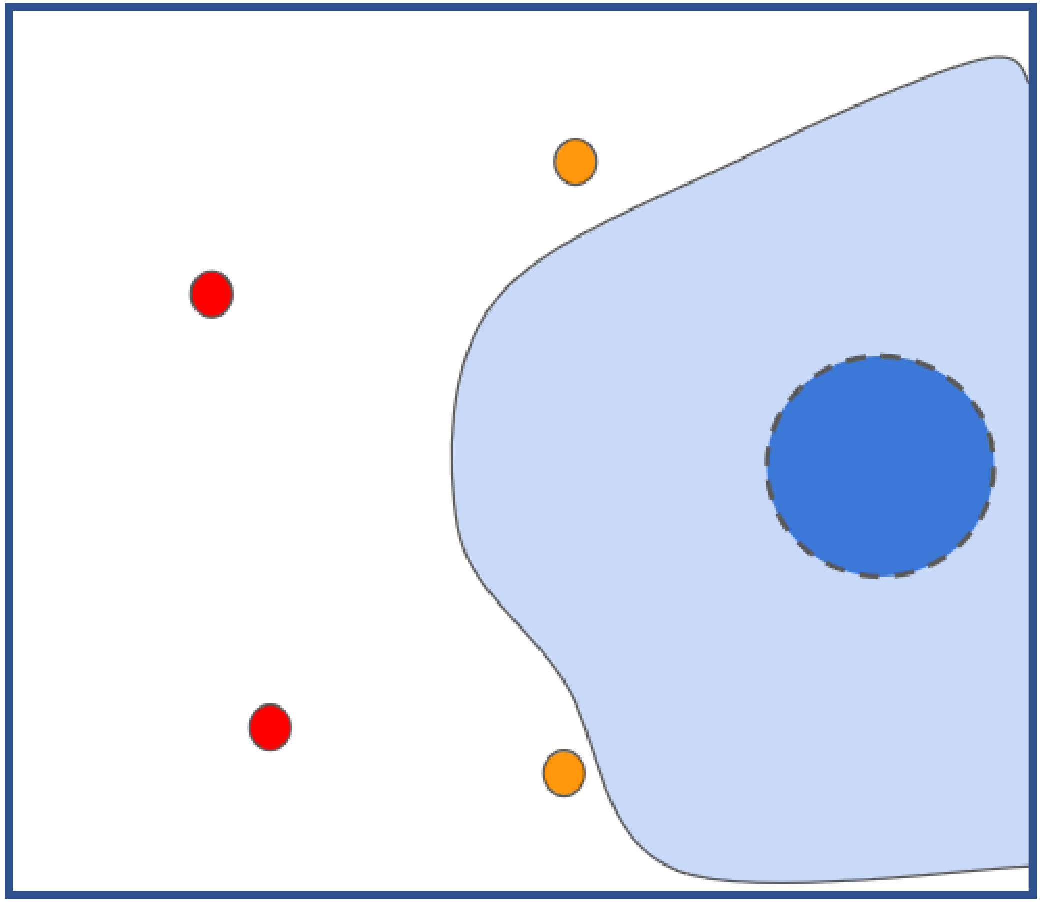


Domain Estimation Algorithm (DEA)



Domain Estimation Algorithm (DEA)

u_2

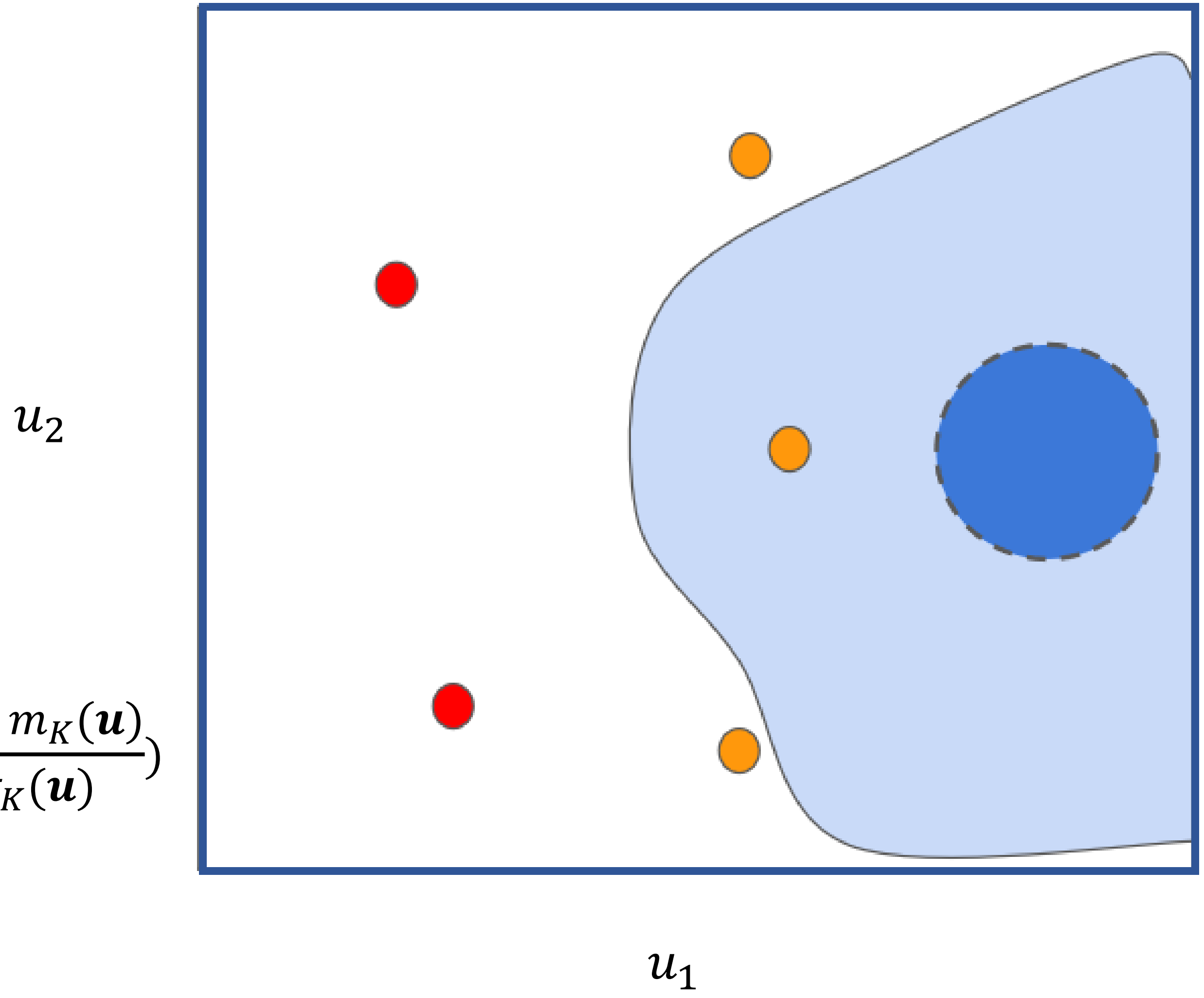


u_1

Sample a new point accordingly to:

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

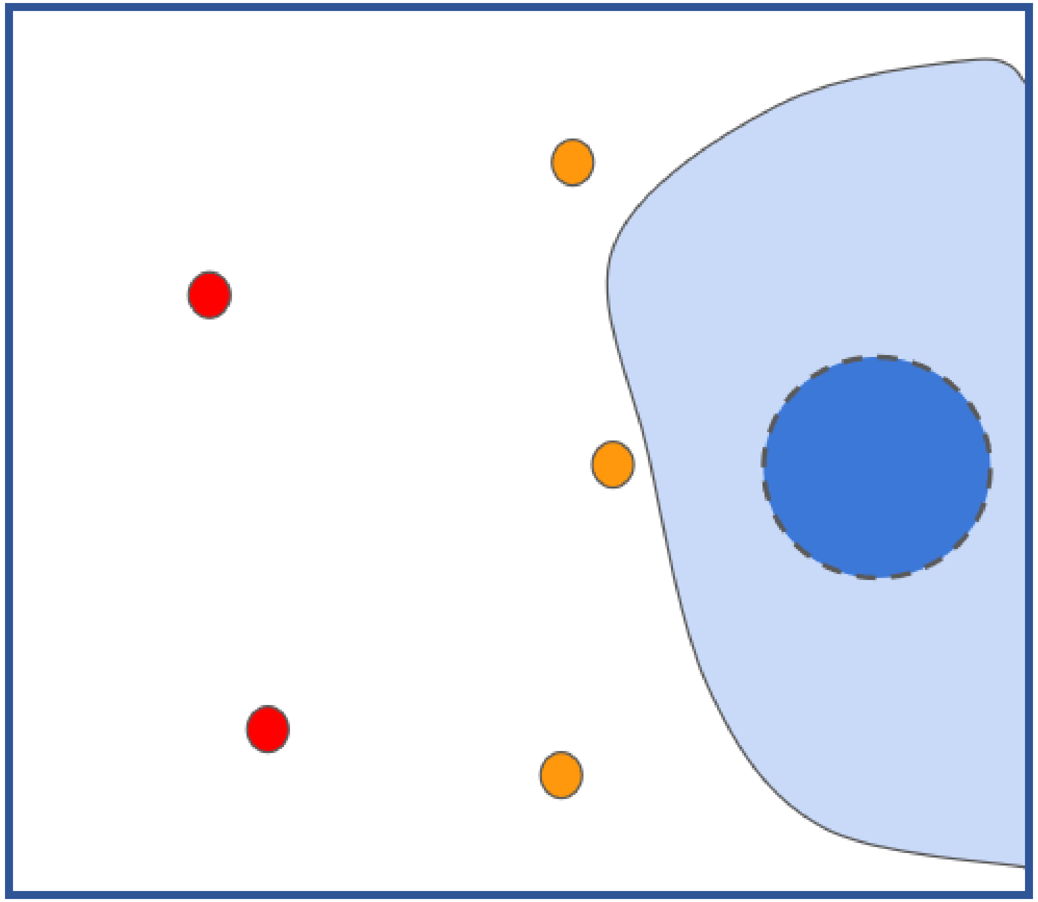
Domain Estimation Algorithm (DEA)



Sample a new point accordingly to:

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

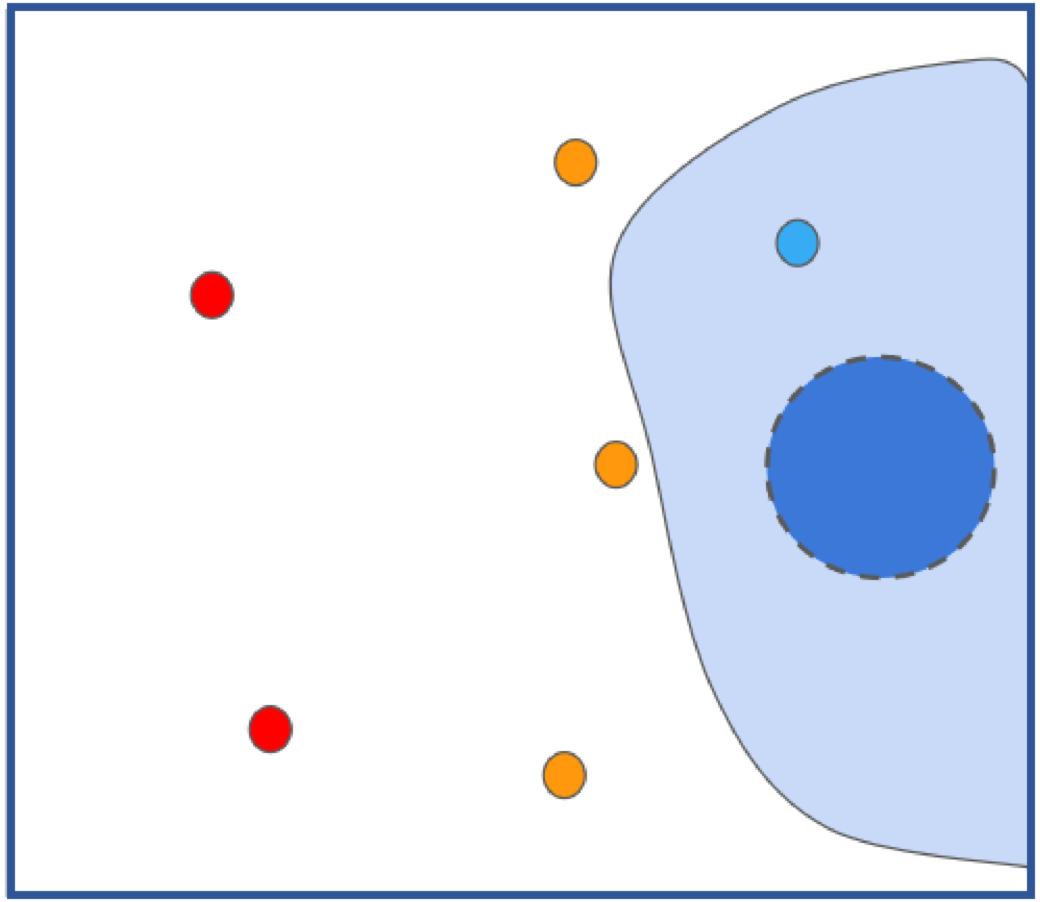
Domain Estimation Algorithm (DEA)



Sample a new point accordingly to:

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

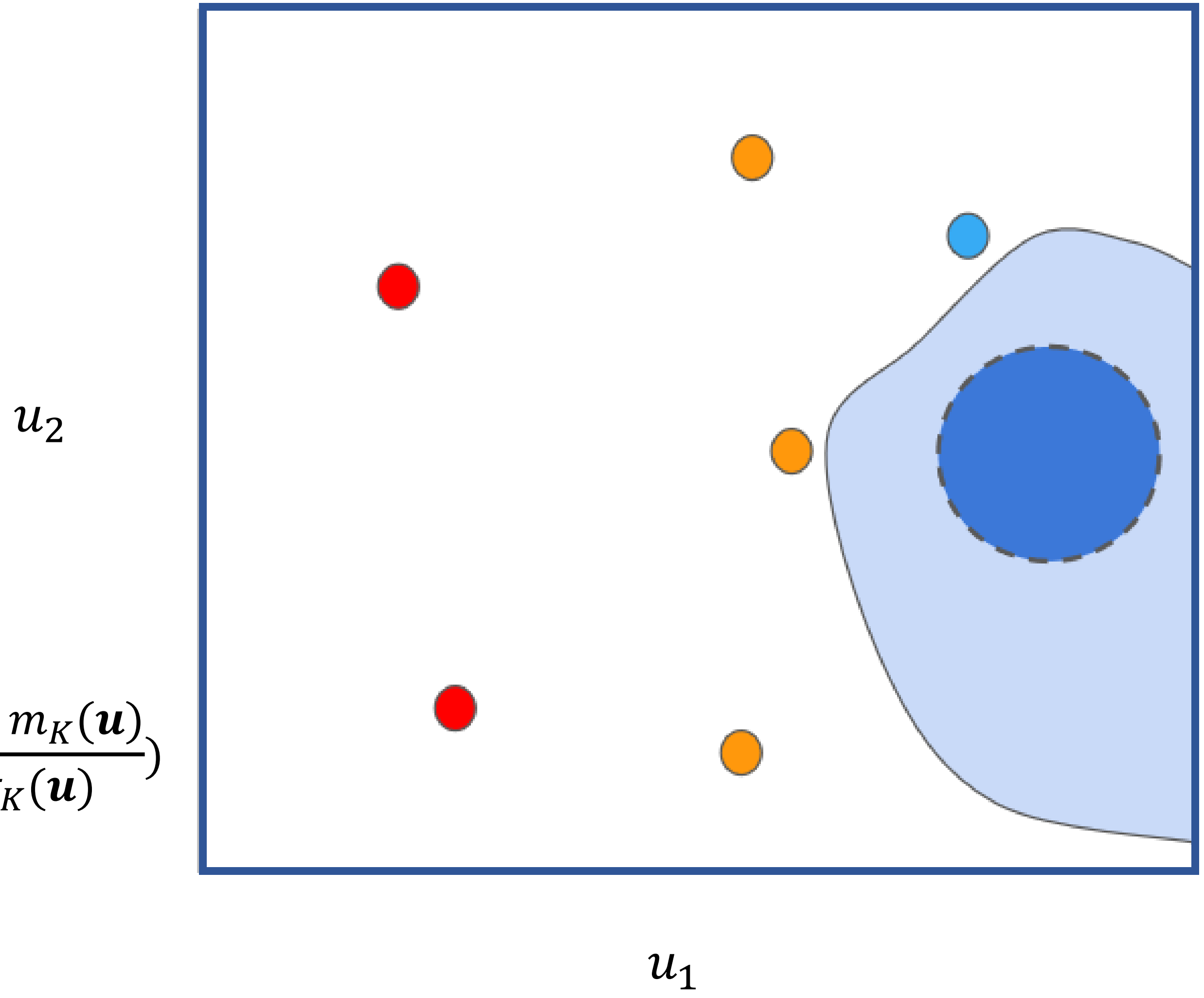
Domain Estimation Algorithm (DEA)



Sample a new point accordingly to:

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

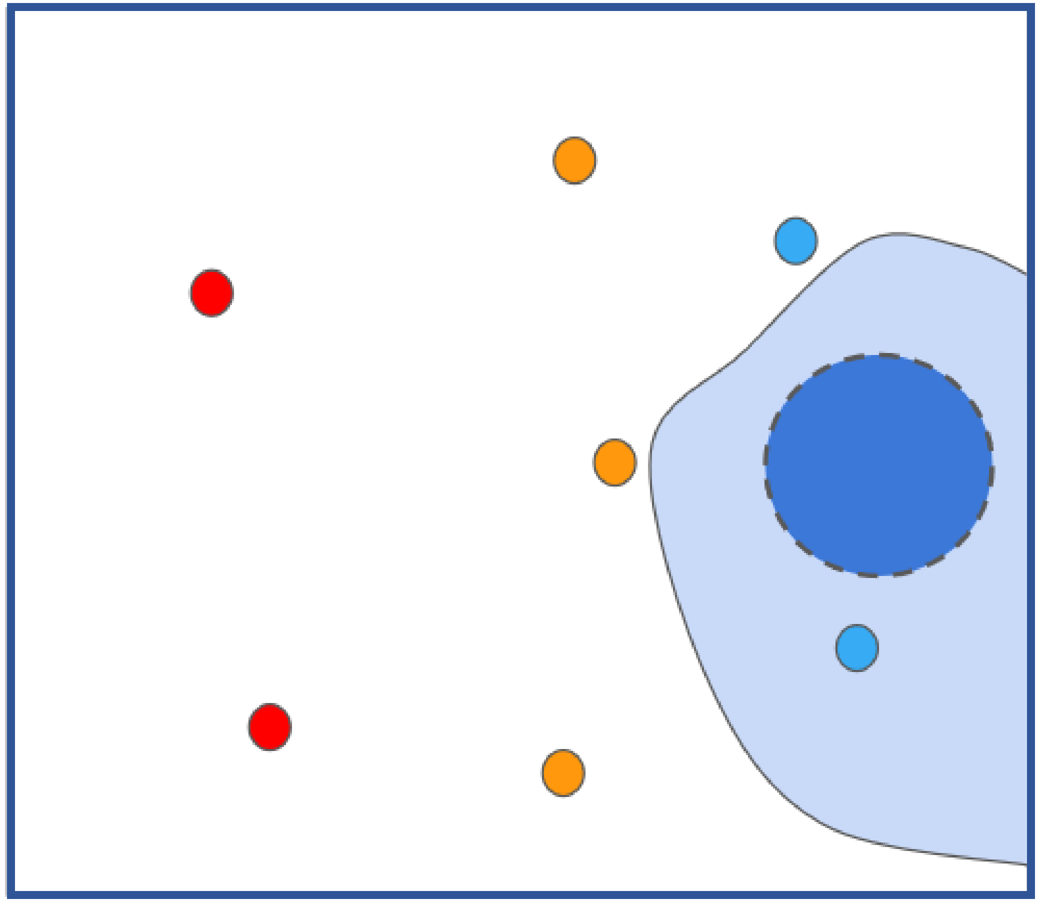
Domain Estimation Algorithm (DEA)



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)



u_2

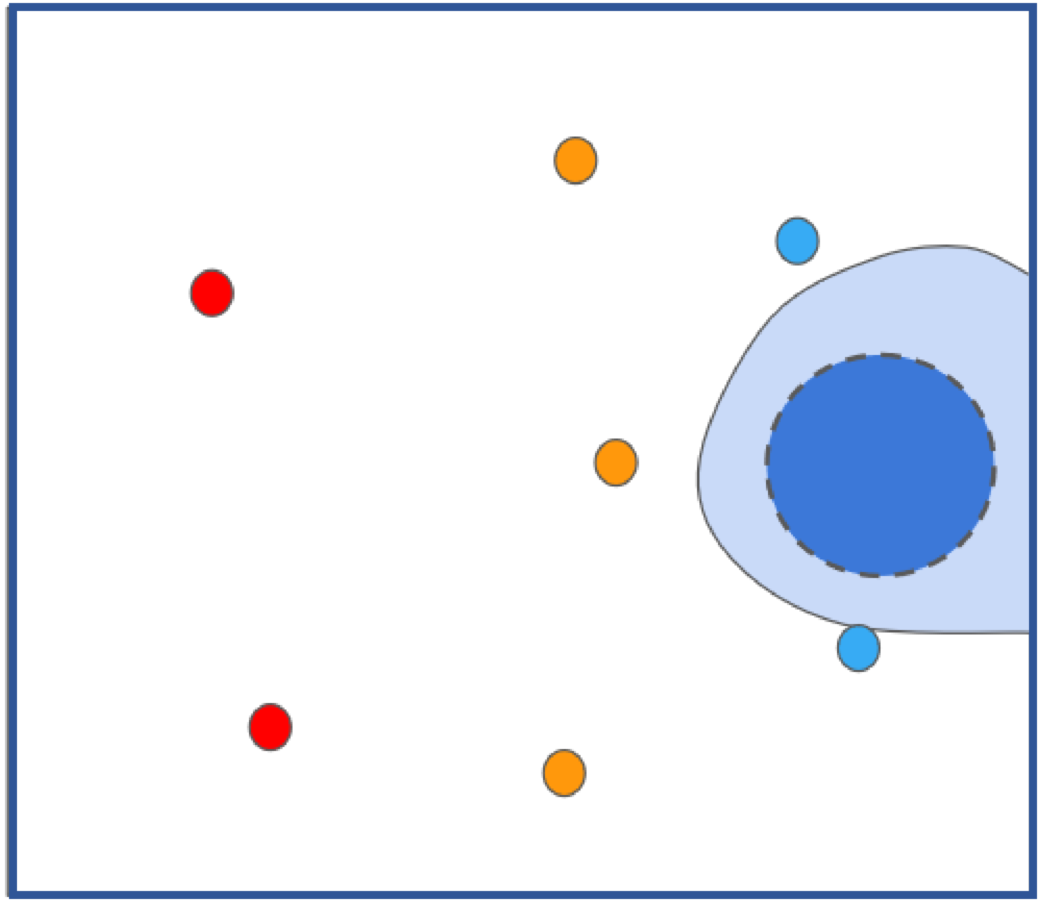
u_1

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



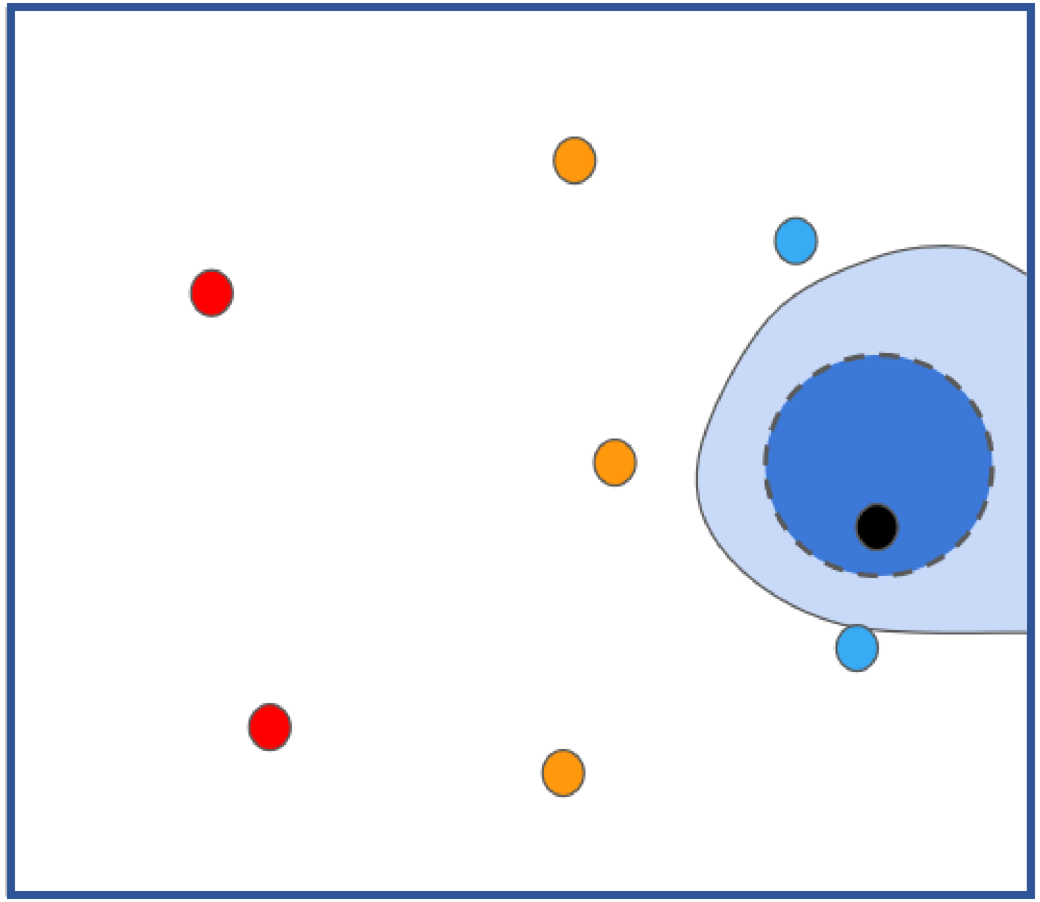
u_1

Sample a new point accordingly to:

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

Domain Estimation Algorithm (DEA)

u_2



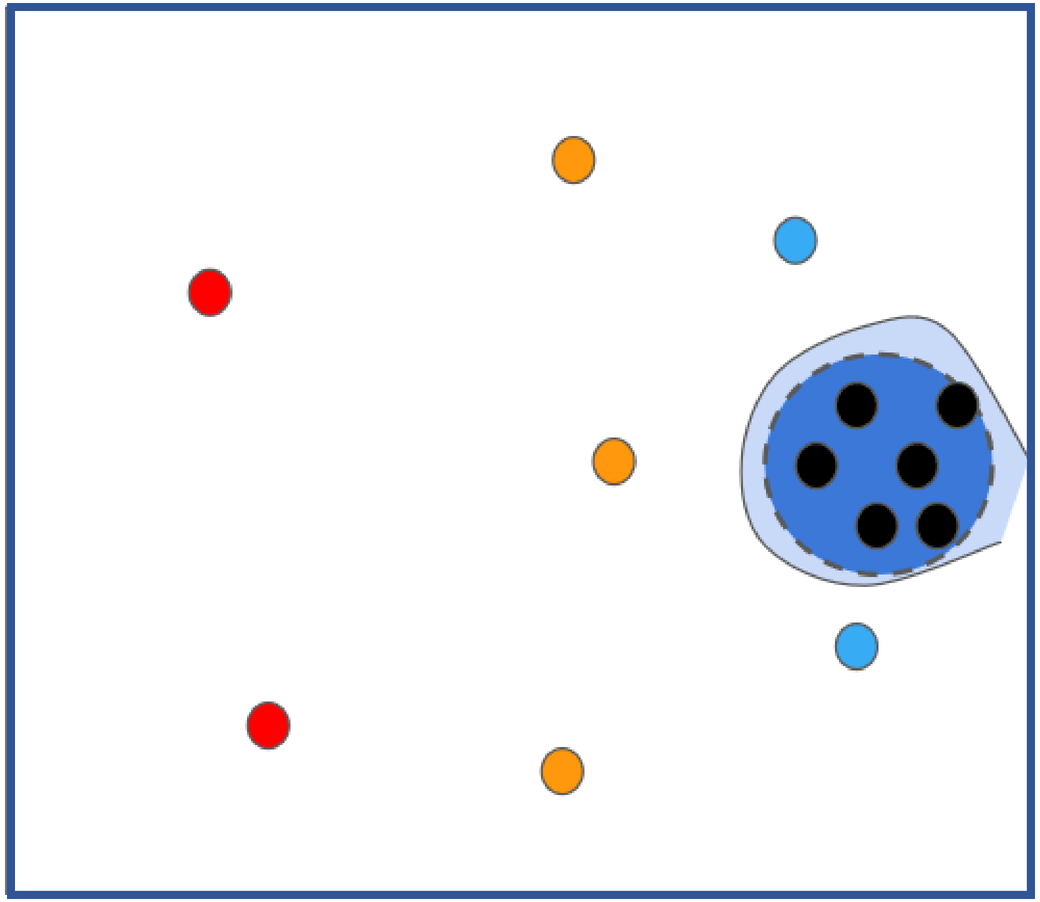
u_1

Sample a new point accordingly to:

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

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 Problem

Finding the trajectories which falsify the requirements, finding $\boldsymbol{u} \in B$

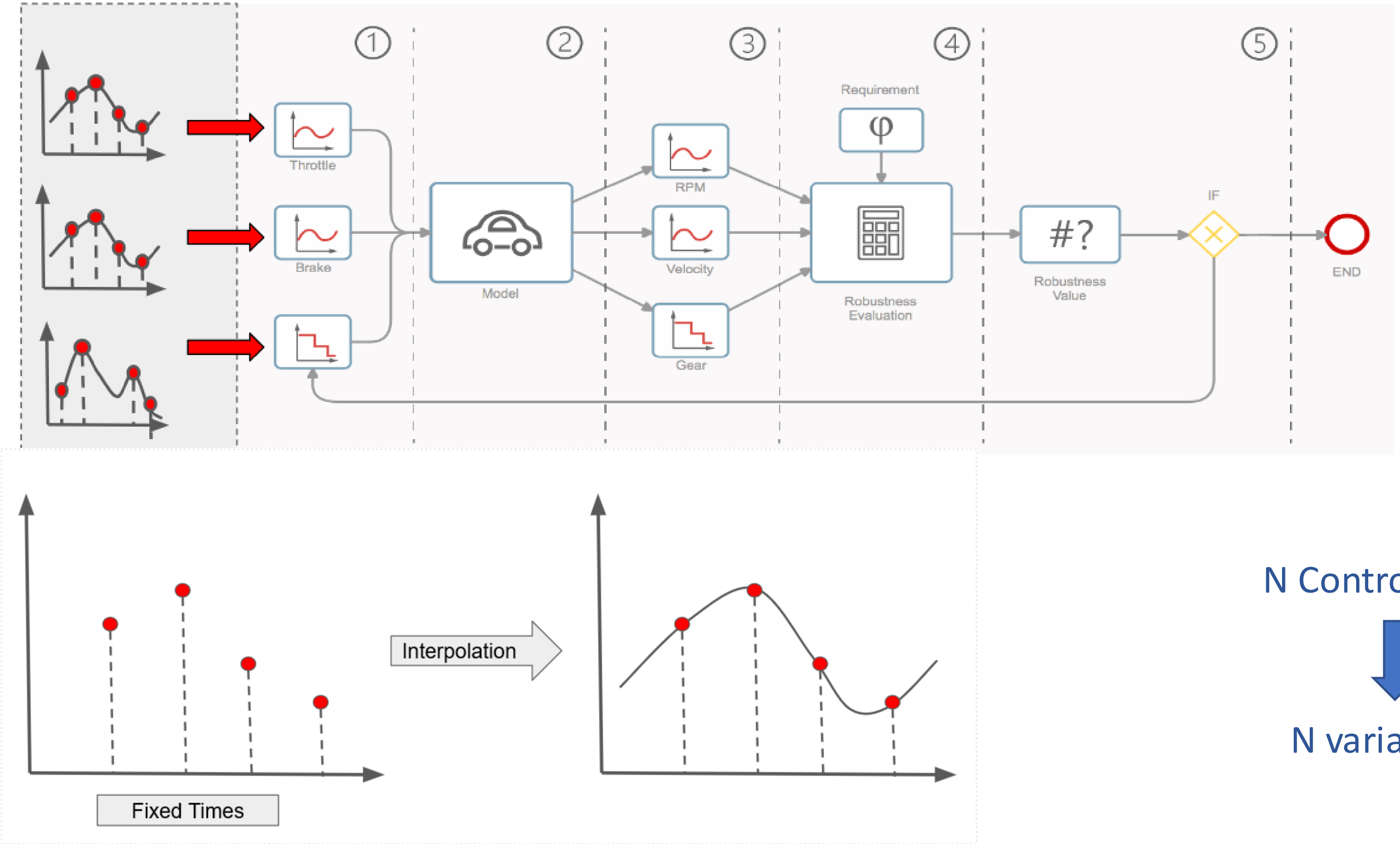
$$\mathrm{B} = \{\boldsymbol{u} \in \mathrm{U} \mid \rho(\phi, \boldsymbol{u}, 0) < 0\} \subseteq U$$

We call B the counterexample set and its elements counterexamples

If B is empty then $\rho(\phi, \boldsymbol{u}, 0) \geq 0$

Solving the domain estimation problem could be extremely difficult because of the infinite dimensionality of the input space, which is a space of functions

Finite Parameterization



N Control points

N variable

Domain Estimation Problem

Finding the trajectories which falsify the requirements, finding $\hat{c} \in \hat{B}$

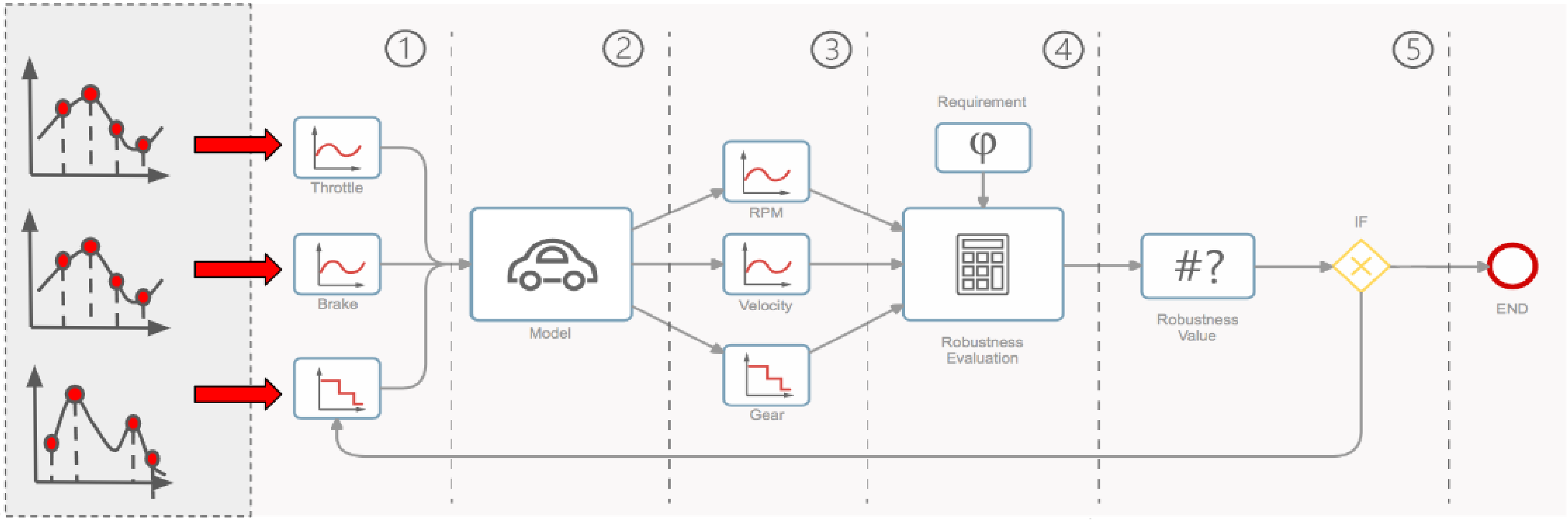
$$\hat{B} = \{ \hat{c} \in U_{n_1} \times \cdots \times U_{n_{|U|}} \mid \rho(\phi, P_{\boldsymbol{n}}(\hat{c}), 0)) < 0 \}$$

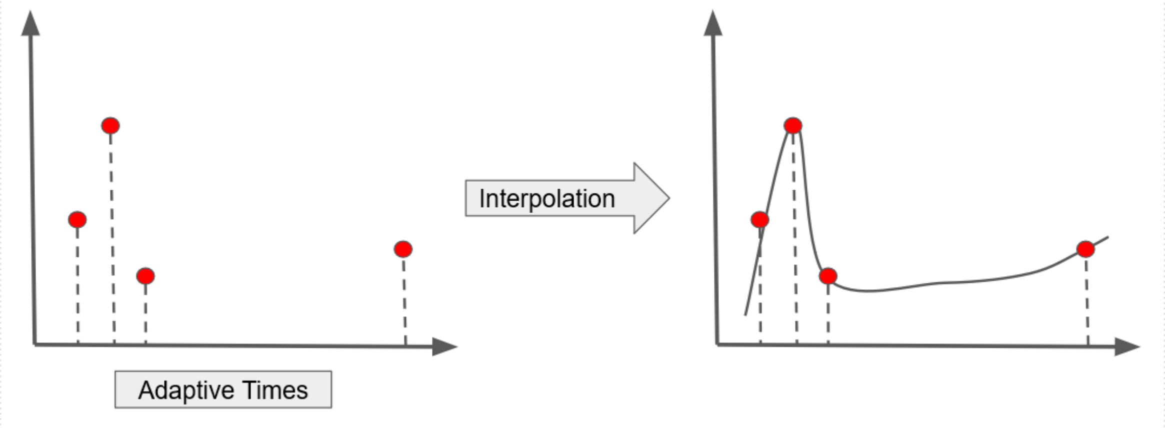
Where $c_k = \{(t_1^k, u_{n_k}^k), \ldots, (t_{n_k}^k, u_{k_n})\}$ and $P_{\boldsymbol{n}} = (P_{n_1}, \ldots, P_{n_{|U|}})$

Piecewise linear or polynomial functions are known to be dense in the space of continuous functions!

Then, B has at least one element $\Longleftrightarrow \exists n \in \omega^{|U|}$, $\hat{B}$ has at least one element.

Adaptive Parameterization



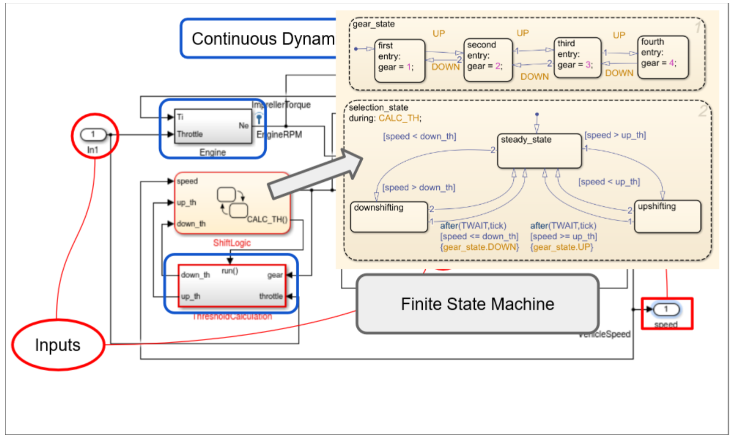


N Control points

2N variable

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)



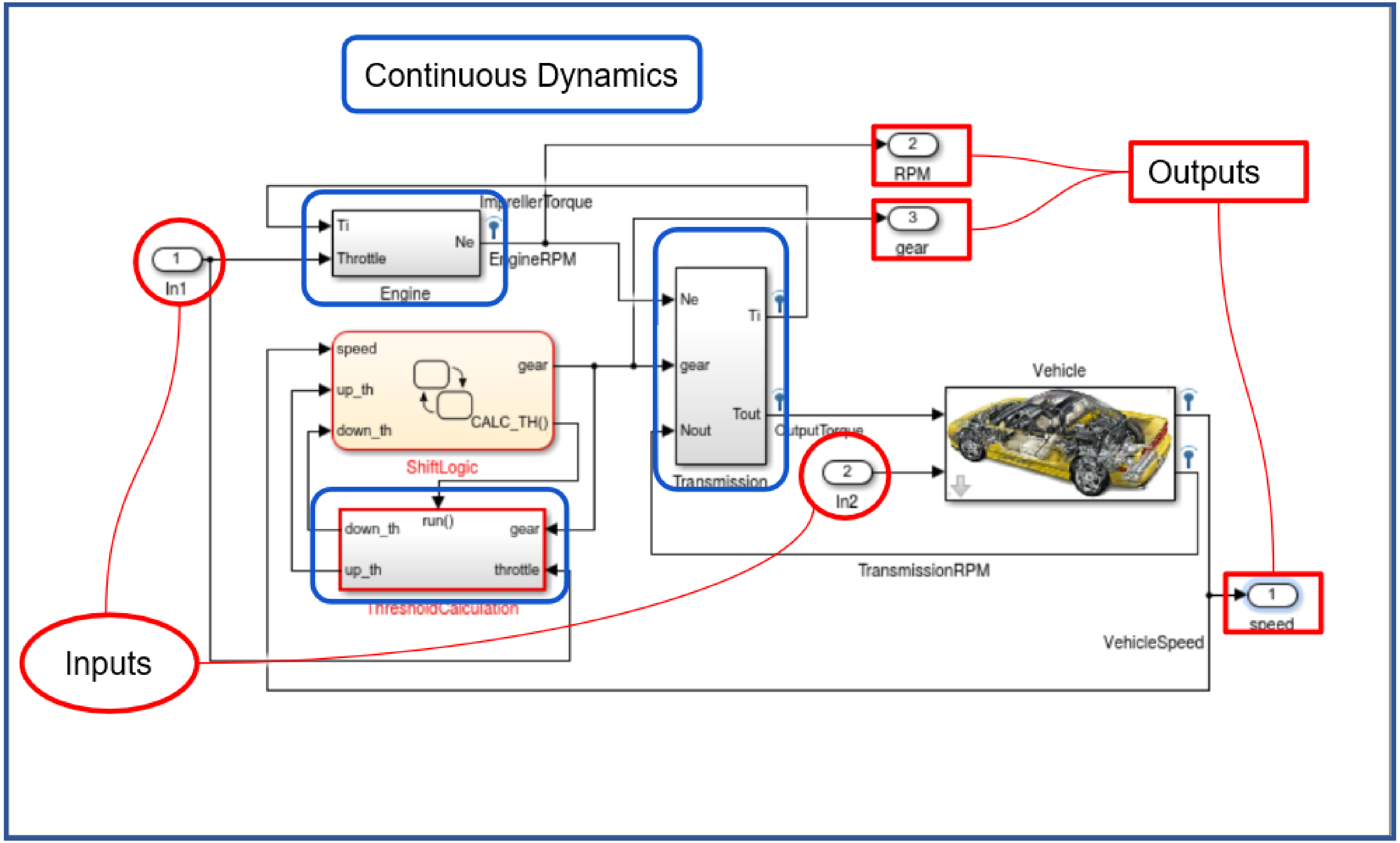
	Adaptive DEA		Adaptive GP-UCB		S-TaLiRo		
Req	nval	times	nval	times	nval	times	Alg
ϕ_1	$\mathbf{4.42 \pm 0.53}$	2.16 ± 0.61	$\mathbf{4.16 \pm 2.40}$	0.55 ± 0.30	5.16 ± 4.32	0.57 ± 0.48	UR
ϕ_1	$\mathbf{6.90 \pm 2.22}$	5.78 ± 3.88	8.7 ± 1.78	1.52 ± 0.40	39.64 ± 44.49	4.46 ± 4.99	SA
ϕ_2	$\mathbf{3.24 \pm 1.98}$	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

Model

Inputs:

Throttle

Brake



Outputs:

RPM

Gear

Speed

https://it.mathworks.com/help/simulink/slref/modeling-an-automatic-transmission-controller.html

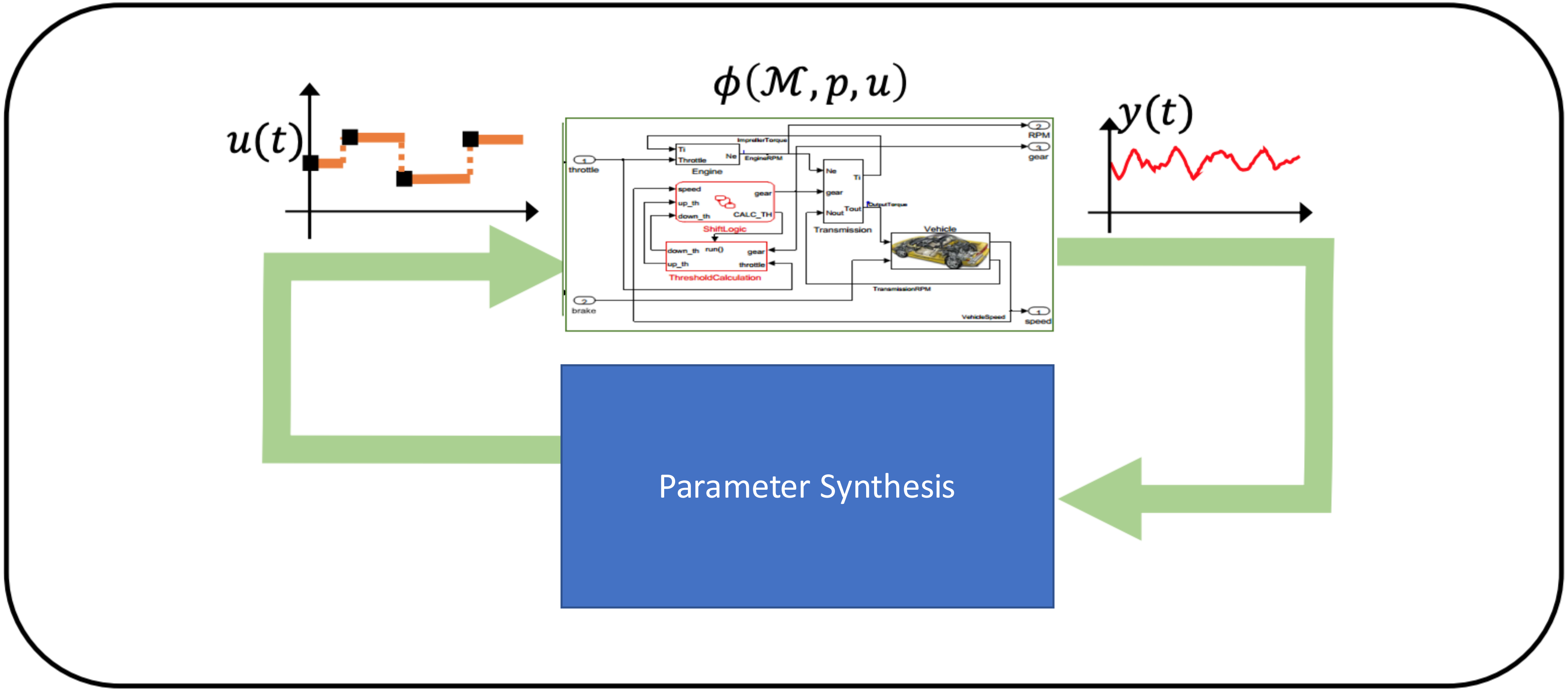
Bibliography

Falsification:

- 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.
- 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
- 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)
- 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.

- Parameter synthesis

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

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

Gaussian Process Regression

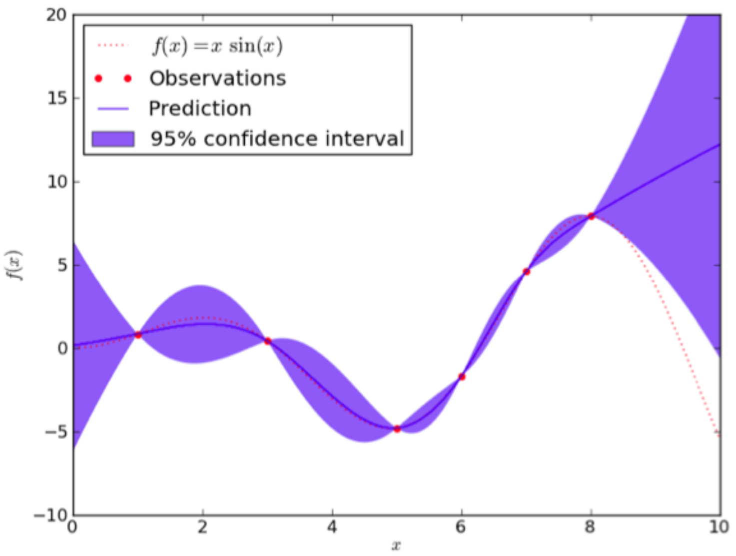
Gaussian Processes can be used for Bayesian prediction and classification tasks.

Idea: put a **GP prior** on functions; condition on **observed data (training set)** (x_i, y_i); we compute a **posterior** distribution on functions; make **predictions**.

Latent function: f , GP ; **Noise model**: $p(y_i|f(x_i))$

Prediction (latent function f^* at x^*)

$$p(f^*|\mathbf{y}) \propto \int df(\mathbf{x})p(f^*, f(\mathbf{x}))p(\mathbf{y}|f(\mathbf{x}))$$



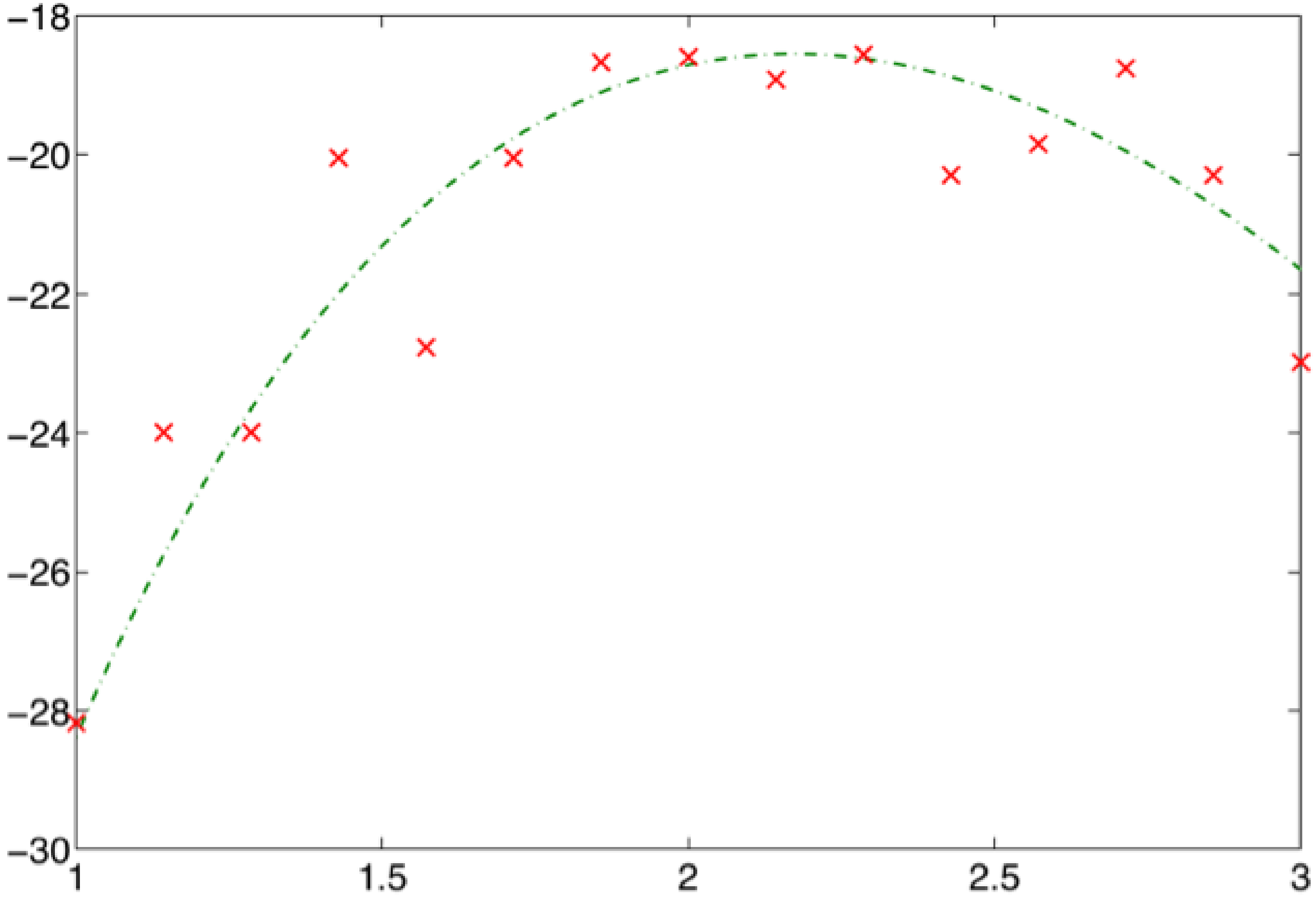
Under Gaussian noise $y(\mathbf{x}) = f(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2)$
predictions have an analytic expression.

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X)+\sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right)$$

$$\begin{aligned} \mathbf{f}_*|X,\mathbf{y},X_* &\sim \mathcal{N}(\bar{\mathbf{f}}_*, \operatorname{cov}(\mathbf{f}_*)), \quad \text{where} \\ \bar{\mathbf{f}}_* &\triangleq \mathbb{E}[\mathbf{f}_*|X,\mathbf{y},X_*] = K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}\mathbf{y}, \\ \operatorname{cov}(\mathbf{f}_*) &= K(X_*,X_*) - K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}K(X,X_*) \end{aligned}$$

(1) Sample

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

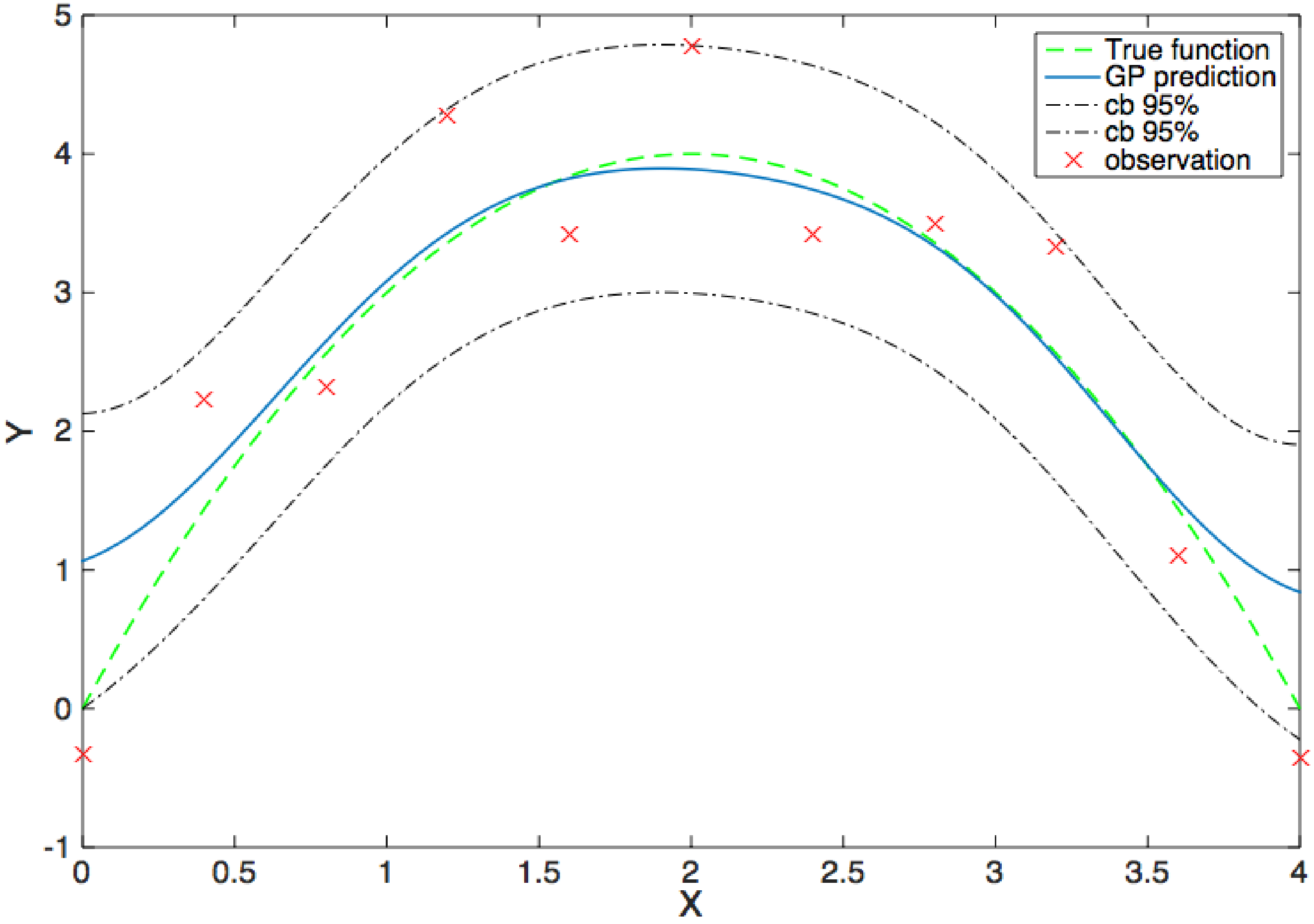


(2) The GP Regression

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

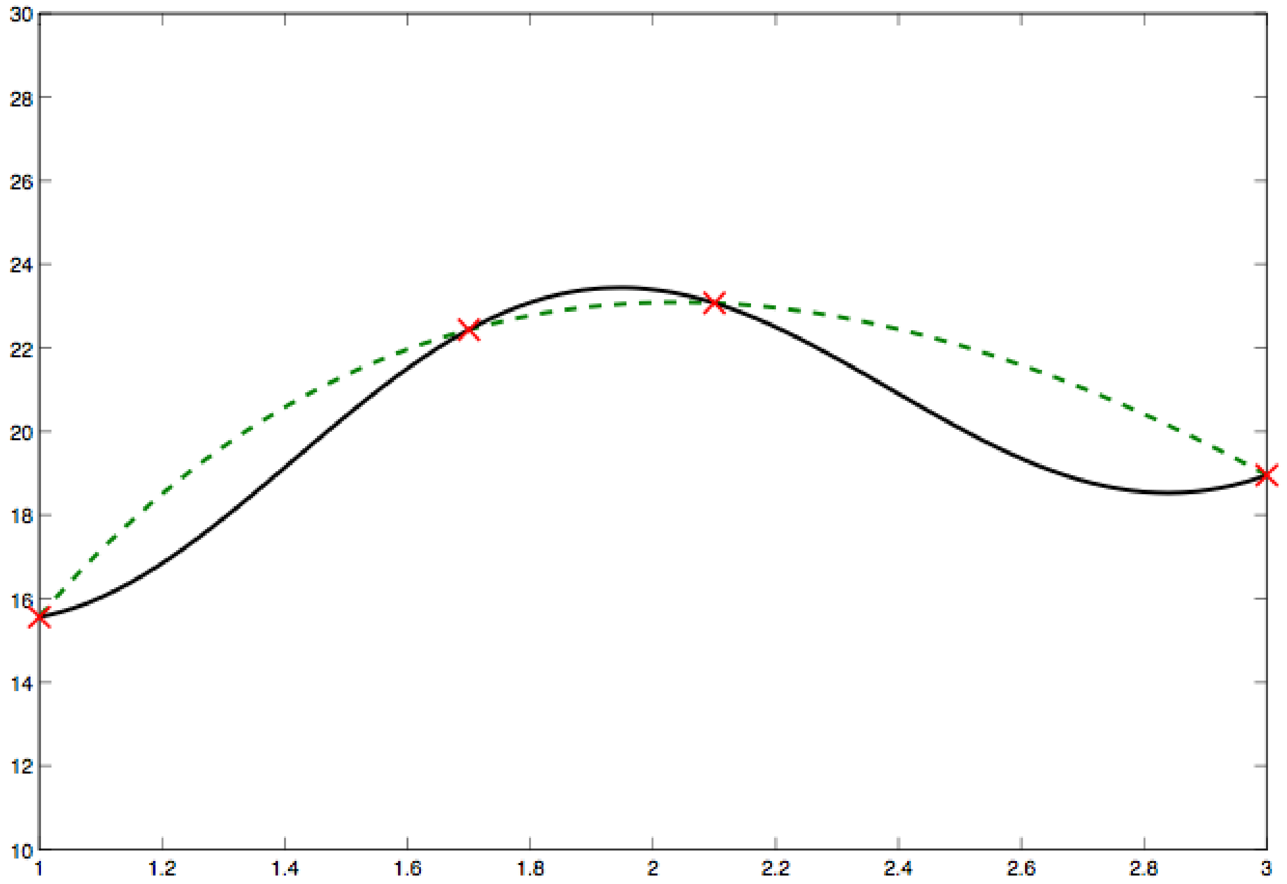
(2) The GP Regression

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



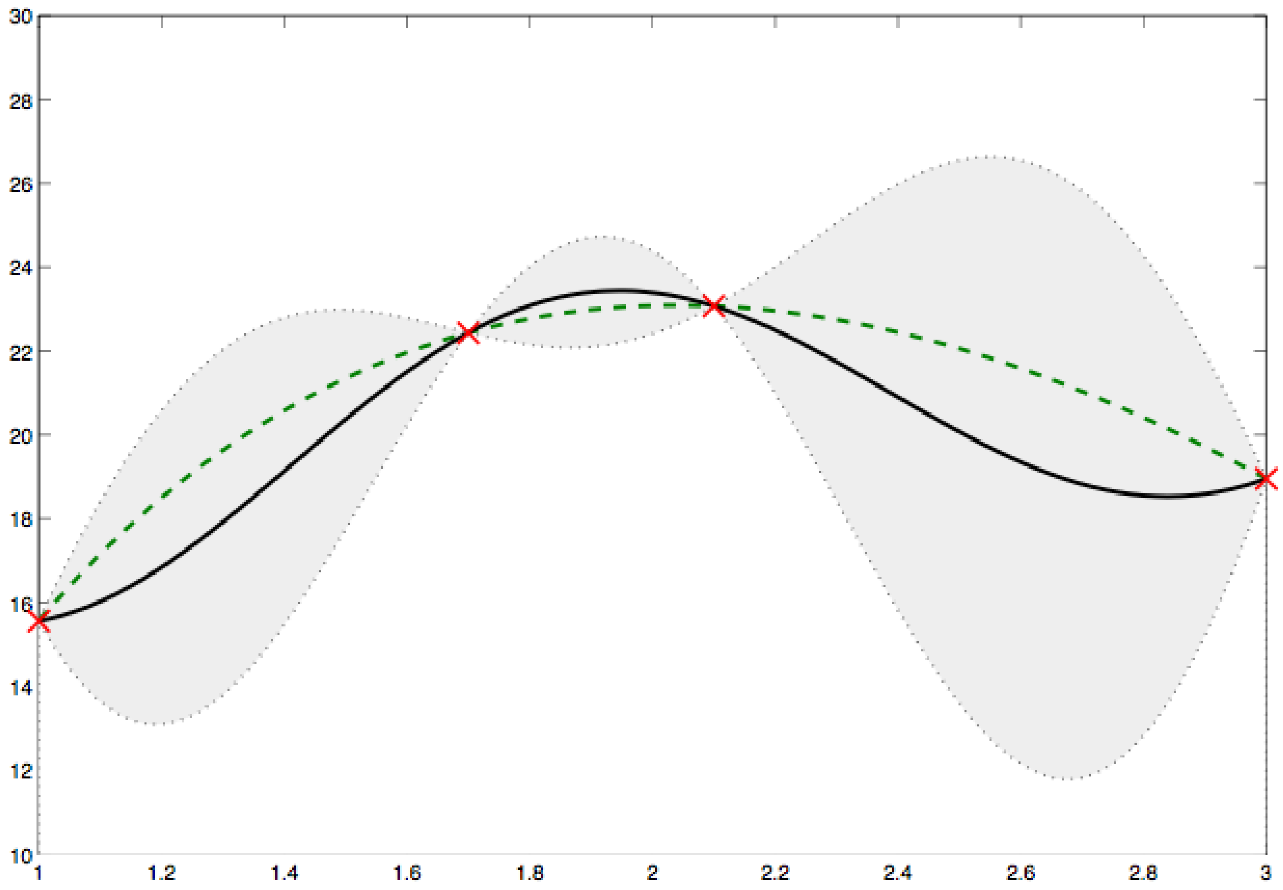
(3) The GP-UCB Algorithm

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



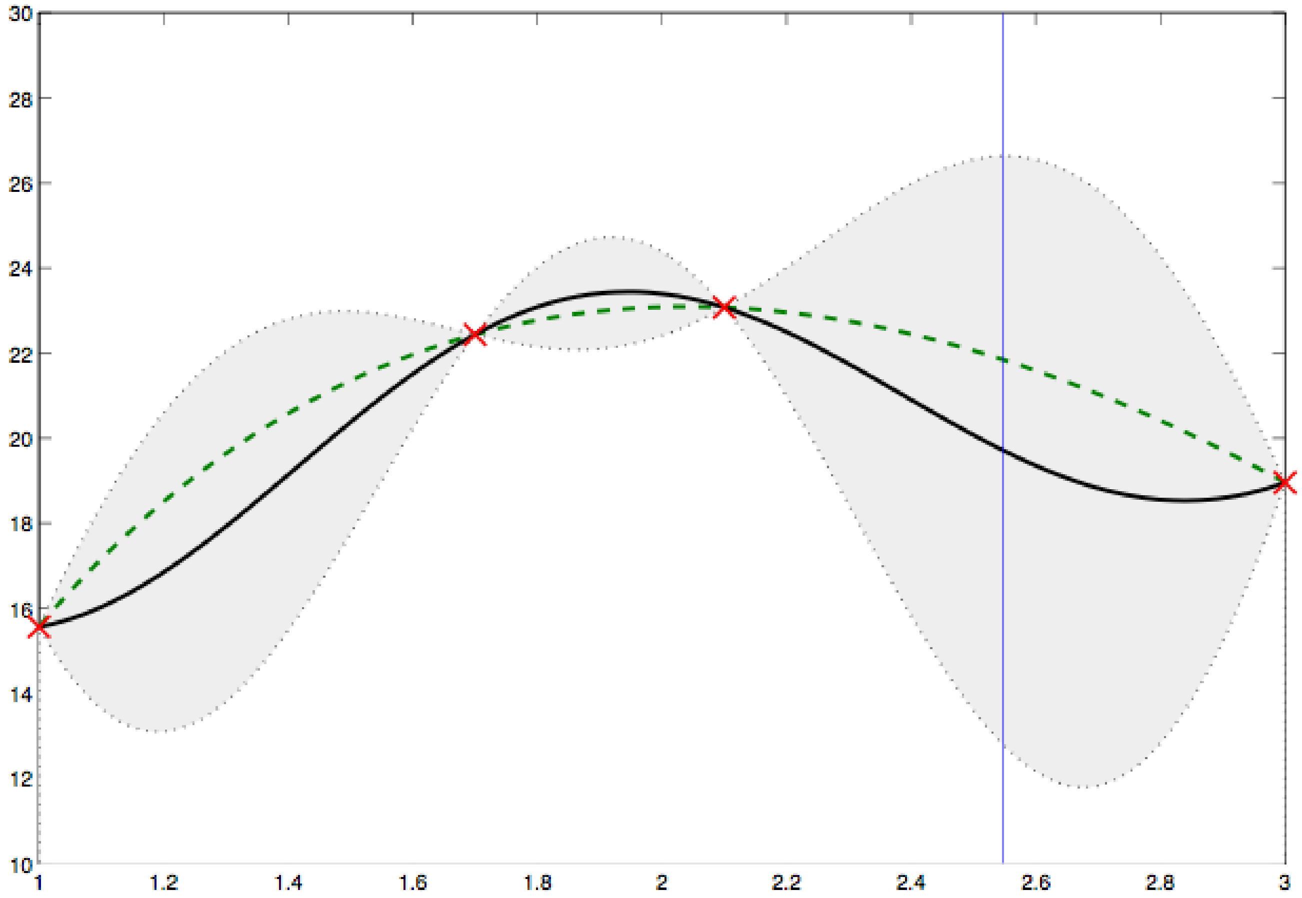
(3) The GP-UCB Algorithm

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



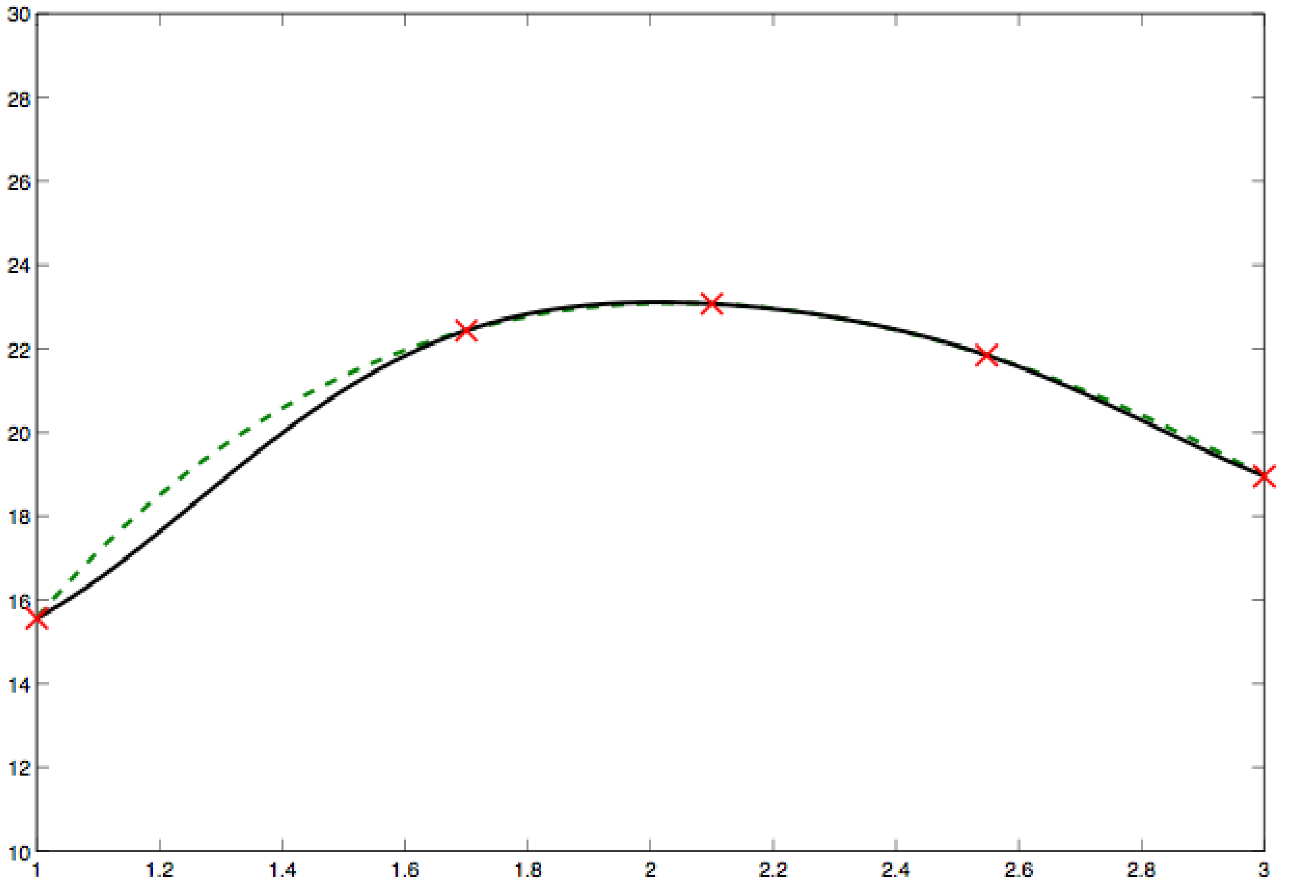
(3) The GP-UCB Algorithm

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



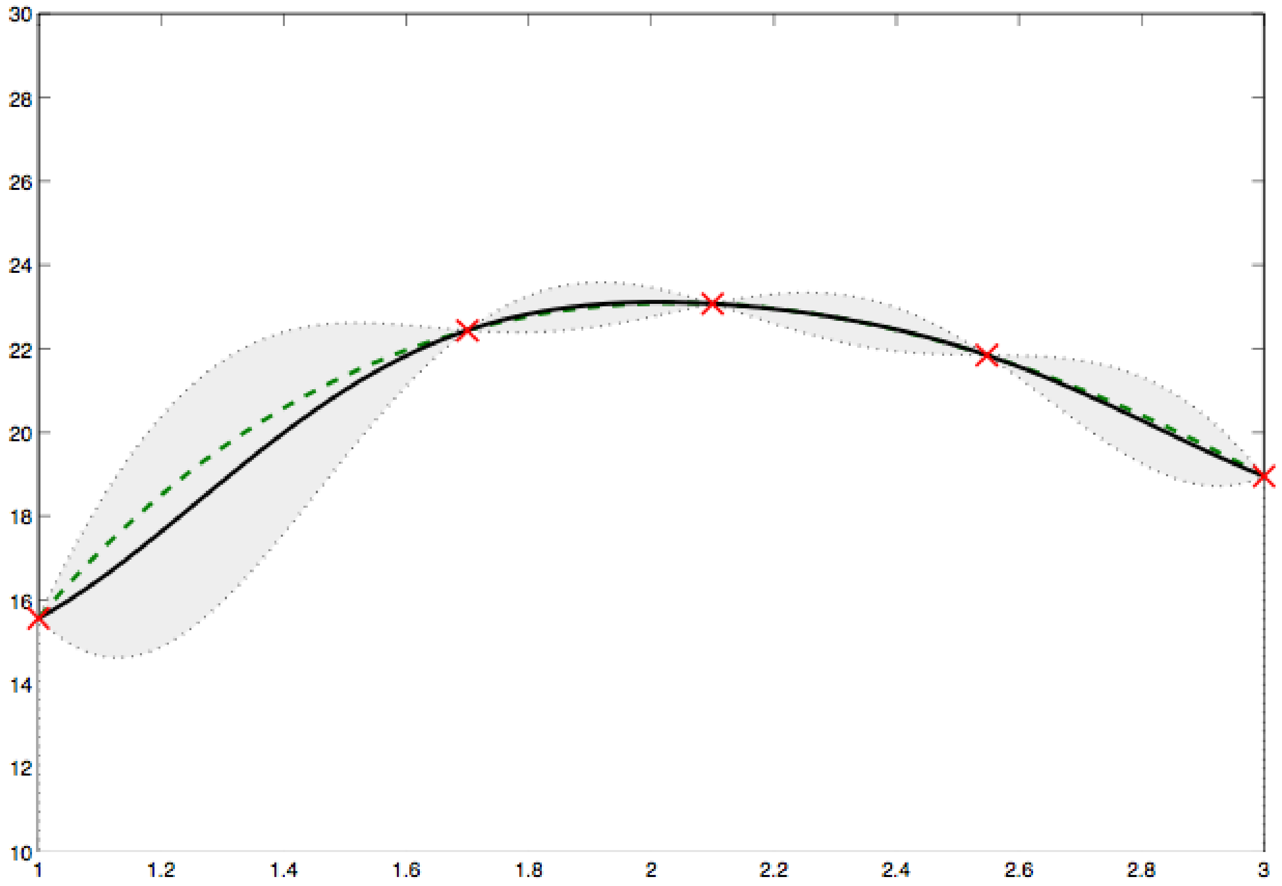
(3) The GP-UCB Algorithm

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



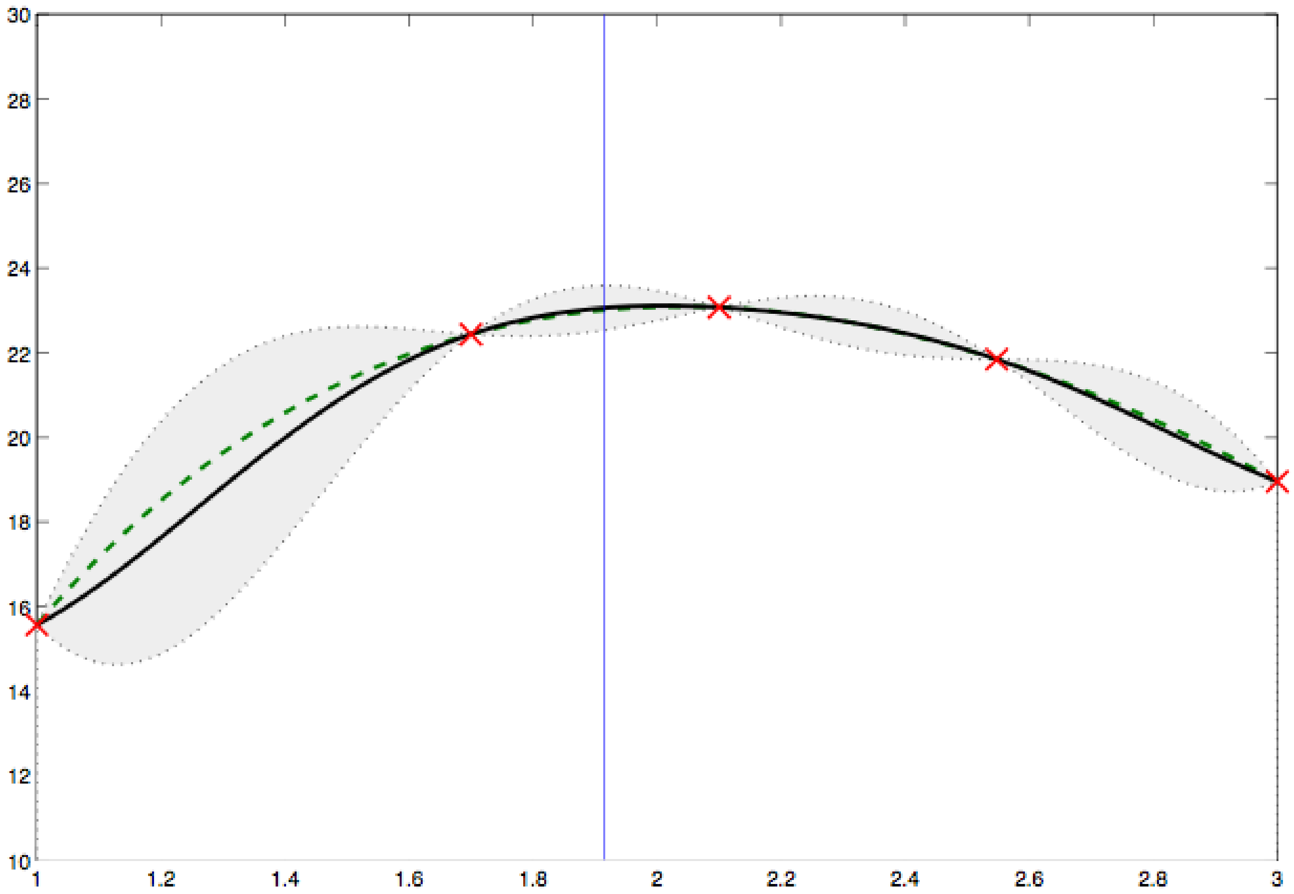
(3) The GP-UCB Algorithm

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**: $\theta_{t+1} = argmax_{\theta}[\mu^*(\theta) + \beta_t \sqrt{k^*(\theta,\theta)}]$



Bibliography

Parameter Synthesis:

- Ezio Bartocci, Luca Bortolussi, Laura Nenzi, Guido Sanguinetti, System design of stochastic models using robustness of temporal properties. Theor. Comput. Sci. 587: 3-25 (2015)
- Bortolussi L., Silvetti S. (2018*) Bayesian Statistical Parameter Synthesis for Linear Temporal Properties of Stochastic Models*. TACAS 2018. LNCS, vol 10806. Springer, Cham