Calibration of complex system models for the construction of the digital twin of the autonomous vehicle

Soutenance de thèse de Doctorat en Mathématiques appliquées

Soutenue publiquement par Clara CARLIER
le 30 septembre 2024

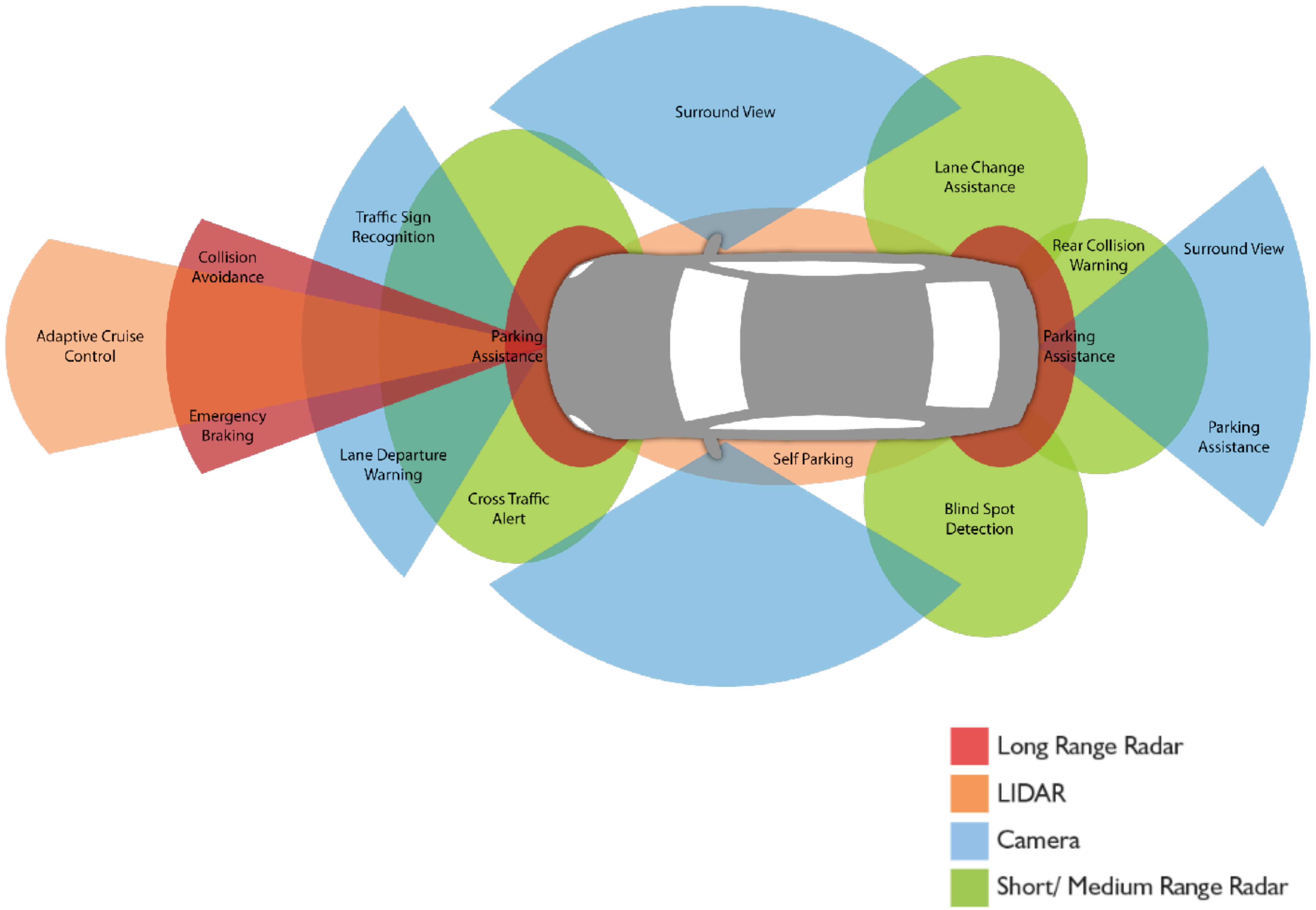
Directeurs de thèse
Matthieu LERASLE (CREST, ENSAE, IP Paris)
Arnaud FRANJU (Groupe Renault)

1. General introduction

General introduction

Context

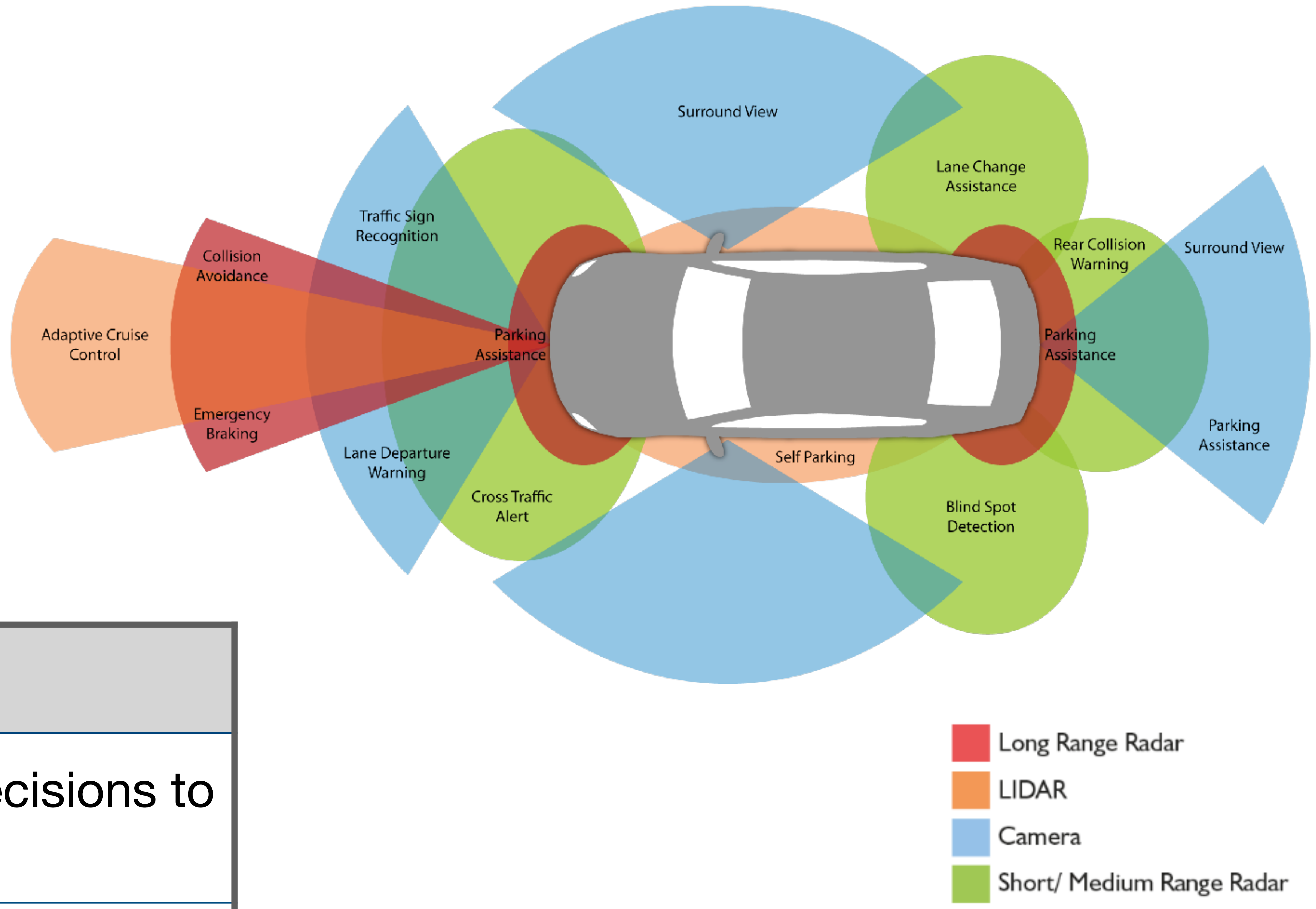
- Advanced driver assistance systems (ADAS) are becoming increasingly important and complex around the world



General introduction

Context

- Advanced driver assistance systems (ADAS) are becoming increasingly important and complex around the world

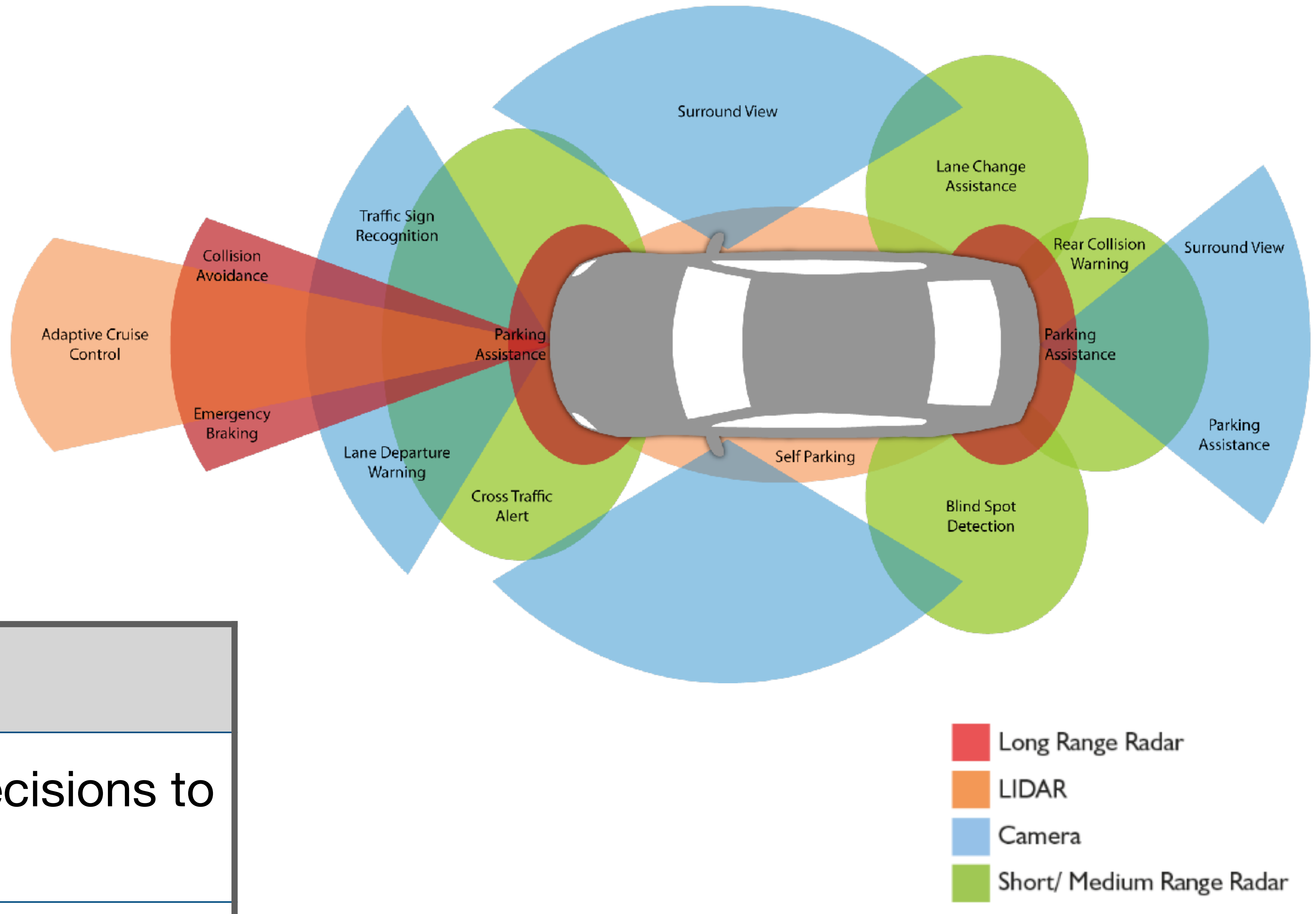


Passive ADAS	**Active ADAS**
Provide drivers with information to help them make safer decisions	Autonomously making decisions to prevent accidents
• forward collision warning • lane change assist	• **automatic emergency braking** • adaptive cruise control

General introduction

Context

- Advanced driver assistance systems (ADAS) are becoming increasingly important and complex around the world



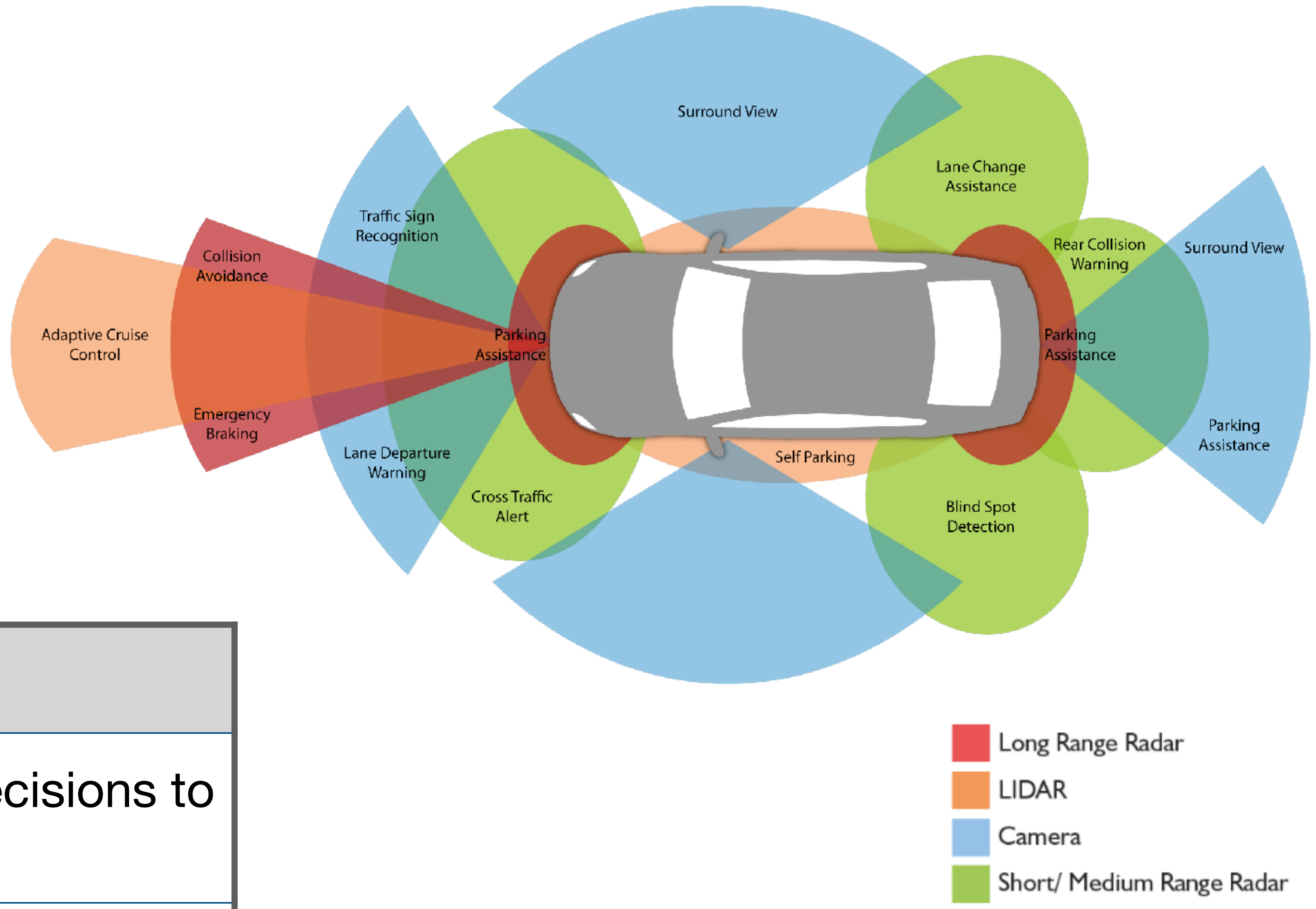
Passive ADAS	**Active ADAS**
Provide drivers with information to help them make safer decisions	Autonomously making decisions to prevent accidents
• forward collision warning • lane change assist	• **automatic emergency braking** • adaptive cruise control

- Some statistics
  - 96% of new vehicles are equipped with at least one ADAS function
  - 86% of people surveyed express doubts about their reliability

General introduction

Context

- Advanced driver assistance systems (ADAS) are becoming increasingly important and complex around the world



Passive ADAS	**Active ADAS**
Provide drivers with information to help them make safer decisions	Autonomously making decisions to prevent accidents
• forward collision warning • lane change assist	• **automatic emergency braking** • adaptive cruise control

- Some statistics
  - 96% of new vehicles are equipped with at least one ADAS function
  - 86% of people surveyed express doubts about their reliability

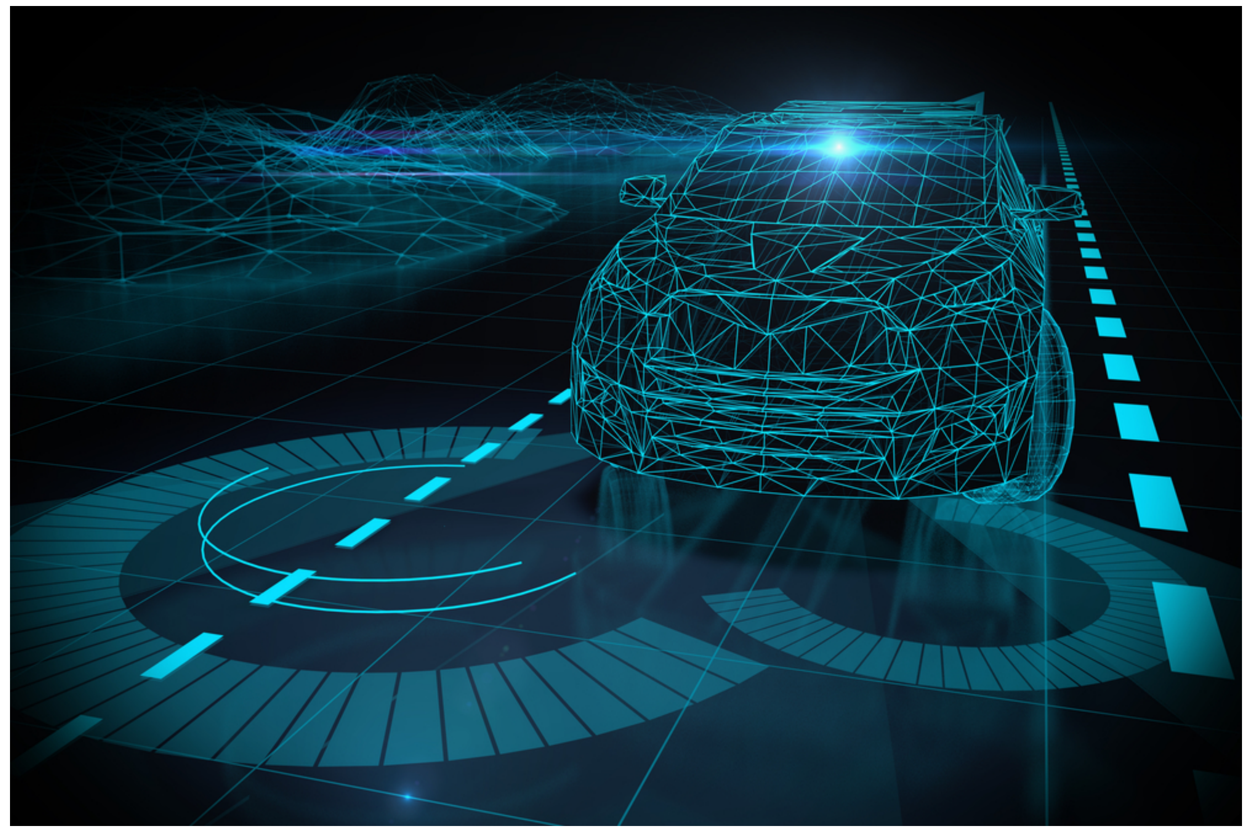
This underlines the importance of strict regulation of these technologies

General introduction

Context

Validation and certification of ADAS

- Real-life on-track experiments are costly and time-consuming
- Use simulated experiments to integrate them into the vehicle homologation process



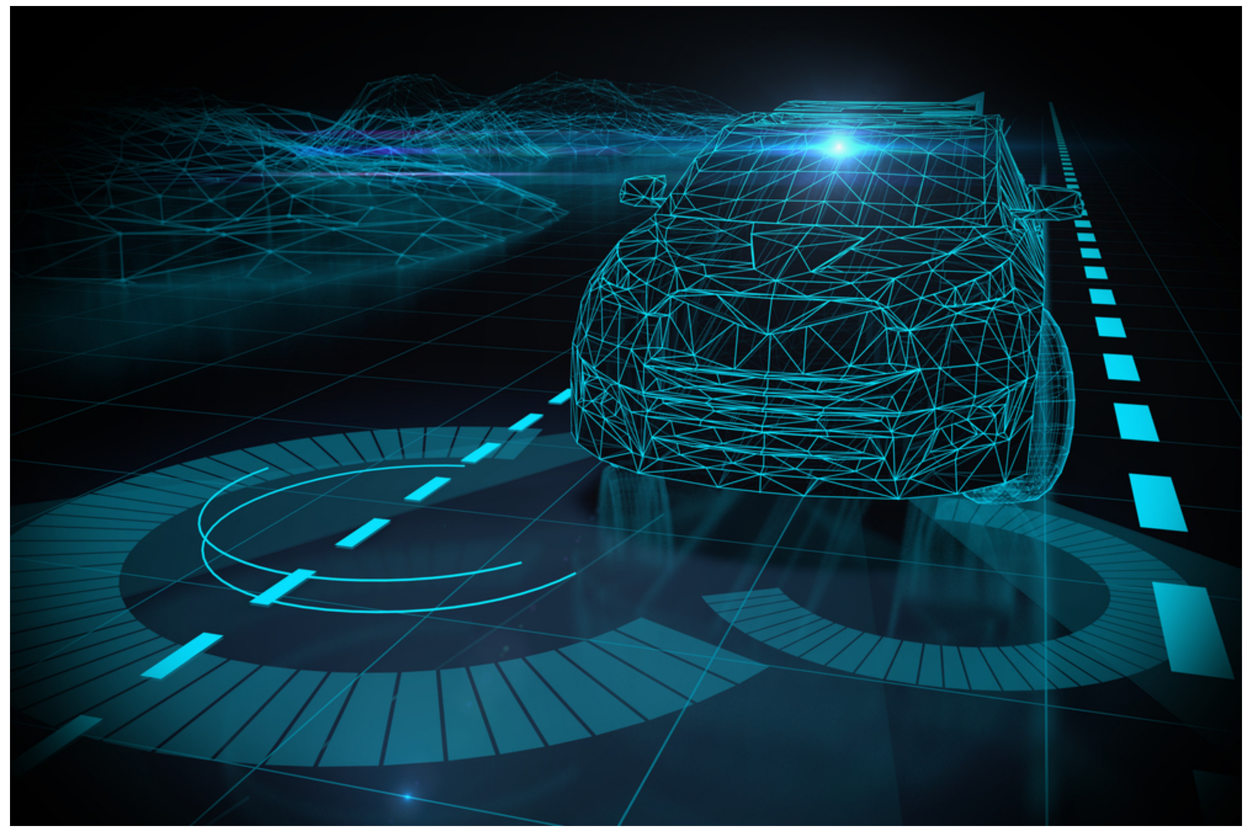
General introduction

Context

Validation and certification of ADAS

- Real-life on-track experiments are costly and time-consuming
- Use simulated experiments to integrate them into the vehicle homologation process

Are digital simulations sufficiently correlated with reality to be used legally?



General introduction

Context

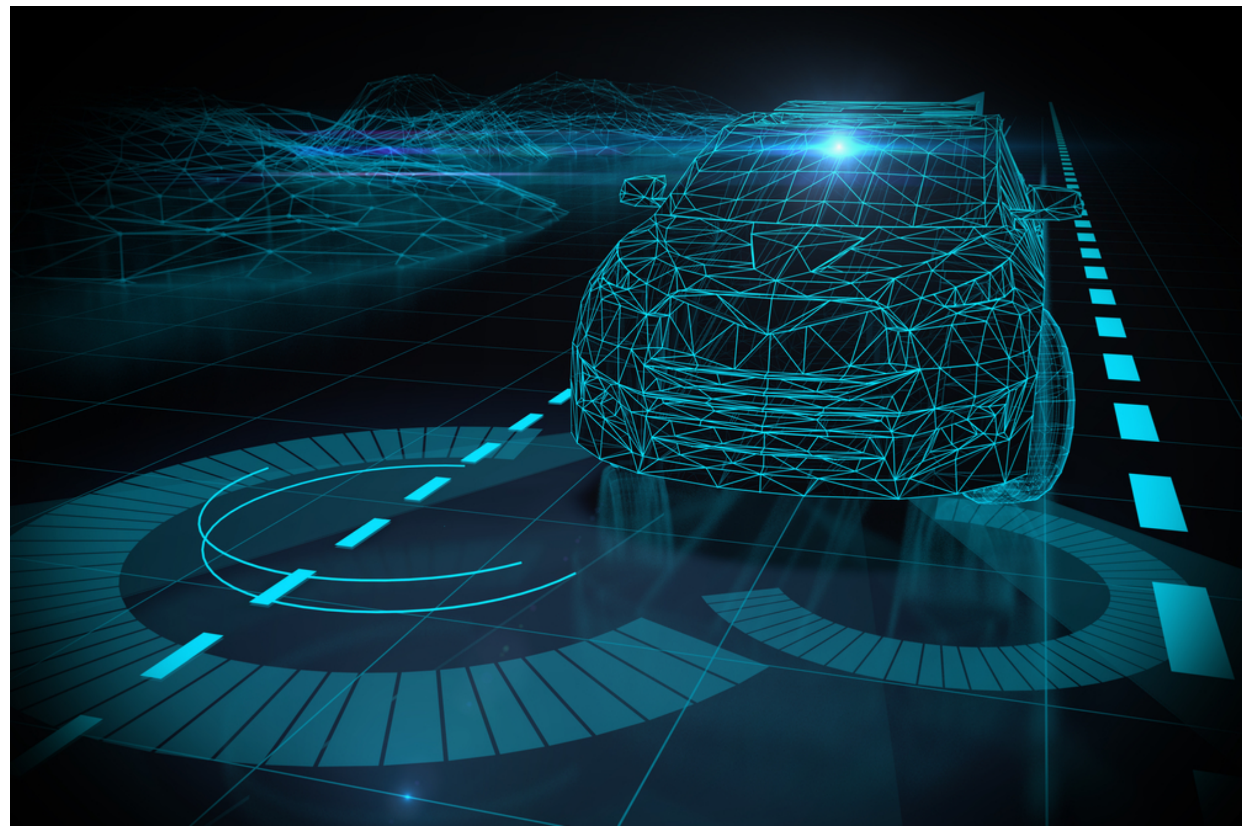
Validation and certification of ADAS

- Real-life on-track experiments are costly and time-consuming
- Use simulated experiments to integrate them into the vehicle homologation process

Are digital simulations sufficiently correlated with reality to be used legally?

Global objective: calibrate the simulator

- Simulate time series most correlated with reality by identifying the input parameters used to generate it
- Develop a methodology for gauging the quality of simulations and modifying the input parameters to improve correlation automatically



General introduction

How the simulator works

Two primary types of experiments are used

- **Simulated ones:** generated using the digital simulator, offering great flexibility in experimental design
- **Real on-track ones:** physical tests performed on track, high cost and infrequently performed

Simulated time series

Reference on-track time series

General introduction

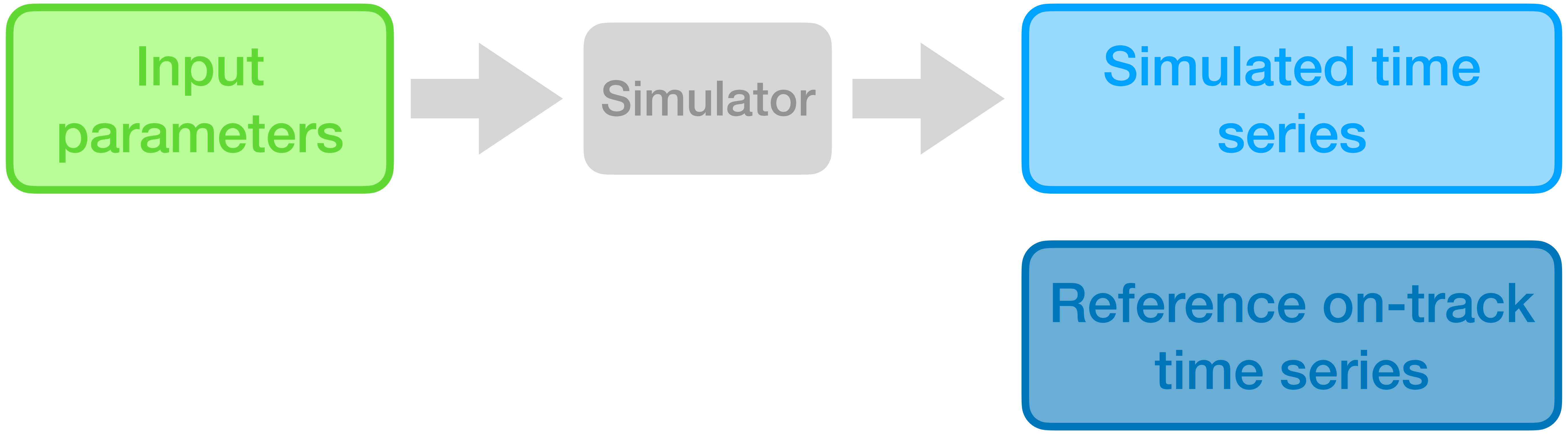
How the simulator works

Two primary types of experiments are used

- **Simulated ones:** generated using the digital simulator, offering great flexibility in experimental design
- **Real on-track ones:** physical tests performed on track, high cost and infrequently performed

A **scenario** is defined as a combination of **two main components**

- **Input parameters:** establish the necessary conditions for conducting the simulations
- **Output time series:** capture the dynamic of the vehicles throughout the experience



General introduction

How the simulator works

Two primary types of experiments are used

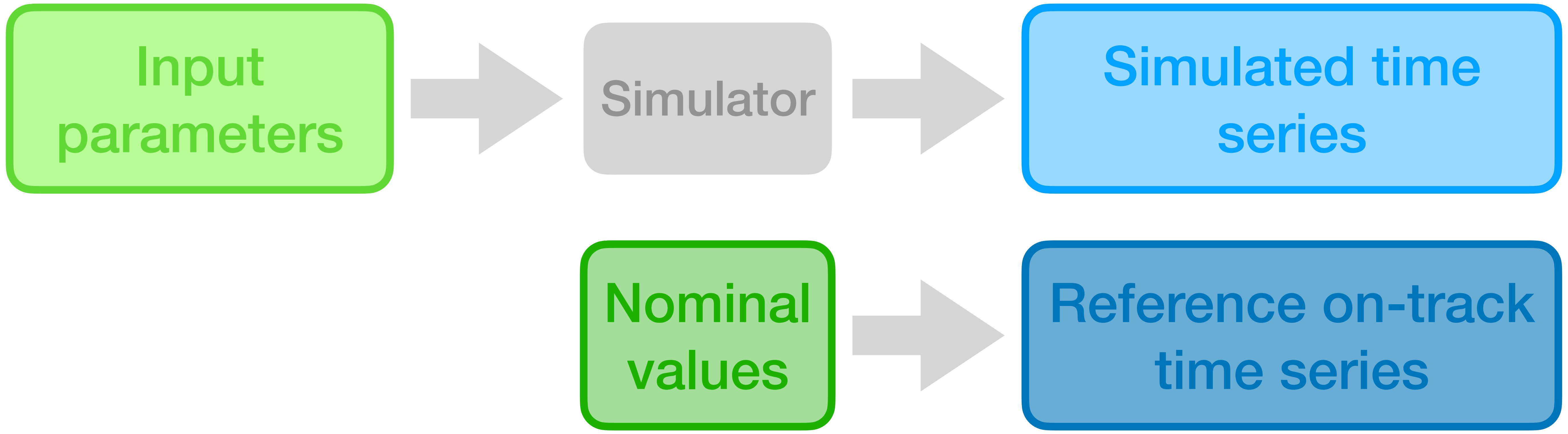
- **Simulated ones:** generated using the digital simulator, offering great flexibility in experimental design
- **Real on-track ones:** physical tests performed on track, high cost and infrequently performed

A **scenario** is defined as a combination of **two main components**

- **Input parameters:** establish the necessary conditions for conducting the simulations
- **Output time series:** capture the dynamic of the vehicles throughout the experience

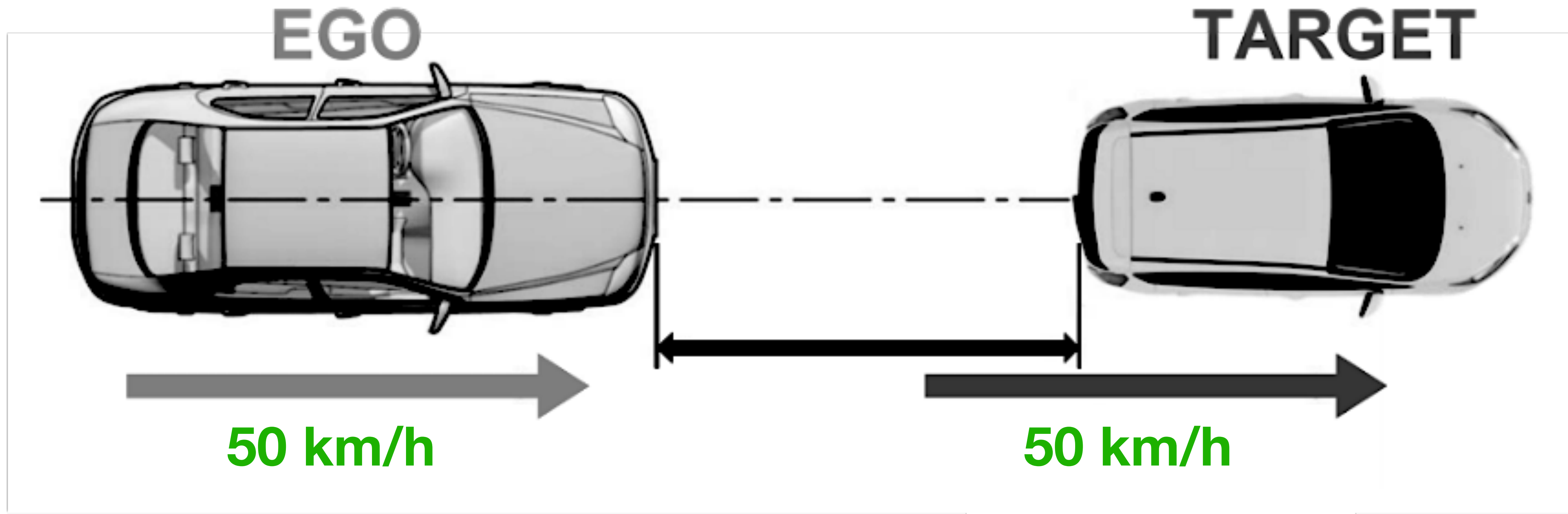
Nominal values

- Define the reference scenario to be tested and used as initial values during on-track experiments

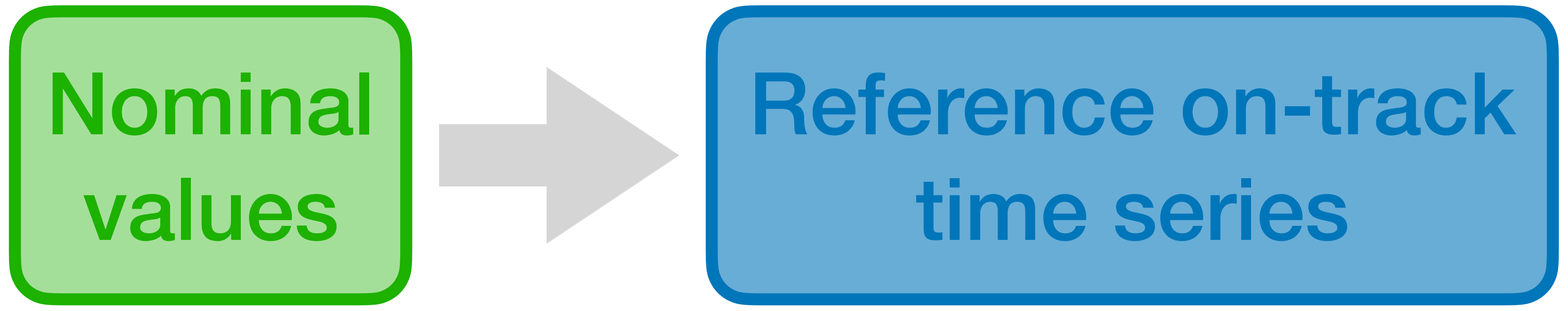


General introduction

Global objective of the thesis

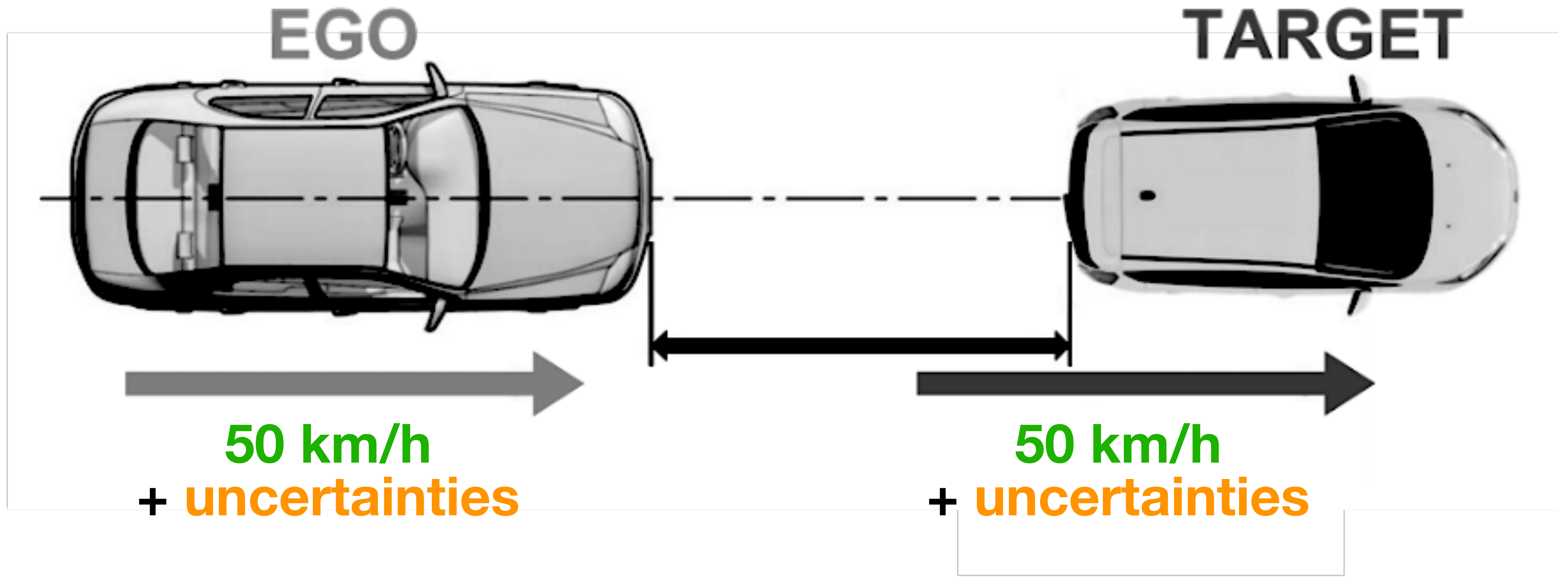


- **For each** **real-life on-track time series:** **nominal values** are given



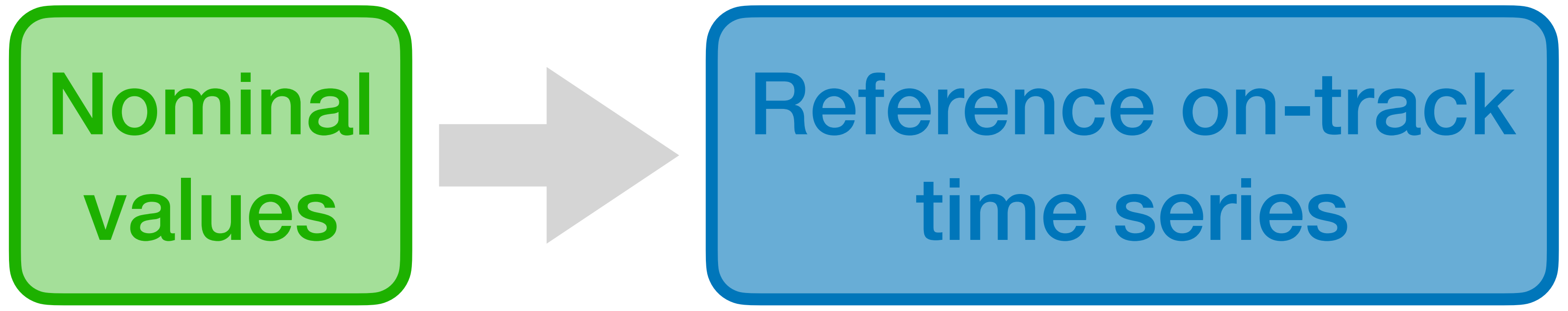
General introduction

Global objective of the thesis



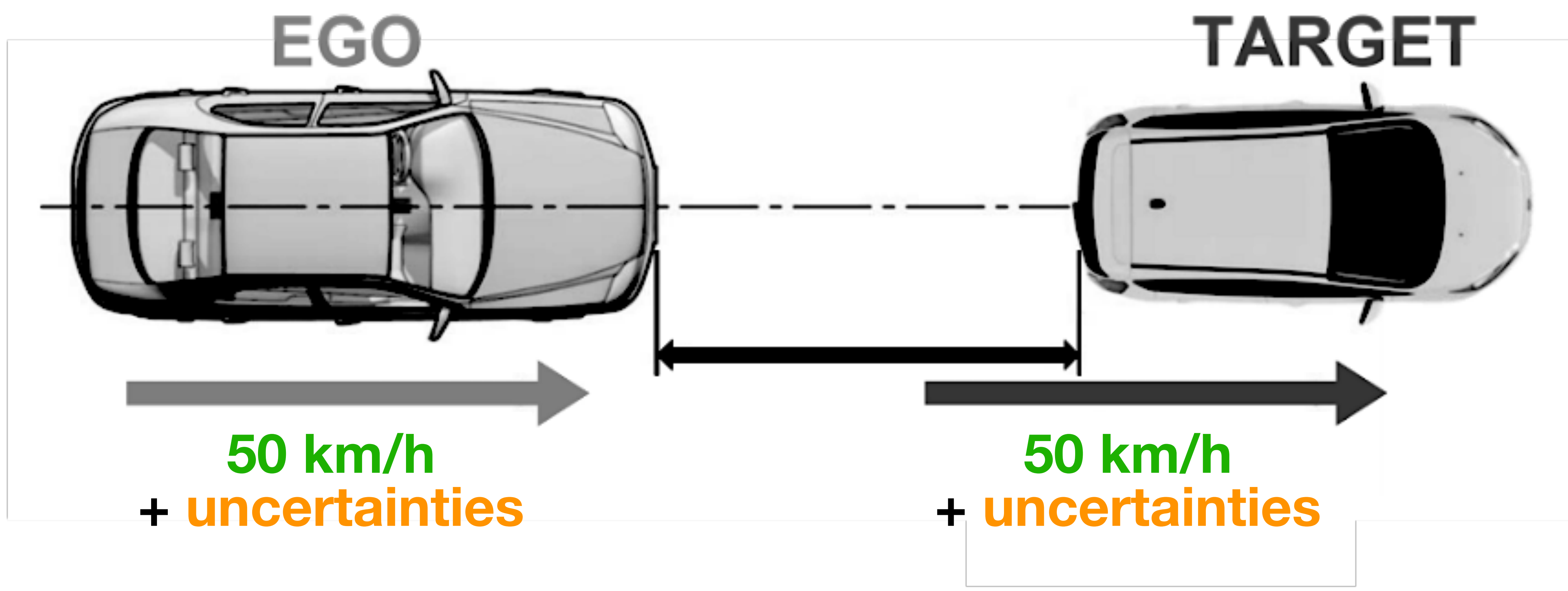
- **For each real-life on-track time series: nominal values** are given
- **Nominal values:** subject to uncertainties due to sensor errors and a certain tolerance

If the theoretical initial speed is 50 km/h, telling the simulation to start at 50.1 km/h can help simulate more realistic time series.



General introduction

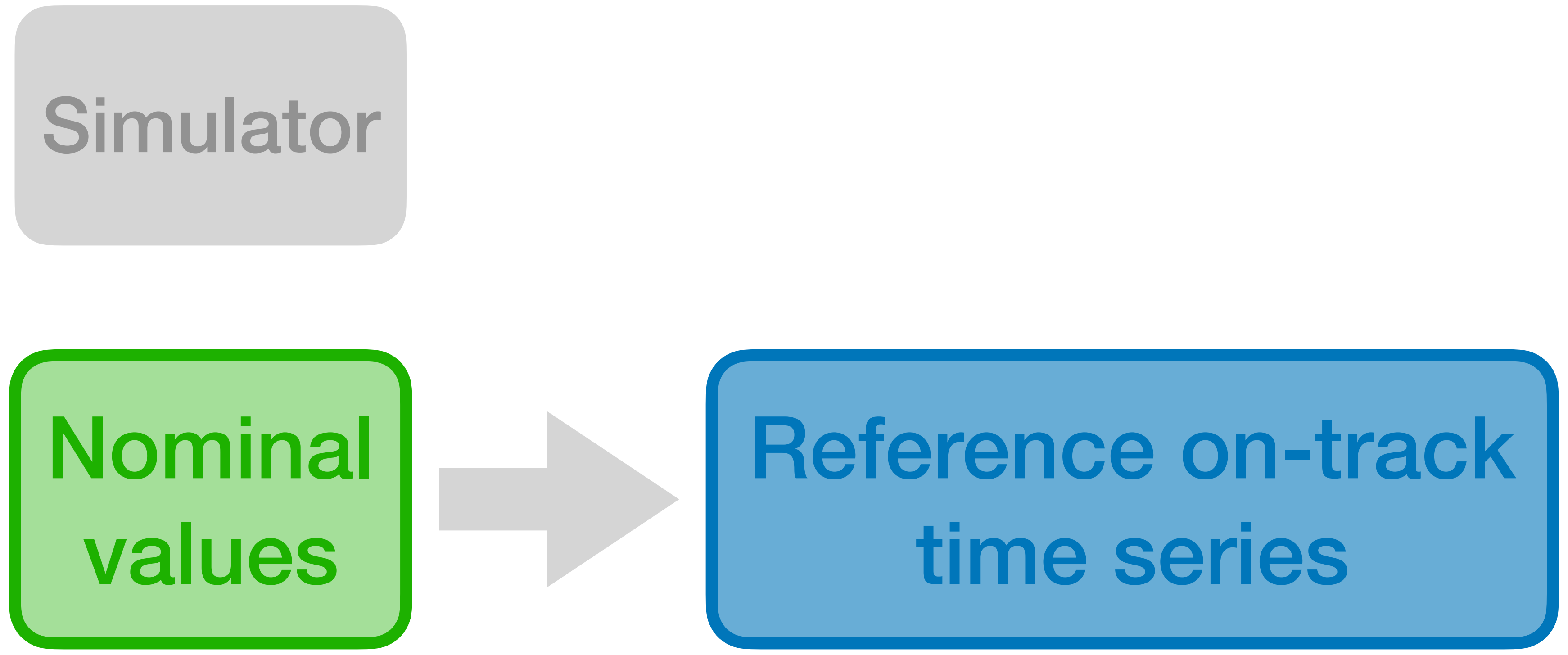
Global objective of the thesis



- **For each real-life on-track time series: nominal values** are given
- **Nominal values:** subject to uncertainties due to sensor errors and a certain tolerance

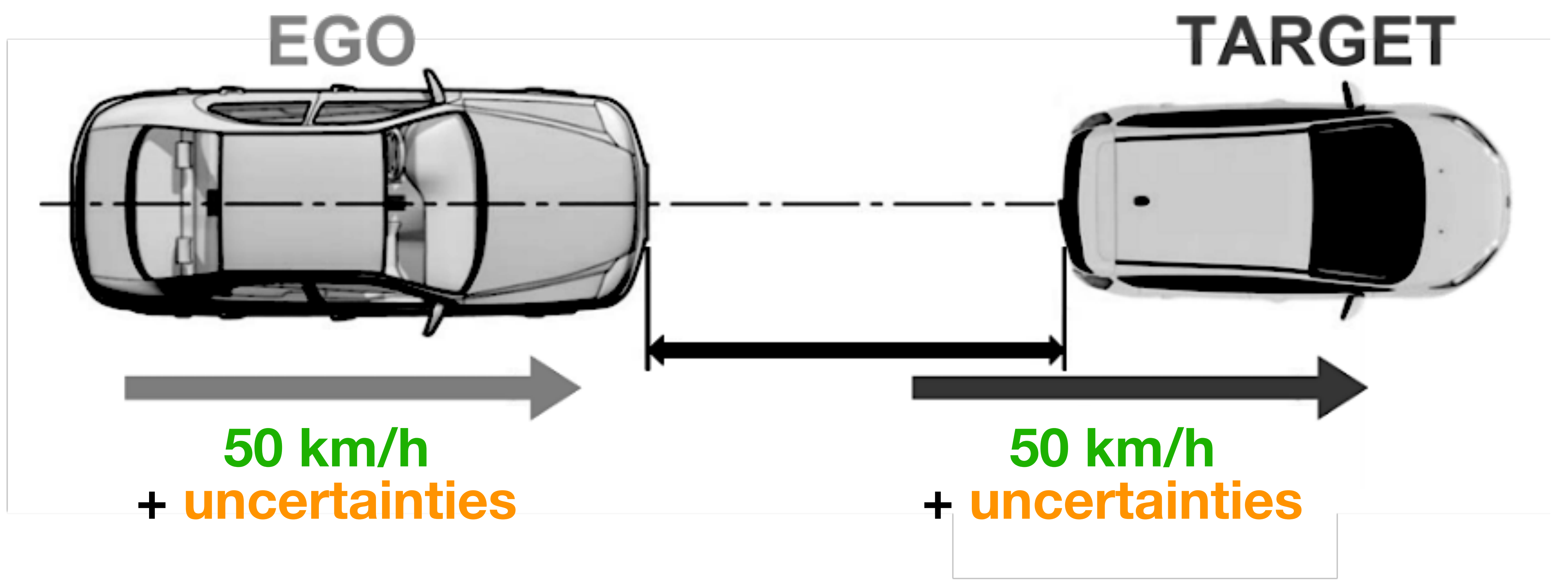
If the theoretical initial speed is 50 km/h, telling the simulation to start at 50.1 km/h can help simulate more realistic time series.

- **To calibrate the simulator:** by testing values around nominal values



General introduction

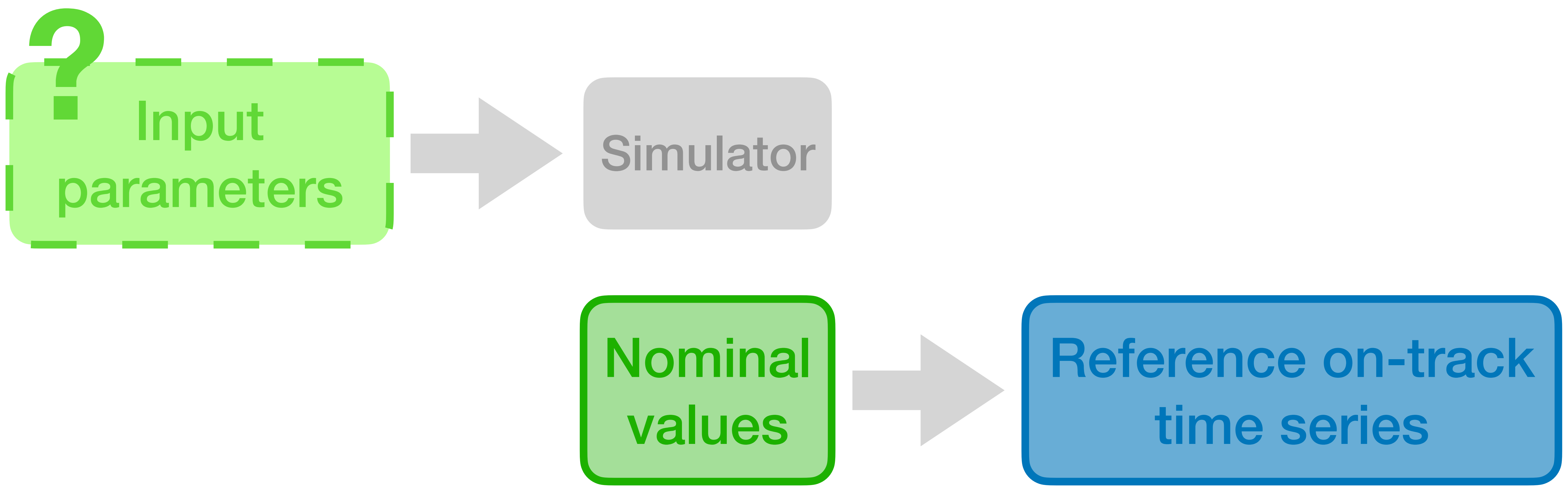
Global objective of the thesis



- **For each real-life on-track time series: nominal values** are given
- **Nominal values:** subject to uncertainties due to sensor errors and a certain tolerance

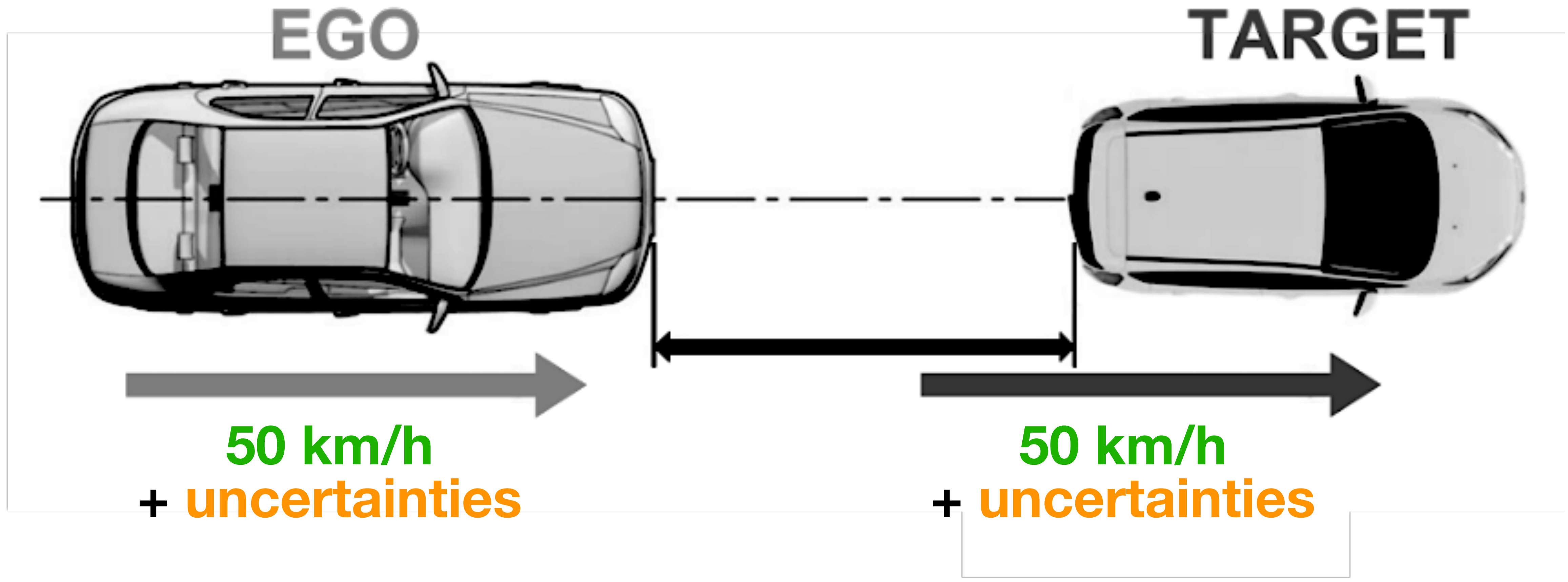
If the theoretical initial speed is 50 km/h, telling the simulation to start at 50.1 km/h can help simulate more realistic time series.

- **To calibrate the simulator:** by testing values around nominal values determine which input parameters



General introduction

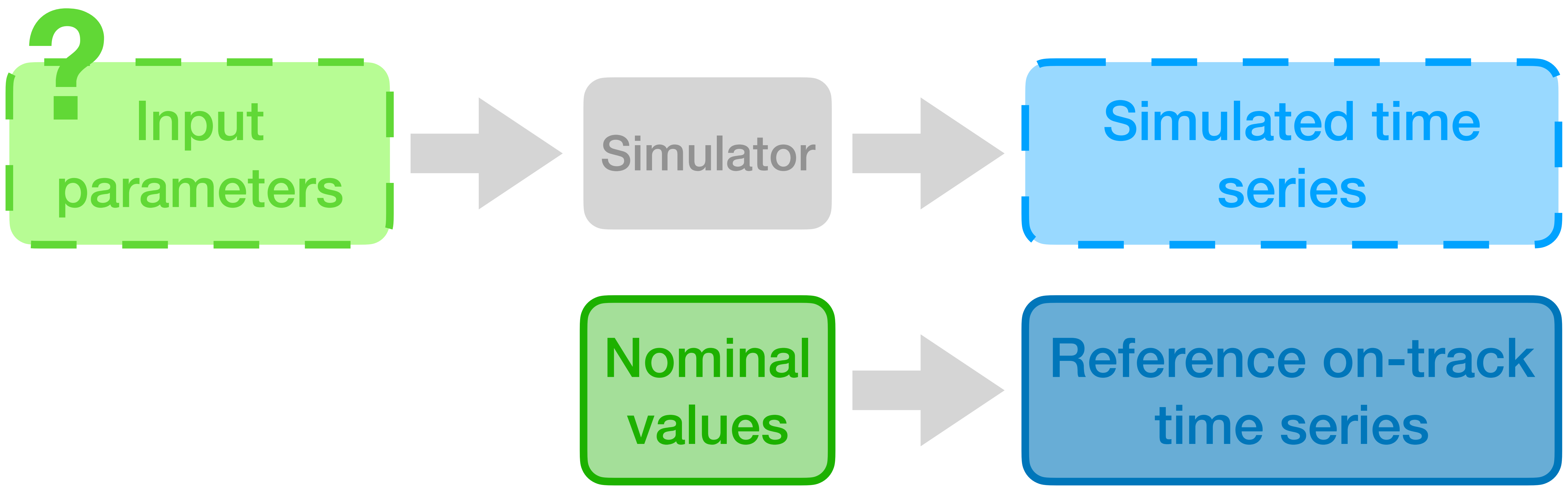
Global objective of the thesis



- **For each real-life on-track time series: nominal values** are given
- **Nominal values:** subject to uncertainties due to sensor errors and a certain tolerance

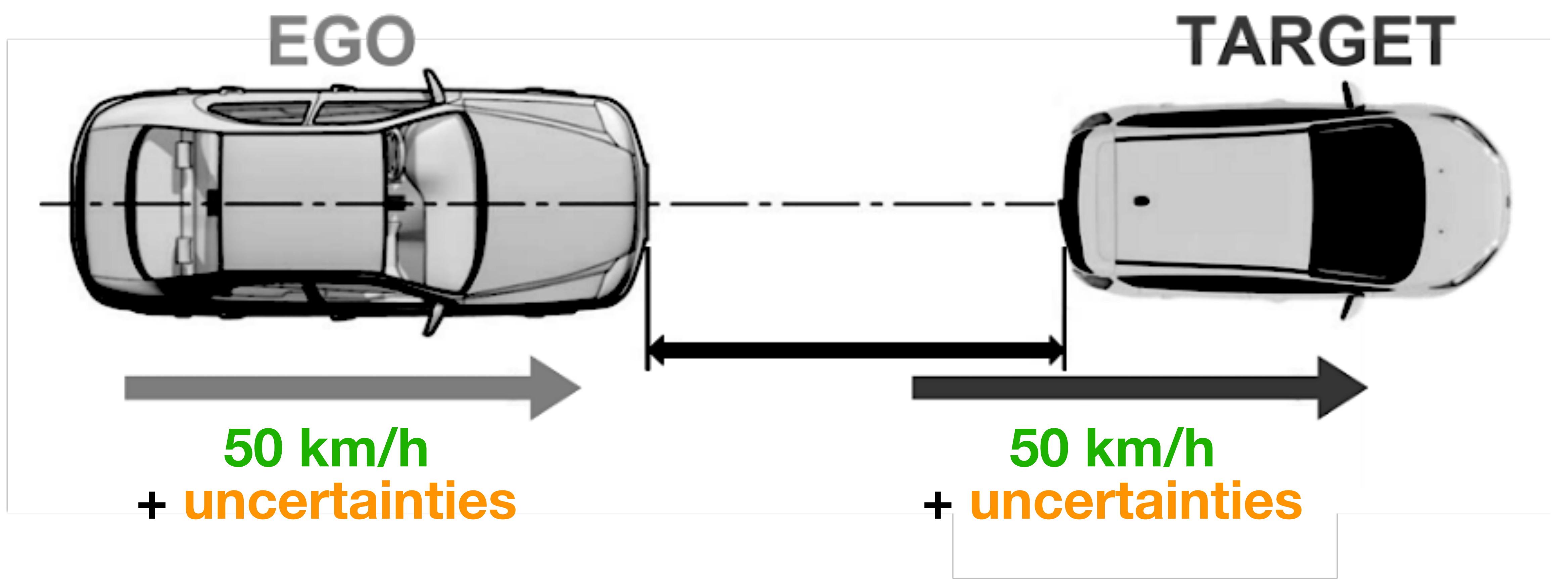
If the theoretical initial speed is 50 km/h, telling the simulation to start at 50.1 km/h can help simulate more realistic time series.

- **To calibrate the simulator:** by testing values around nominal values determine which input parameters provide the simulated time series



General introduction

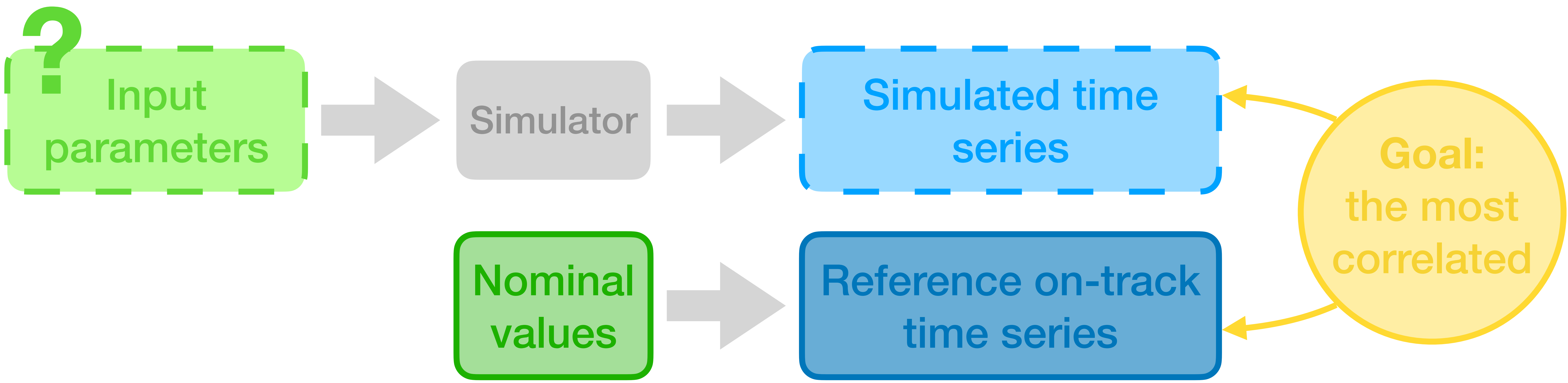
Global objective of the thesis



- **For each real-life on-track time series: nominal values** are given
- **Nominal values:** subject to **uncertainties** due to sensor errors and a certain tolerance

If the theoretical initial speed is 50 km/h, telling the simulation to start at 50.1 km/h can help simulate more realistic time series.

- **To calibrate the simulator:** by testing values around nominal values determine which **input parameters** provide the **simulated time series** the **most correlated** to real on-track ones



General introduction

Global objective of the thesis

Input values
θ

Model
$y = S(\theta) + \varepsilon$

Output values
y

General introduction

Global objective of the thesis

Input values
θ

Model
$y = S(\theta) + \varepsilon$

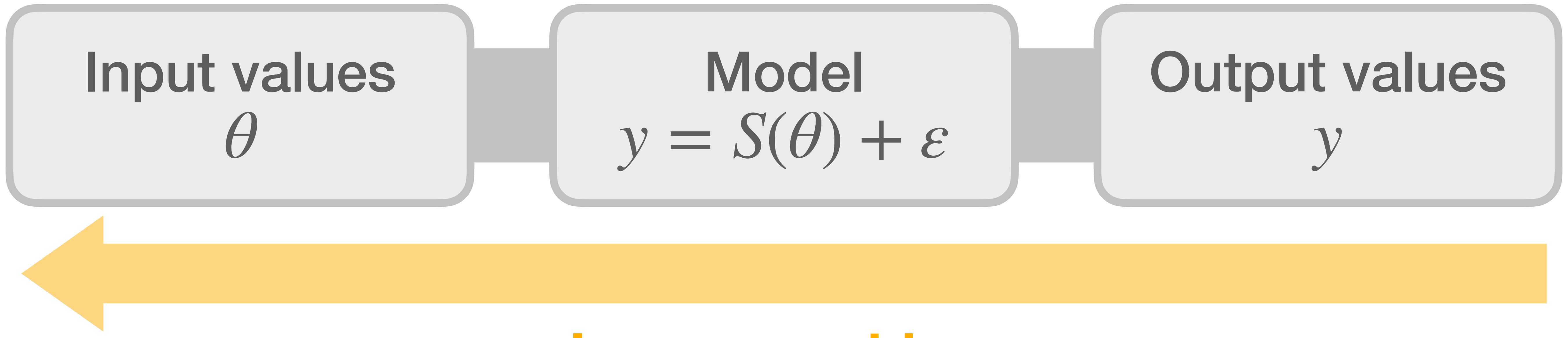
Output values
y

Inverse problem

Infer $\widehat{\theta}$ such that $y = S(\widehat{\theta})$

General introduction

Global objective of the thesis



Inverse problem

Infer $\hat{\theta}$ such that $y = S(\hat{\theta})$

Solve the following inverse problem

- For a reference on-track time series, recover the input parameters that simulate the closest time series

$$\underset{\theta \in \Theta}{\arg\min}\; d\big(S(\theta) + \varepsilon, y_\varphi\big)$$

Θ *: the parameter space to explore ;* d *: a distance ;* S *: the simulator ;* y_φ *: the on-track reference time series*

2. Solving the inverse problem

Solving the inverse problem

Introduction

Bayesian approach

- Estimate the posterior distribution given by Bayes' theorem

$$\pi(\theta \,|\, Y = y_\varphi) \propto p(Y = y_\varphi \,|\, \theta)\, \pi_0(\theta)$$

$\pi_0(\theta)$: *prior distribution* ; $p(Y = y_\varphi \,|\, \theta)$: *likelihood*

Solving the inverse problem

Introduction

Bayesian approach

- Estimate the posterior distribution given by Bayes' theorem

$$\pi(\theta \,|\, Y = y_\varphi) \propto p(Y = y_\varphi \,|\, \theta)\, \pi_0(\theta)$$

$\pi_0(\theta)$ *: prior distribution ;* $p(Y = y_\varphi \,|\, \theta)$ *: likelihood*

When computing the likelihood function is impractical or intractable

- Use *likelihood-free* methods to generate a sample $\theta_1, \ldots, \theta_n$ of the posterior and then approximate it

$$\pi(\theta \,|\, y_\varphi) \approx \frac{1}{n} \sum_{i=1}^{n} \delta_{\theta_i}(\theta)$$

Solving the inverse problem

Introduction

Bayesian approach

- Estimate the posterior distribution given by Bayes' theorem

$$\pi(\theta\,|\,Y = y_\varphi) \propto p(Y = y_\varphi\,|\,\theta)\,\pi_0(\theta)$$

$\pi_0(\theta)$ *: prior distribution ;* $p(Y = y_\varphi\,|\,\theta)$ *: likelihood*

When computing the likelihood function is impractical or intractable

- Use *likelihood-free* methods to generate a sample $\theta_1, \ldots, \theta_n$ of the posterior and then approximate it

$$\pi(\theta\,|\,y_\varphi) \approx \frac{1}{n}\sum_{i=1}^{n} \delta_{\theta_i}(\theta)$$

How do you generate such a sample?

Solving the inverse problem

Approximate Bayesian computation (ABC)

Generic rejection sampler

Initialization:

- Draw the first particles according to the prior $\theta_1^{(0)}, \dots, \theta_n^{(0)} \sim \pi_0$

Iterations:

1: **for** $k = 0, \dots, K$ **do**

2: Generate the associated time series $y_i^{(k)} = S\big(\theta_i^{(k)}\big) \quad \forall i = 1, \dots, n$

3: Determine accepted particles $A_k = \{i : d(y_i^{(k)}, y_\varphi) \leq h_k\} \subseteq \{1, \dots, n\}$

4: **Update step:**

8: Generate the new particles with a *rejection sampling* method

$$\theta_1^{(k+1)}, \dots, \theta_n^{(k+1)} \sim \pi_0$$

Solving the inverse problem

Approximate Bayesian computation (ABC)

Weighted rejection sampler

Initialization:

- Draw the first particles according to the prior $\theta_1^{(0)}, \dots, \theta_n^{(0)} \sim \pi_0$

Iterations:

1: **for** $k = 0, \dots, K$ **do**

2: Generate the associated time series $y_i^{(k)} = S\big(\theta_i^{(k)}\big) \quad \forall i = 1, \dots, n$

3: Determine accepted particles $A_k = \{i : d(y_i^{(k)}, y_\varphi) \leq h_k\} \subseteq \{1, \dots, n\}$

4: **Update step:**

6: Compute the weights $\omega_i^{(k)} = \exp\big(-d(y_i^{(k)}, y_\varphi)/h_k\big) \quad \forall i \in A_k$

8: Generate the new particles with a *rejection sampling* method

$$\theta_1^{(k+1)}, \dots, \theta_n^{(k+1)} \sim \pi_0$$

1) Assign importance weight to accepted particles

$$\exp\Big(-\frac{1}{h} d\big(S(\theta), y_\varphi\big)\Big)$$

Solving the inverse problem

Approximate Bayesian computation (ABC)

Sequential Monte Carlo sampler with weighted particles

Initialization:

- Draw the first particles according to the prior $\theta_1^{(0)}, \dots, \theta_n^{(0)} \sim \pi_0$

Iterations:

1: **for** $k = 0, \dots, K$ **do**
2: Generate the associated time series $y_i^{(k)} = S\big(\theta_i^{(k)}\big) \quad \forall i = 1, \dots, n$
3: Determine accepted particles $A_k = \{i : d(y_i^{(k)}, y_\varphi) \leq h_k\} \subseteq \{1, \dots, n\}$
4: **Update step:**
6: Compute the weights $\omega_i^{(k)} = \exp\big(-d(y_i^{(k)}, y_\varphi)/h_k\big) \quad \forall i \in A_k$
7: Construct the next distribution with a weighted KDE

$$\pi_{k+1}(\theta) = \frac{1}{|A_k|} \sum_{i \in A_k} \omega_i^{(k)} G_\rho\left(\theta - \theta_i^{(k)}\right)$$

8: Generate the new particles with a *rejection sampling* method

$$\theta_1^{(k+1)}, \dots, \theta_n^{(k+1)} \sim \pi_{k+1}$$

1) Assign importance weight to accepted particles

$$\exp\Big(-\frac{1}{h} d\big(S(\theta), y_\varphi\big)\Big)$$

2) Construct a sequence of slowly changing intermediary distibutions $\pi_1, \dots, \pi_{K-1}$ which bridge between the prior and the posterior

Solving the inverse problem

Approximate Bayesian computation (ABC)

Sequential Monte Carlo sampler with weighted particles and decreasing tolerance

Initialization:

- Initiate the changing tolerance $h_0 = h$
- Draw the first particles according to the prior $\theta_1^{(0)}, \ldots, \theta_n^{(0)} \sim \pi_0$

Iterations:

1: **for** $k = 0, \ldots, K$ **do**

2: Generate the associated time series $y_i^{(k)} = S\big(\theta_i^{(k)}\big) \quad \forall i = 1, \ldots, n$

3: Determine accepted particles $A_k = \{i : d(y_i^{(k)}, y_\varphi) \le h_k\} \subseteq \{1, \ldots, n\}$

4: **Update step:**

5: Compute the new tolerance $h_{k+1} = h_0(k+1)^{-1/\gamma}$

6: Compute the weights $\omega_i^{(k)} = \exp\big(-d(y_i^{(k)}, y_\varphi)/h_k\big) \quad \forall i \in A_k$

7: Construct the next distribution with a weighted KDE

$$\pi_{k+1}(\theta) = \frac{1}{|A_k|} \sum_{i \in A_k} \omega_i^{(k)} G_\rho\left(\theta - \theta_i^{(k)}\right)$$

8: Generate the new particles with a *rejection sampling* method

$$\theta_1^{(k+1)}, \ldots, \theta_n^{(k+1)} \sim \pi_{k+1}$$

1) Assign importance weight to accepted particles

$$\exp\left(-\frac{1}{h} d\big(S(\theta), y_\varphi\big)\right)$$

2) Construct a sequence of slowly changing intermediary distibutions $\pi_1, \ldots, \pi_{K-1}$ which bridge between the prior and the posterior

3) Monotonically decrease tolerance parameter

$$h_k = k^{-1/\gamma} h_0$$

Solving the inverse problem

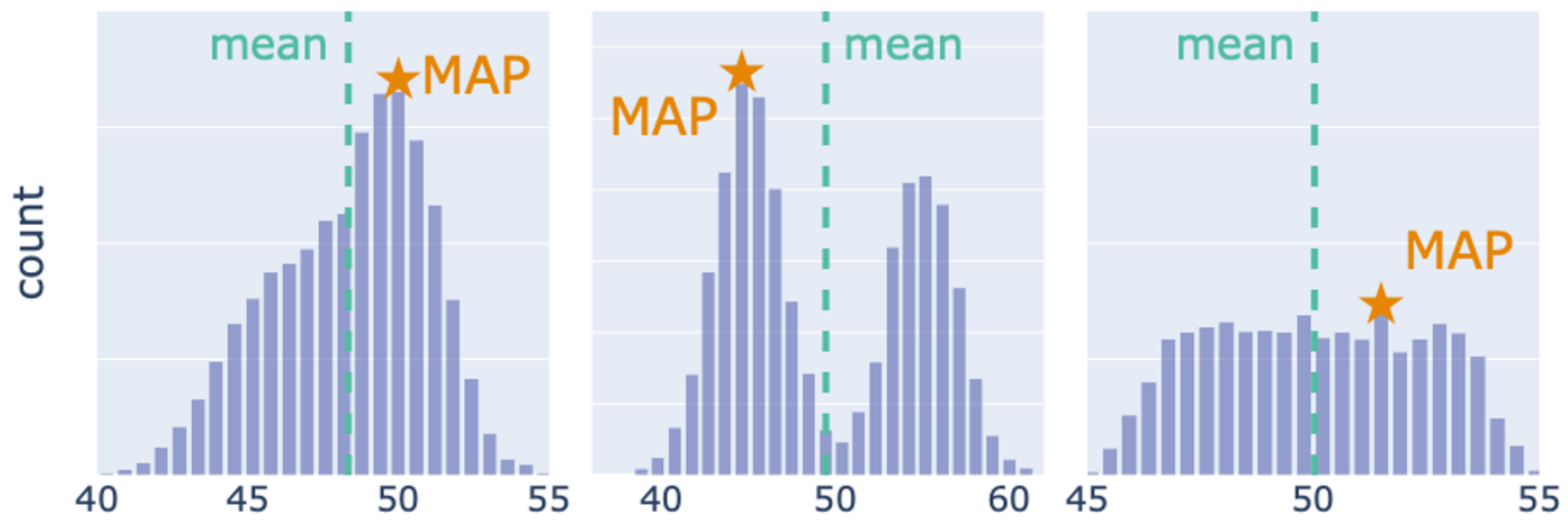
Constructing and evaluating estimators

How to transform posterior distribution into a point value estimation?

Solving the inverse problem

Constructing and evaluating estimators

How to transform posterior distribution into a point value estimation?



A single point value estimation

1) Weighted mean
2) Coordinate-wise maximum a posteriori

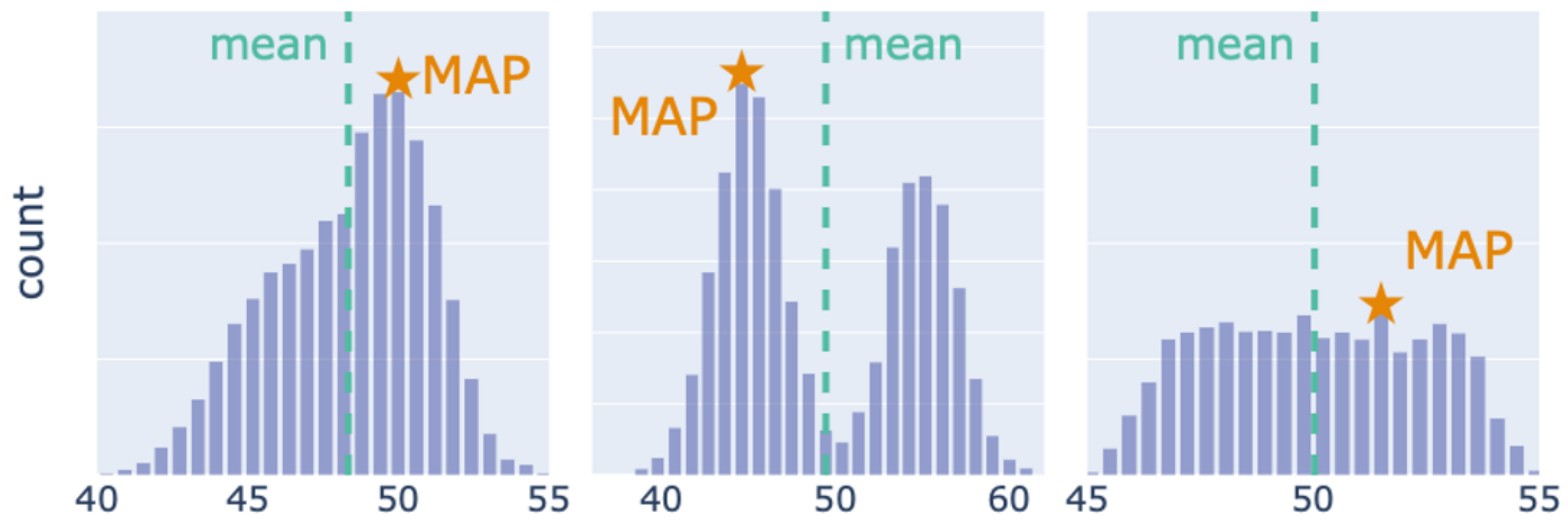
- Maximum a posteriori criterion

$$\hat{\theta}_{\text{MAP}} \in \underset{\theta \in \Theta}{\arg\max}\ \pi(\theta \mid Y = y_\varphi)$$

Solving the inverse problem

Constructing and evaluating estimators

How to transform posterior distribution into a point value estimation?



A single point value estimation

1) Weighted mean
2) Coordinate-wise maximum a posteriori

- Maximum a posteriori criterion

$$\hat{\theta}_{\text{MAP}} \in \underset{\theta \in \Theta}{\arg\max}\ \pi(\theta \mid Y = y_\varphi)$$

How to evaluate the quality of the obtained results?

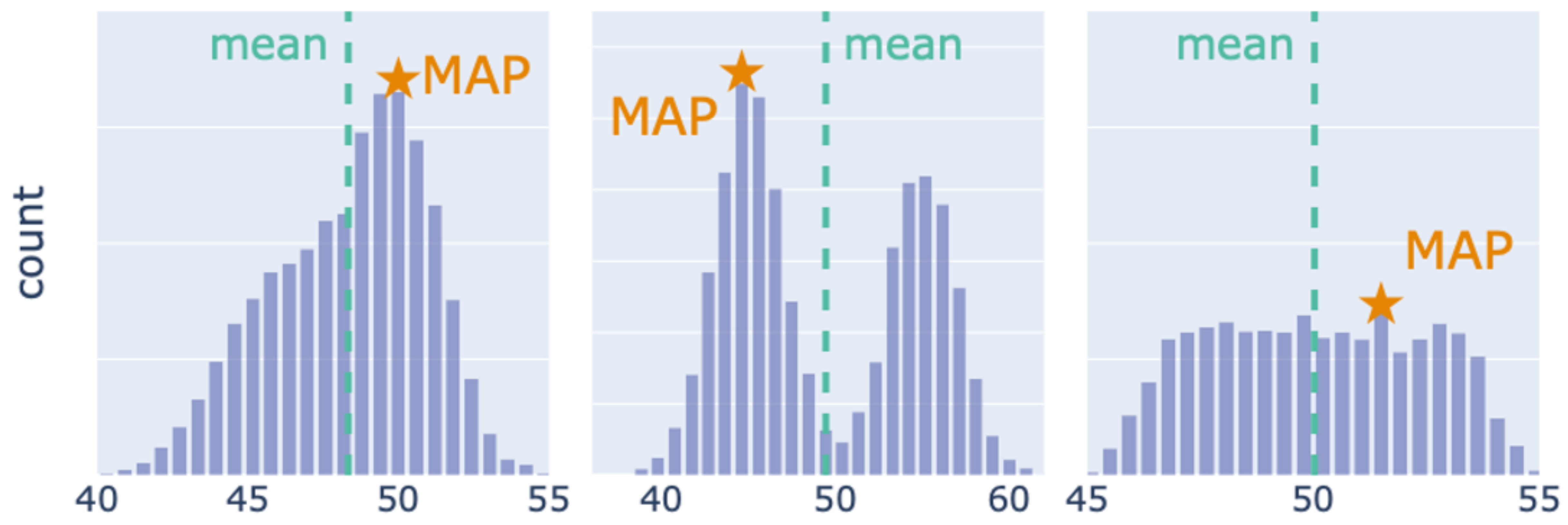
Simulator

y_φ

Solving the inverse problem

Constructing and evaluating estimators

How to transform posterior distribution into a point value estimation?



A single point value estimation

1) Weighted mean

2) Coordinate-wise maximum a posteriori

- Maximum a posteriori criterion

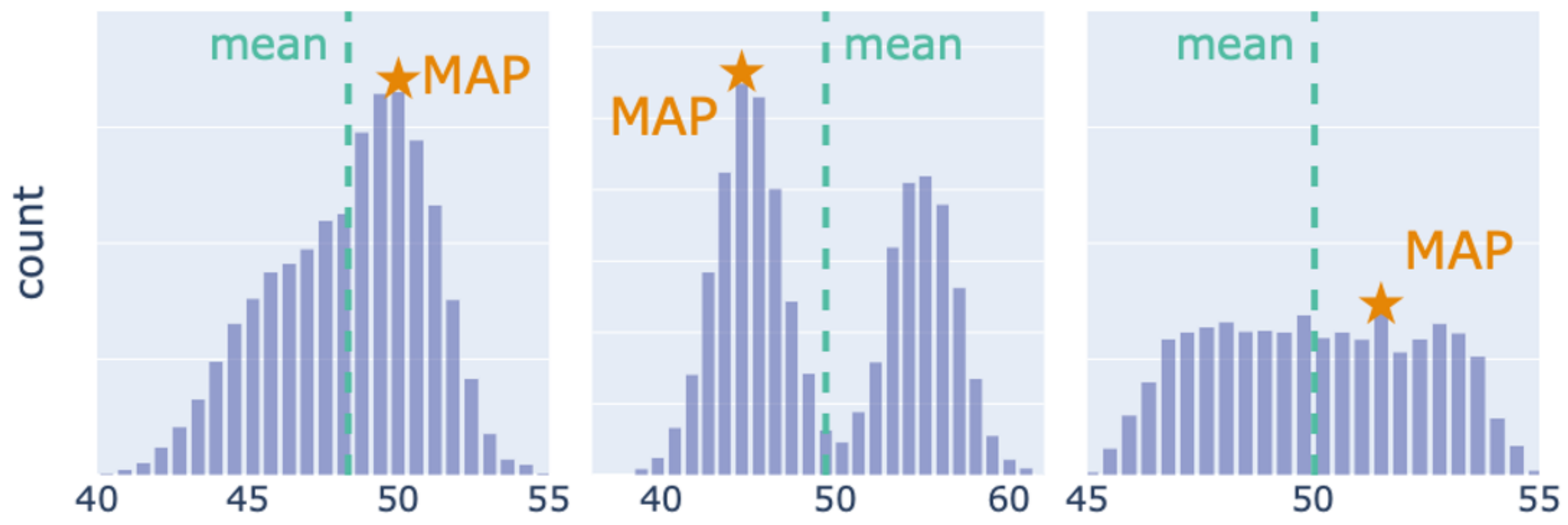
$$\hat{\theta}_{\text{MAP}} \in \underset{\theta \in \Theta}{\text{arg max}}\ \pi(\theta \mid Y = y_\varphi)$$

How to evaluate the quality of the obtained results?

Solving the inverse problem

Constructing and evaluating estimators

How to transform posterior distribution into a point value estimation?



A single point value estimation

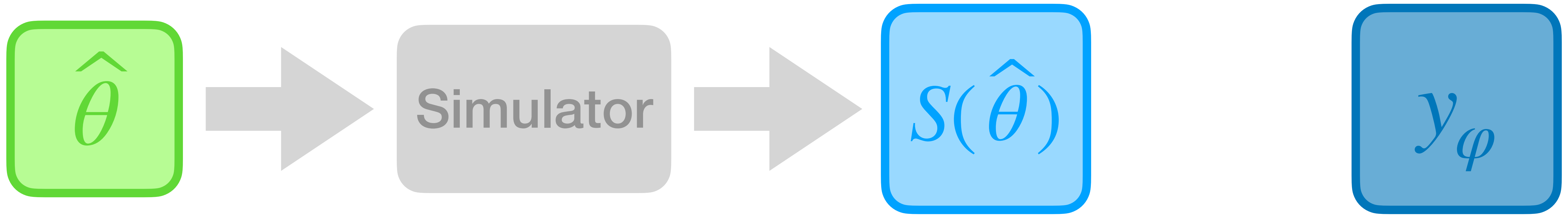
1) Weighted mean

2) Coordinate-wise maximum a posteriori

▸ Maximum a posteriori criterion

$$\hat{\theta}_{\text{MAP}} \in \underset{\theta \in \Theta}{\arg\max}\ \pi(\theta \,|\, Y = y_\varphi)$$

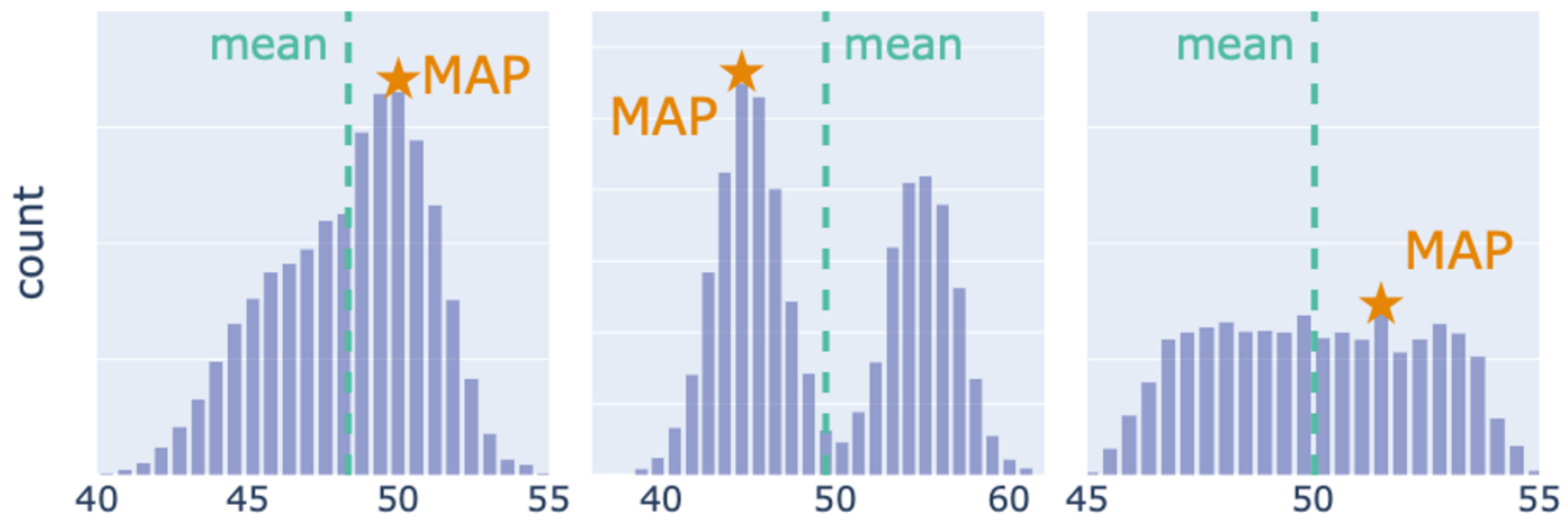
How to evaluate the quality of the obtained results?



Solving the inverse problem

Constructing and evaluating estimators

How to transform posterior distribution into a point value estimation?



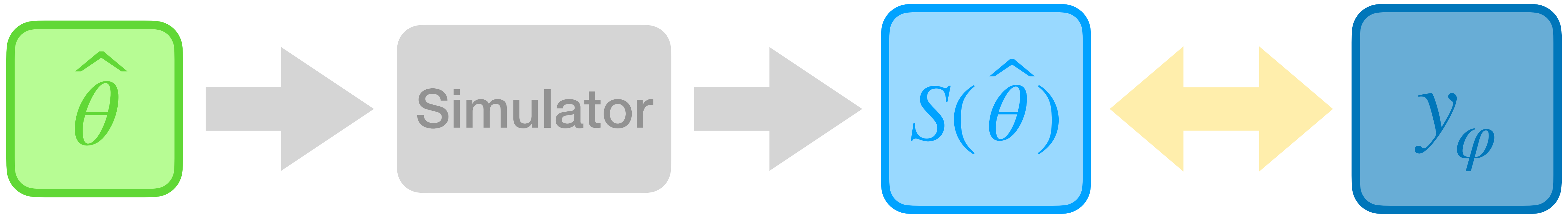
A single point value estimation

1) Weighted mean
2) Coordinate-wise maximum a posteriori

- Maximum a posteriori criterion

$$\widehat{\theta}_{\text{MAP}} \in \underset{\theta \in \Theta}{\arg\max}\ \pi(\theta \,|\, Y = y_\varphi)$$

How to evaluate the quality of the obtained results?



Solving the inverse problem

Conclusion

	generic rejection	SMC	SMC
option	mean	mean	MAP
s-RMSE	0.1787	0.1673	**0.1628**
time	9 min	2 min	2.5 min

Conclusion

- Our SMC sampler performs better
- Computing the MAP provides a better result
- Reasonable computation times

Solving the inverse problem

Conclusion

	generic rejection	SMC	SMC
option	mean	mean	MAP
s-RMSE	0.1787	0.1673	**0.1628**
time	9 min	2 min	2.5 min

Conclusion

- Our SMC sampler performs better
- Computing the MAP provides a better result
- Reasonable computation times

Other possibilities

- Optimization techniques with *gradient-free* methods

Solving the inverse problem

Conclusion

	generic rejection	SMC	SMC
option	mean	mean	MAP
s-RMSE	0.1787	0.1673	**0.1628**
time	9 min	2 min	2.5 min

Conclusion

- Our SMC sampler performs better
- Computing the MAP provides a better result
- Reasonable computation times

Model

$$y = S(\theta) + \varepsilon$$

Other possibilities

- Optimization techniques with *gradient-free* methods
- Depending on the nature of the noise ε : Bayesian synthetic likelihood (BSL)

Solving the inverse problem

A new problematic

Input values θ → Model $y = S(\theta) + \varepsilon$ → Output values y

One difficulty araises

- *Likelihood-free* methods require the generation of numerous outputs y
- The simulator is computationally too expensive

Solving the inverse problem

A new problematic

Input values	Model	Output values
θ	$y = S(\theta) + \varepsilon$	y

One difficulty araises

- *Likelihood-free* methods require the generation of numerous outputs y
- The simulator is computationally too expensive

For example

- We generated 2000 outputs to obtain the previous result with SMC sampler

	1 output	**2000 outputs** *
time	5 minutes	7 days

* *stop and restard the simulator at each step*

Solving the inverse problem

A new problematic

Forward problem

Predict $\widehat{y} = \widehat{S}(\theta)$

Input values θ → Model $y = S(\theta) + \varepsilon$ → Output values y

One difficulty araises

- *Likelihood-free* methods require the generation of numerous outputs y
- The simulator is computationally too expensive

For example

- We generated 2000 outputs to obtain the previous result with SMC sampler

	1 output	2000 outputs *
time	5 minutes	7 days

- We need to train a surrogate model $\widehat{S}$ which mimics the simulator S
- Predicts y for a given θ using $\widehat{y} = \widehat{S}(\theta)$

* *stop and restard the simulator at each step*

3. Construction of the surrogate model

Construction of the surrogate model

Introduction

Objectives

- Construct a surrogate model $\widehat{S}$ which mimics the simulator S
- For a given set of parameters θ, predict simulated time series y by $\widehat{y} = \widehat{S}(\theta)$
- Accurately replicates the simulator's behavior while maintaining reasonable calculation times

Construction of the surrogate model

Introduction

Objectives

- Construct a surrogate model $\widehat{S}$ which mimics the simulator S
- For a given set of parameters θ, predict simulated time series y by $\widehat{y} = \widehat{S}(\theta)$
- Accurately replicates the simulator's behavior while maintaining reasonable calculation times

Problematic

- In traditionnal time series forecasting, the objective is to predict the future from the past
- Here, we aim to predict a whole time series from a set of non-temporal parameters

Construction of the surrogate model

Introduction

Objectives

- Construct a surrogate model $\widehat{S}$ which mimics the simulator S
- For a given set of parameters θ, predict simulated time series y by $\widehat{y} = \widehat{S}(\theta)$
- Accurately replicates the simulator's behavior while maintaining reasonable calculation times

Problematic

- In traditionnal time series forecasting, the objective is to predict the future from the past
- Here, we aim to predict a whole time series from a set of non-temporal parameters

We want to built a generative model

Construction of the surrogate model

Introduction

Objectives

- Construct a surrogate model $\widehat{S}$ which mimics the simulator S
- For a given set of parameters θ, predict simulated time series y by $\widehat{y} = \widehat{S}(\theta)$
- Accurately replicates the simulator's behavior while maintaining reasonable calculation times

Problematic

- In traditionnal time series forecasting, the objective is to predict the future from the past
- Here, we aim to predict a whole time series from a set of non-temporal parameters

We want to built a generative model

What to do

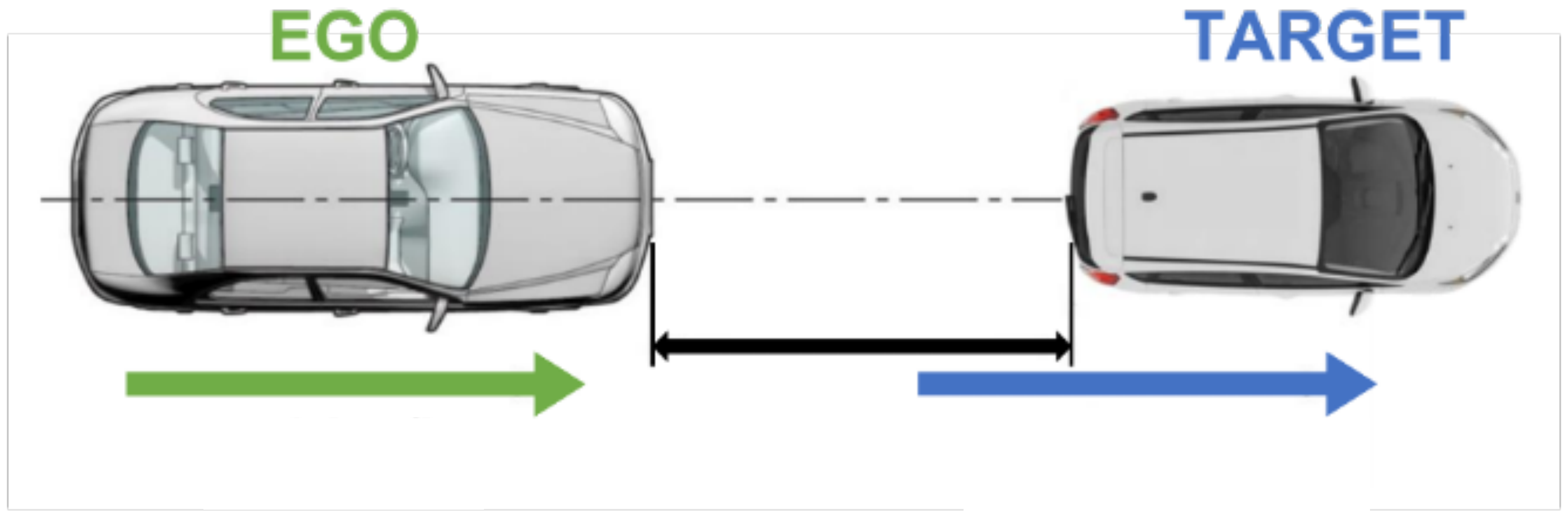
- Construct a dataset to build the model: beforehand simulated scenarios output by the simulator
- Choosing the right input parameter spaces

Construction of the surrogate model

Data description

Considered scenario

- Two vehicles in motion, the **target** vehicle is in front and the **ego** vehicle behind
- We test the automatic emergency braking of the ego vehicle following sudden braking by the target vehicle

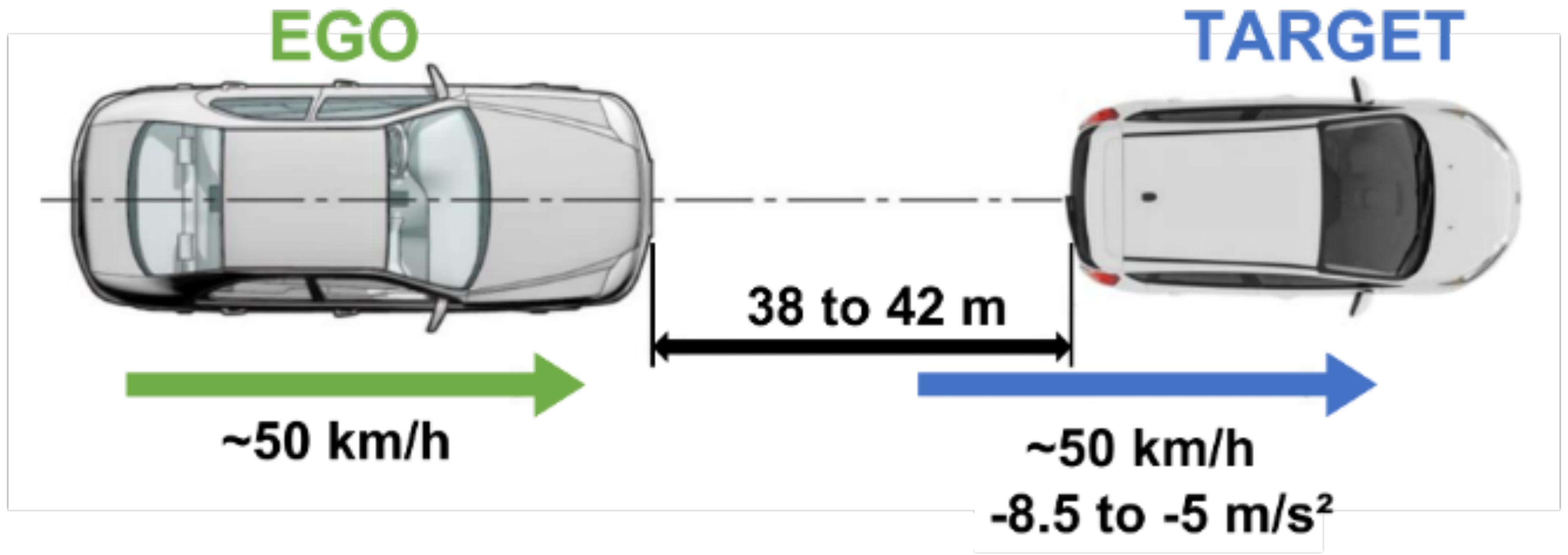


Construction of the surrogate model

Data description

Considered scenario

- Two vehicles in motion, the **target** vehicle is in front and the **ego** vehicle behind
- We test the automatic emergency braking of the ego vehicle following sudden braking by the target vehicle



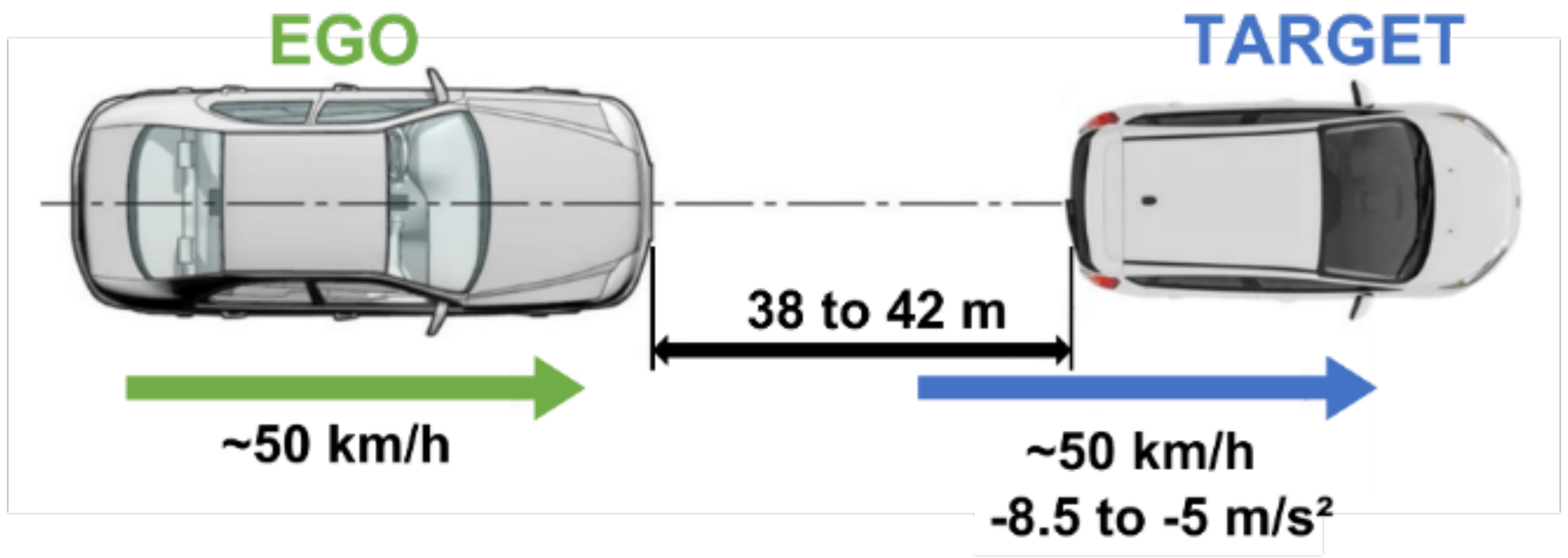
Input parameters: a total of 8 values

Construction of the surrogate model

Data description

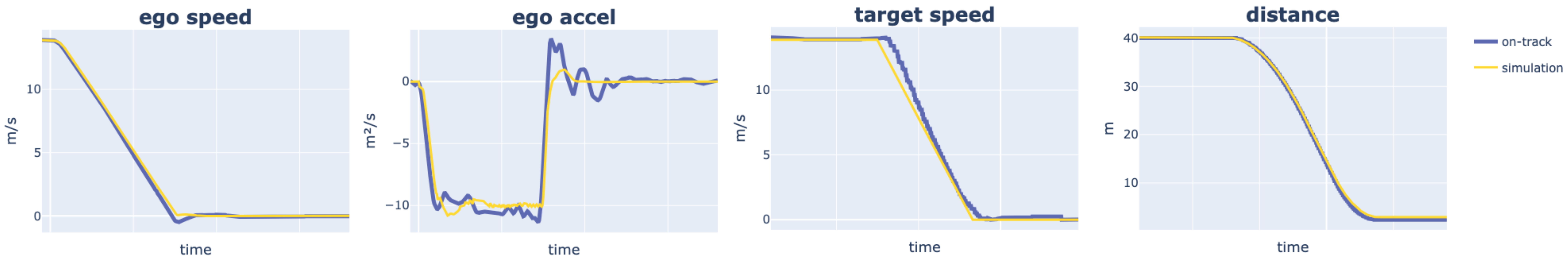
Considered scenario

- Two vehicles in motion, the target vehicle is in front and the ego vehicle behind
- We test the automatic emergency braking of the ego vehicle following sudden braking by the target vehicle



Input parameters: a total of 8 values

Output time series: 4 distinct types of time series, each one has a different number of time steps



Construction of the surrogate model

Expert aggregation model

To deal with time series, we consider expert aggregation

- We have M experts noted $(f_m)_{m=1,\ldots,M}$
- For a given input parameter θ, each one provides a prediction $f_{m,t}(\theta)$ for each time step t
- At each time step t, a weight vector ω_t is constructed to predict

$$\hat{y}_t = \sum_{m=1}^{M} \omega_{m,t} f_{m,t}(\theta)$$

Construction of the surrogate model

Expert aggregation model

To deal with time series, we consider expert aggregation

- We have M experts noted $(f_m)_{m=1,\ldots,M}$
- For a given input parameter θ, each one provides a prediction $f_{m,t}(\theta)$ for each time step t
- At each time step t, a weight vector ω_t is constructed to predict

$$\hat{y}_t = \sum_{m=1}^{M} \omega_{m,t} f_{m,t}(\theta)$$

How to construct weight vectors?

Construction of the surrogate model

Expert aggregation model

To deal with time series, we consider expert aggregation

- We have M experts noted $(f_m)_{m=1,\ldots,M}$
- For a given input parameter θ, each one provides a prediction $f_{m,t}(\theta)$ for each time step t
- At each time step t, a weight vector ω_t is constructed to predict

$$\hat{y}_t = \sum_{m=1}^{M} \omega_{m,t} \, f_{m,t}(\theta)$$

How to construct weight vectors?

Aggregated model

- Exponentially weighted aggregation (EWA) is used to compute weight vectors sequentially

$$\omega_{m,t} = \frac{\exp(-\eta L_{m,t-1}(\theta))}{\sum_j \exp(-\eta L_{j,t-1}(\theta))} \qquad \text{where} \qquad L_{m,T}(\theta) := \sum_{t=1}^{T} \ell(f_{m,t}(\theta), y_t)$$

Construction of the surrogate model

Expert aggregation model

To deal with time series, we consider expert aggregation

- We have M experts noted $(f_m)_{m=1,\ldots,M}$
- For a given input parameter θ, each one provides a prediction $f_{m,t}(\theta)$ for each time step t
- At each time step t, a weight vector ω_t is constructed to predict

$$\hat{y}_t = \sum_{m=1}^{M} \omega_{m,t} \, f_{m,t}(\theta)$$

How to construct weight vectors?

Aggregated model

- Exponentially weighted aggregation (EWA) is used to compute weight vectors sequentially

$$\omega_{m,t} = \frac{\exp(-\eta L_{m,t-1}(\theta))}{\sum_j \exp(-\eta L_{j,t-1}(\theta))} \qquad \text{where} \qquad L_{m,T}(\theta) := \sum_{t=1}^{T} \ell(f_{m,t}(\theta), y_t)$$

Hybrid model

- Select a single method per time step
- Only one value of the weight vector is equal to 1, and the rest is equal to 0

Construction of the surrogate model

Choice of experts

Classical machine learning

Construction of the surrogate model

Choice of experts

Classical machine learning

- k-nearest neighbors (k-NN)
  - One of the most fundamental and straightforward algorithm
  - 2 parameters: k and the norm $||.||$ to mesure distances between points

Construction of the surrogate model

Choice of experts

Classical machine learning

- k-nearest neighbors (k-NN)
  - One of the most fundamental and straightforward algorithm
  - 2 parameters: k and the norm $||.||$ to mesure distances between points
- Random forests (RF)
  - Improve bootstrap aggregating (= bagging) to reduce prediction variance
  - 2 parameters: A the number of trees and J the number of features to select

Construction of the surrogate model

Choice of experts

Classical machine learning

- k-nearest neighbors (k-NN)
  - One of the most fundamental and straightforward algorithm
  - 2 parameters: k and the norm $||.||$ to mesure distances between points
- Random forests (RF)
  - Improve bootstrap aggregating (= bagging) to reduce prediction variance
  - 2 parameters: A the number of trees and J the number of features to select
- Kernel ridge regression (KRR)
  - Combination of ridge regression and kernel trick
  - 2 main parameters: λ the regularization strength and the kernel mapping used

Construction of the surrogate model

Choice of experts

Classical machine learning

- k-nearest neighbors (k-NN)
  - One of the most fundamental and straightforward algorithm
  - 2 parameters: k and the norm $||.||$ to mesure distances between points
- Random forests (RF)
  - Improve bootstrap aggregating (= bagging) to reduce prediction variance
  - 2 parameters: A the number of trees and J the number of features to select
- Kernel ridge regression (KRR)
  - Combination of ridge regression and kernel trick
  - 2 main parameters: λ the regularization strength and the kernel mapping used

Dimensionality reduction

- Principal component analysis (PCA)
  - Classical: decomposes the covariance matrix into its eigenvalues
  - Functional: eigenfunctions instead of eigenvalues
  - Sparse: introduces sparsity structures to the variables

Construction of the surrogate model

Choice of experts

Classical machine learning

- k-nearest neighbors (k-NN)
  - One of the most fundamental and straightforward algorithm
  - 2 parameters: k and the norm $||.||$ to mesure distances between points
- Random forests (RF)
  - Improve bootstrap aggregating (= bagging) to reduce prediction variance
  - 2 parameters: A the number of trees and J the number of features to select
- Kernel ridge regression (KRR)
  - Combination of ridge regression and kernel trick
  - 2 main parameters: λ the regularization strength and the kernel mapping used

Dimensionality reduction

- Principal component analysis (PCA)
  - Classical: decomposes the covariance matrix into its eigenvalues
  - Functional: eigenfunctions instead of eigenvalues
  - Sparse: introduces sparsity structures to the variables

Neural networks

- Convolutionnal neural network (CNN)
  - Effective with image data
  - Spatial feature extraction capabilities are used to adapt visual analysis assets to the analysis of temporal data

Construction of the surrogate model

Construction of hybrid and aggregated models

Scaled RMSE $(\times 10^{-2})$ between true and predicted values

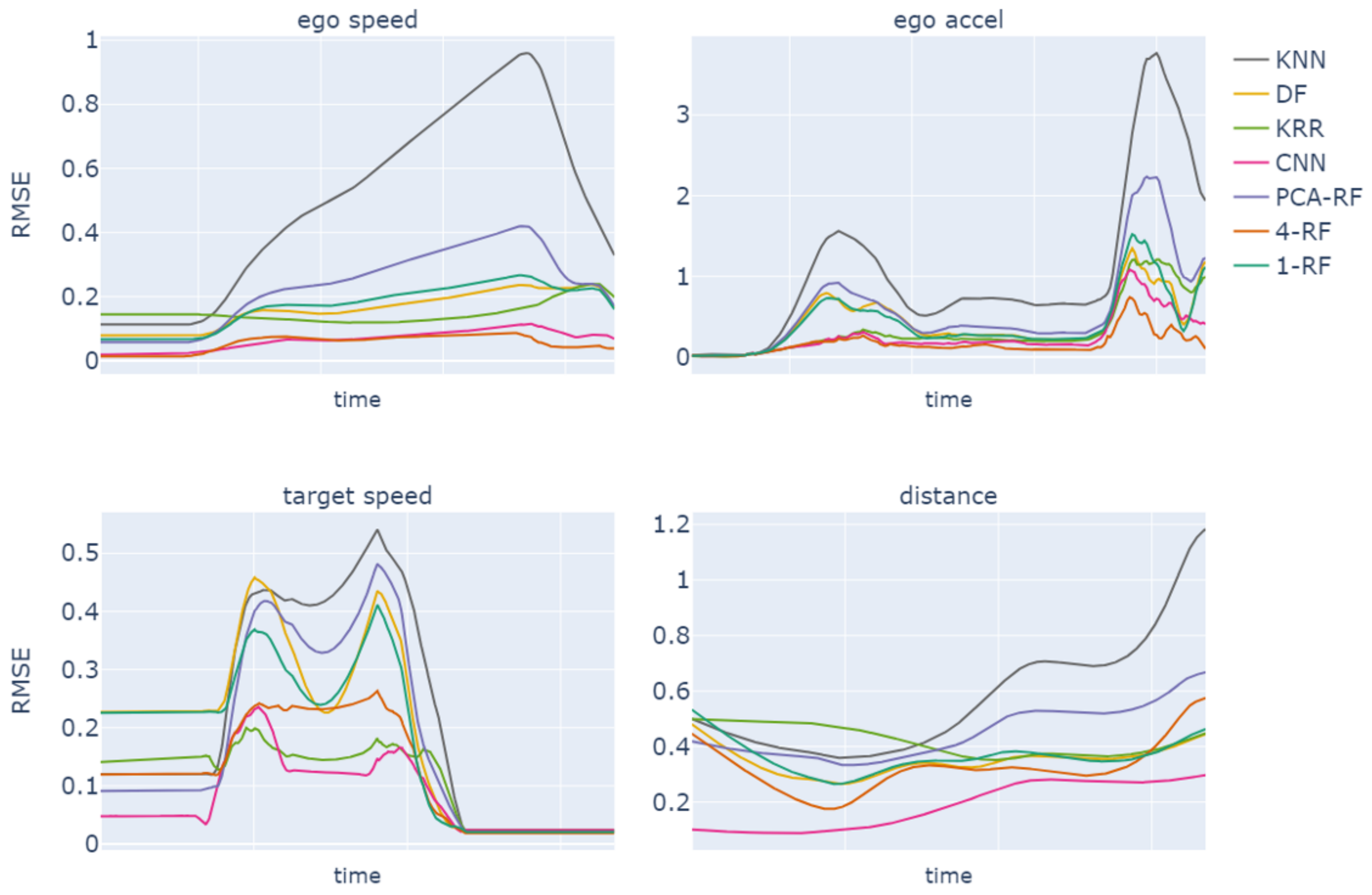
	Ego speed (m/s)	Ego accel (m/s^2)	Target speed (m/s)	Distance (m)
k-NN	11.00	64.77	8.02	36.83
1-RF	1.36	9.54	4.86	13.31
4-RF	**0.12**	**1.47**	2.34	10.84
KRR	2.20	7.14	1.63	17.86
PCA-RF	2.33	20.53	5.58	20.96
CNN	0.18	3.85	**1.06**	**4.16**

Construction of the surrogate model

Construction of hybrid and aggregated models

Scaled RMSE $(\times 10^{-2})$ between true and predicted values

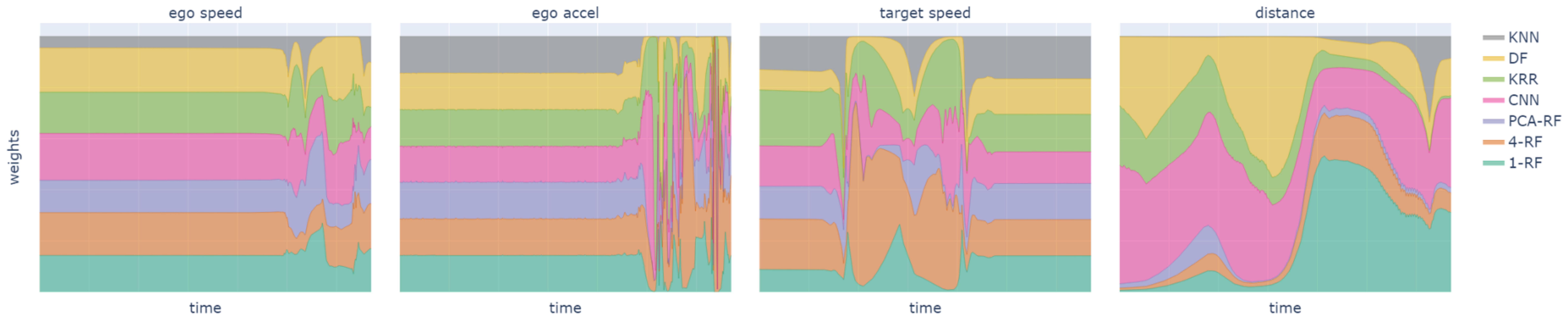
	Ego speed (m/s)	Ego accel (m/s²)	Target speed (m/s)	Distance (m)
k-NN	11.00	64.77	8.02	36.83
1-RF	1.36	9.54	4.86	13.31
4-RF	**0.12**	**1.47**	2.34	10.84
KRR	2.20	7.14	1.63	17.86
PCA-RF	2.33	20.53	5.58	20.96
CNN	0.18	3.85	**1.06**	**4.16**



Construction of the surrogate model

Construction of hybrid and aggregated models

Weight vectors for the aggregated model (EWA algorithm)



Construction of the surrogate model

Construction of hybrid and aggregated models

Two hybrid models

Construction of the surrogate model

Construction of hybrid and aggregated models

Two hybrid models

- Hybrid 1: select best method for each time step

Hybrid 1

Construction of the surrogate model

Construction of hybrid and aggregated models

Two hybrid models

- Hybrid 1: select best method for each time step
- Hybrid 2: select best method for each time step among the three best ones

Construction of the surrogate model

Results of hybrid and aggregated models

Validation set: used to compute weights

	Ego speed	Ego accel	Target speed	Distance	mean
CNN	0.18	3.85	1.06	4.16	2.31
hybrid 1	0.11	1.46	0.93	4.16	1.66
hybrid 2	0.11	1.46	1.04	4.16	1.69
aggregated	**0.07**	**0.59**	**0.24**	**1.36**	**0.56**

Construction of the surrogate model

Results of hybrid and aggregated models

Validation set: used to compute weights

	Ego speed	Ego accel	Target speed	Distance	mean
CNN	0.18	3.85	1.06	4.16	2.31
hybrid 1	0.11	1.46	0.93	4.16	1.66
hybrid 2	0.11	1.46	1.04	4.16	1.69
aggregated	**0.07**	**0.59**	**0.24**	**1.36**	**0.56**

Test set: unknown data

	Ego speed	Ego accel	Target speed	Distance	mean
CNN	0.23	3.34	1.00	2.52	1.77
hybrid 1	0.12	1.35	1.12	2.52	1.28
hybrid 2	0.12	1.35	**1.00**	2.52	**1.25**
aggregated	0.64	3.81	1.35	4.97	2.69

Construction of the surrogate model

Results of hybrid and aggregated models

Calculation times

	CNN	hybrid 1 (+)	hybrid 2 (+)	aggregated (+)	simulation
training time	59 min	0.29 sec	0.18 sec	2 min 13	
100 outputs	1.39 sec	10.25 sec	8.59 sec	2 min 17	
2000 outputs	30 seconds	3.5 minutes	3 minutes	46 minutes	7 days *

(+) *add the prediction times for each method*

***** *stop and restard the simulator at each step*

Construction of the surrogate model

Results of hybrid and aggregated models

Calculation times

	CNN	hybrid 1 (+)	hybrid 2 (+)	aggregated (+)	simulation
training time	59 min	0.29 sec	0.18 sec	2 min 13	
100 outputs	1.39 sec	10.25 sec	8.59 sec	2 min 17	
2000 outputs	30 seconds	3.5 minutes	3 minutes	46 minutes	7 days *

(+) *add the prediction times for each method*
***** *stop and restard the simulator at each step*

Perspective

- Aggregated model is disapointing because of its poor generalization to unknown data
- The EWA algorithm is not initially designed for time-dependent weights

Construction of the surrogate model

Results of hybrid and aggregated models

Calculation times

	CNN	hybrid 1 (+)	hybrid 2 (+)	aggregated (+)	simulation
training time	59 min	0.29 sec	0.18 sec	2 min 13	
100 outputs	1.39 sec	10.25 sec	8.59 sec	2 min 17	
2000 outputs	30 seconds	3.5 minutes	3 minutes	46 minutes	7 days *

(+) *add the prediction times for each method*

***** *stop and restard the simulator at each step*

Perspective

- Aggregated model is disapointing because of its poor generalization to unknown data
- The EWA algorithm is not initially designed for time-dependent weights

Will the optimization tools work to compute time weights of the experts?

Construction of the surrogate model

Results of hybrid and aggregated models

Calculation times

	CNN	hybrid 1 (+)	hybrid 2 (+)	aggregated (+)	simulation
training time	59 min	0.29 sec	0.18 sec	2 min 13	
100 outputs	1.39 sec	10.25 sec	8.59 sec	2 min 17	
2000 outputs	30 seconds	3.5 minutes	3 minutes	46 minutes	7 days *

(+) *add the prediction times for each method*

***** *stop and restard the simulator at each step*

Perspective

- Aggregated model is disapointing because of its poor generalization to unknown data
- The EWA algorithm is not initially designed for time-dependent weights

Will the optimization tools work to compute time weights of the experts?

Goal: define a framework that incorporates this functional setting to ensure algorithm convergence

4. Theoretical guarantees for functional expert aggregation

Theoretical guarantees for functional expert aggregation

Introduction

Expert aggregation

- We have M experts noted $(f_m)_{m=1,\ldots,M}$
- For a given input parameter θ, each one provides a prediction $f_{m,t}(\theta)$ for each time step t
- Construct a temporal weight vector ω_t to predict y_t by

$$\widehat{y}_t = \sum_{m=1}^{M} \omega_{m,t} f_{m,t}(\theta) \quad \text{with} \quad \sum_{t=1}^{T} \ell\left(f_{m,t}(\theta), y_t\right)$$

Theoretical guarantees for functional expert aggregation

Introduction

Expert aggregation

- We have M experts noted $(f_m)_{m=1,\dots,M}$
- For a given input parameter θ, each one provides a prediction $f_{m,t}(\theta)$ for each time step t
- Construct a temporal weight vector ω_t to predict y_t by

$$\widehat{y}_t = \sum_{m=1}^{M} \omega_{m,t} f_{m,t}(\theta) \quad \text{with} \quad \sum_{t=1}^{T} \ell\left(f_{m,t}(\theta), y_t\right)$$

Theoretical analysis

- Need to generate the entire time series from a non-temporal input parameter
- Discretization is numerically necessary but should not be included in the theoritical analysis

Theoretical guarantees for functional expert aggregation

Introduction

Expert aggregation

- We have M experts noted $(f_m)_{m=1,\ldots,M}$
- For a given input parameter θ, each one provides a prediction $f_{m,t}(\theta)$ for each time step t
- Construct a temporal weight vector ω_t to predict y_t by

$$\widehat{y}(t) = \sum_{m=1}^{M} \omega_m(t) f_{m,\theta}(t) \quad \text{with} \quad \int_0^1 \left(\widehat{y}(t) - y_\varphi(t)\right)^2 \mathrm{d}t$$

Theoretical analysis

- Need to generate the entire time series from a non-temporal input parameter
- Discretization is numerically necessary but should not be included in the theoritical analysis

Solution

- Consider time series and experts' weights as continuous functions in time

Theoretical guarantees for functional expert aggregation

Introduction

$$\hat{y}(t) = \sum_{m=1}^{M} \omega_m(t) f_{m,\theta}(t)$$

Theoretical guarantees for functional expert aggregation

Introduction

$$\widehat{y}(t) = \sum_{m=1}^{M} \omega_m(t) f_{m,\theta}(t)$$

Non-constant weights

$\neq$ prevents all linearity arguments

Theoretical guarantees for functional expert aggregation

Introduction

$$\widehat{y}(t) = \sum_{m=1}^{M} \omega_m(t) f_{m,\theta}(t)$$

Non-constant weights
$\neq$ prevents all linearity arguments

Objective

- Define a new framework to generalize expert aggregation to a functional setting with specific
  - Input and output spaces
  - Model
  - Loss
  - Risk

Theoretical guarantees for functional expert aggregation

Introduction

$$\widehat{y}(t) = \sum_{m=1}^{M} \omega_m(t) f_{m,\theta}(t)$$

Non-constant weights

$\neq$ prevents all linearity arguments

Objective

- Define a new framework to generalize expert aggregation to a functional setting with specific
  - Input and output spaces
  - Model
  - Loss
  - Risk

It will permit to use existing optimization tools and guarantee convergence rates of algorithms despite this pseudo-linearity

Theoretical guarantees for functional expert aggregation

Optimization setting

Solve the following optimization problem

$$\operatorname*{arg\,min}_{w \in \mathscr{C}} \left\{ \mathscr{R}(w) \right\} \quad \text{where} \quad \mathscr{R}(w) = \mathbb{E}[\nabla \ell(w)] = \mathbb{E} \int_0^1 \left(\hat{y}(t) - y_\varphi(t) \right)^2 \mathrm{d}t$$

Theoretical guarantees for functional expert aggregation

Optimization setting

Solve the following optimization problem

$$\underset{w \in \mathscr{C}}{\text{arg min}} \left\{ \mathscr{R}(w) \right\} \quad \text{where} \quad \mathscr{R}(w) = \mathbb{E}[\nabla \ell(w)] = \mathbb{E} \int_0^1 \left(\hat{y}(t) - y_\varphi(t) \right)^2 \mathrm{d}t$$

The stochastic projected gradient descent (PGD) algorithm

- The k-th iteration is given by

$$W_{k+1} = \Pi_{\mathscr{C}}\left(W_k - \eta g_k\right)$$

  - g_k is an unbiased estimator of $\nabla \mathscr{R}(W_k)$
  - $\Pi_{\mathscr{C}}$ is the standard orthogonal projection operator on $\mathscr{C}$

Stochastic case: we only have access to $\nabla \ell(w)$ for all w

Theoretical guarantees for functional expert aggregation

Optimization setting

Solve the following optimization problem

$$\underset{w \in \mathscr{C}}{\text{arg min}} \left\{ \mathscr{R}(w) \right\} \quad \text{where} \quad \mathscr{R}(w) = \mathbb{E}[\nabla \ell(w)] = \mathbb{E} \int_0^1 \left(\hat{y}(t) - y_\varphi(t) \right)^2 \mathrm{d}t$$

The stochastic projected gradient descent (PGD) algorithm

- The k-th iteration is given by

$$W_{k+1} = \Pi_{\mathscr{C}}\left(W_k - \eta g_k \right)$$

  - g_k is an unbiased estimator of $\nabla \mathscr{R}(W_k)$
  - $\Pi_{\mathscr{C}}$ is the standard orthogonal projection operator on $\mathscr{C}$

Stochastic case: we only have access to $\nabla \ell(w)$ for all w

Specific framework

+

Needed conditions

1. *Hilbert space to permit convex optimization techniques*
2. *A closed set of contraints*
3. *A μ-strongly convex and ν-smooth loss function*

Theoretical guarantees for functional expert aggregation

Main results with the stochastic PGD algorithm

Proposition 3.5. *Let* $\Pi_{\mathcal{C}}$ *be the orthogonal projection on* $\mathcal{C}$ *and let us assume that*

(i) $\mathbb{E}[g_k]$ *is a subgradient of* $\mathcal{R}$ *at* W_k*,*

(ii) $\mathbb{E}[\|g_k\|^2] \leq B^2$ *for some* $B > 0$*.*

Then, for all $k \geq 1$*,*

$$\mathbb{E}\|W_k - W^*\|^2_{\mathcal{H}} \leq \frac{4B^2}{\mu^2(k+1)}\,.$$

Theoretical guarantees for functional expert aggregation

Main results with the stochastic PGD algorithm

Proposition 3.5. *Let $\Pi_{\mathcal{C}}$ be the orthogonal projection on $\mathcal{C}$ and let us assume that*

(i) $\mathbb{E}[g_k]$ *is a subgradient of* $\mathcal{R}$ *at* W_k*,*

(ii) $\mathbb{E}[\|g_k\|^2] \leq B^2$ *for some* $B > 0$*.*

Then, for all $k \geq 1$*,*

$$\mathbb{E}\|W_k - W^*\|_{\mathcal{H}}^2 \leq \frac{4B^2}{\mu^2(k+1)}.$$

Guarantees can be demonstrated

- In deterministic case
- With mirror gradient descent algorithm

5. General conclusion

General conclusion

Solving the inverse problem

- The sequential Monte Carlo sampler we created specifically for our problem returns very good results, with a clear improvement in the correlation between simulation and reality

General conclusion

Solving the inverse problem

- The sequential Monte Carlo sampler we created specifically for our problem returns very good results, with a clear improvement in the correlation between simulation and reality

Surrogate model

- We have proposed various solutions to solve our forward problem
- Among traditional methods, the best method is convolutional neural networks
- We improved the results by making the approach more complex through the use of hybrid models
- We offer a wide choice of models and each one can decide to focus on either quality or speed

General conclusion

Solving the inverse problem

- The sequential Monte Carlo sampler we created specifically for our problem returns very good results, with a clear improvement in the correlation between simulation and reality

Surrogate model

- We have proposed various solutions to solve our forward problem
- Among traditional methods, the best method is convolutional neural networks
- We improved the results by making the approach more complex through the use of hybrid models
- We offer a wide choice of models and each one can decide to focus on either quality or speed

Theoretical guarantees for functional expert aggregation

- We have developed a framework to generalize expert aggregation to weights and functional experts
- It allows the use of classical optimization algorithms (PGD) while maintaining the same convergence rates in deterministic and stochastic cases
- The application of the mirrror gradient descent and the corresponding proofs are still in progress
- Numerical experiments will be carried out to test the model's performance

Thank you

Appendix

A - Loss function study

- To solve forward or inverse problems, we need to compare time series

What does « two similar time series » means ?

A - Loss function study

- To solve forward or inverse problems, we need to compare time series

What does « two similar time series » means ?

- We need to provide a quantification method adapted to our problem

A - Loss function study

- To solve forward or inverse problems, we need to compare time series

What does « two similar time series » means ?

- We need to provide a quantification method adapted to our problem

There are many possibilities, including

RMSE	**MAE**	**Scaled RMSE**	**Derivative**
$\left(\frac{1}{T}\sum_{t=1}^{T}(y_t - z_t)^2\right)^{1/2}$	$\frac{1}{T}\sum_{t=1}^{T}\lvert y_t - z_t\rvert$	RMSE$(\widetilde{y}, \widetilde{z})$ where $\widetilde{y}$ denotes the scaled vector	$\ell(y, z) + \ell(y', z')$ where y' stands for the discrete derivative
DTW and soft-DTW	**Cross-correlation**	**DILATE**	**Mean, variance, kurtosis and skewness**
$\min_{A \in \mathscr{A}_{T,T}} \langle A, \Delta(y, z)\rangle$ $\min^{\gamma}_{A \in \mathscr{A}_{T,T}} \langle A, \Delta(y, z)\rangle$	$\frac{1}{2T-1}\sum_{k=0}^{2T-2}\sum_{i=0}^{T-1} y_i\, z_{i-k+T-1}$	$\alpha\, \ell_{\text{shape}}(y, z)$ $+(1-\alpha)\, \ell_{\text{temporal}}(y, z)$	$\Big((\mu_y - \mu_z)^2 + (\sigma_y^2 - \sigma_z^2)^2$ $+(k_y - k_z)^2 + (s_y - s_z)^2\Big)^{1/2}$

A - Loss function study

How do we decide which is best?

- For each loss ℓ, among $y_1, \ldots, y_n$, we will determine the best y_i such that
$$\underset{y_i,\ i=1,\ldots,n}{\text{arg min}}\ \ell(y_i, y_\varphi)$$
- We look at the eight selected scenarios and compare them

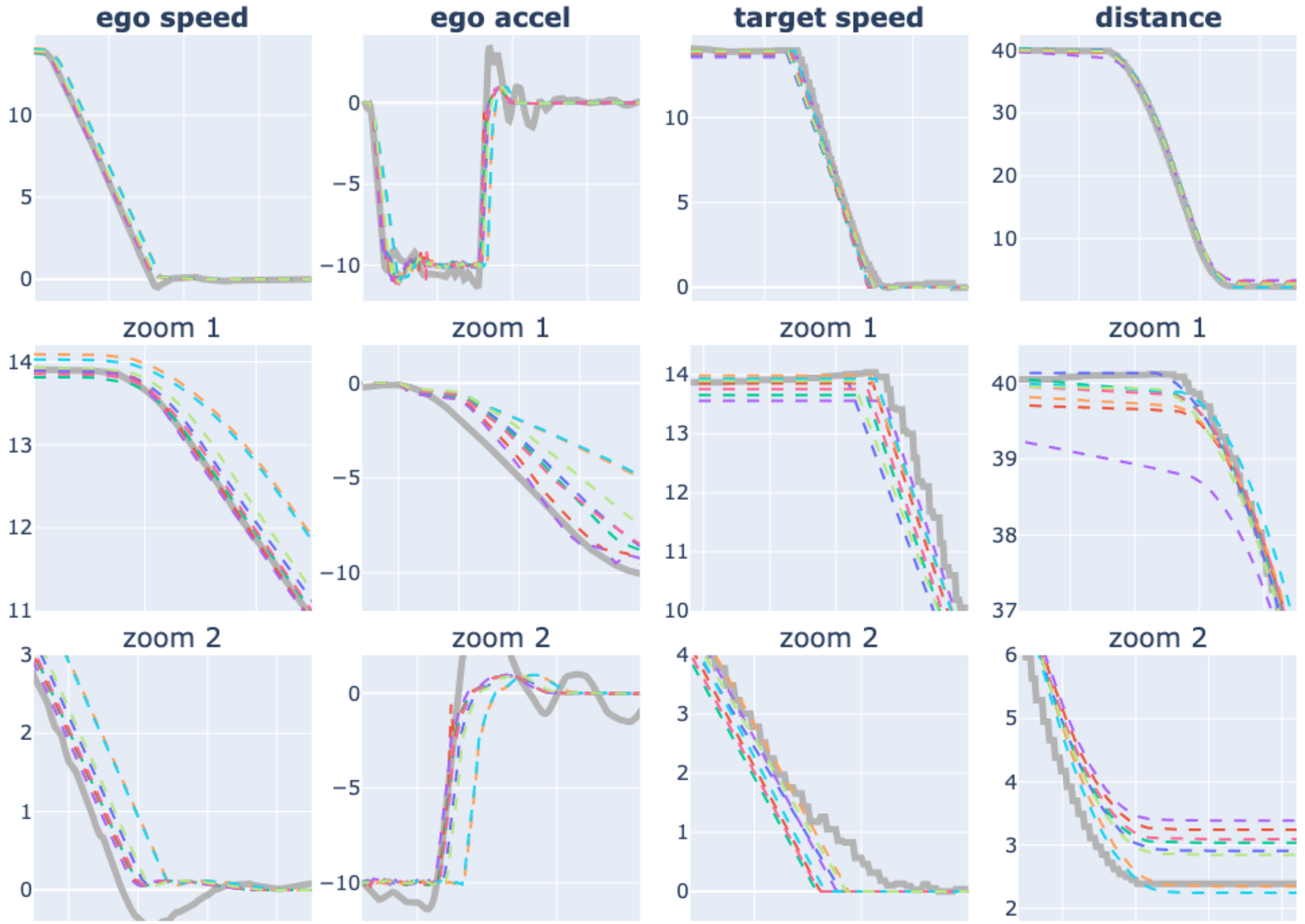
A - Loss function study

How do we decide which is best?

- For each loss ℓ, among $y_1, \ldots, y_n$, we will determine the best y_i such that

$$\underset{y_i,\ i=1,\ldots,n}{\text{arg min}}\ \ell(y_i, y_\varphi)$$

- We look at the eight selected scenarios and compare them



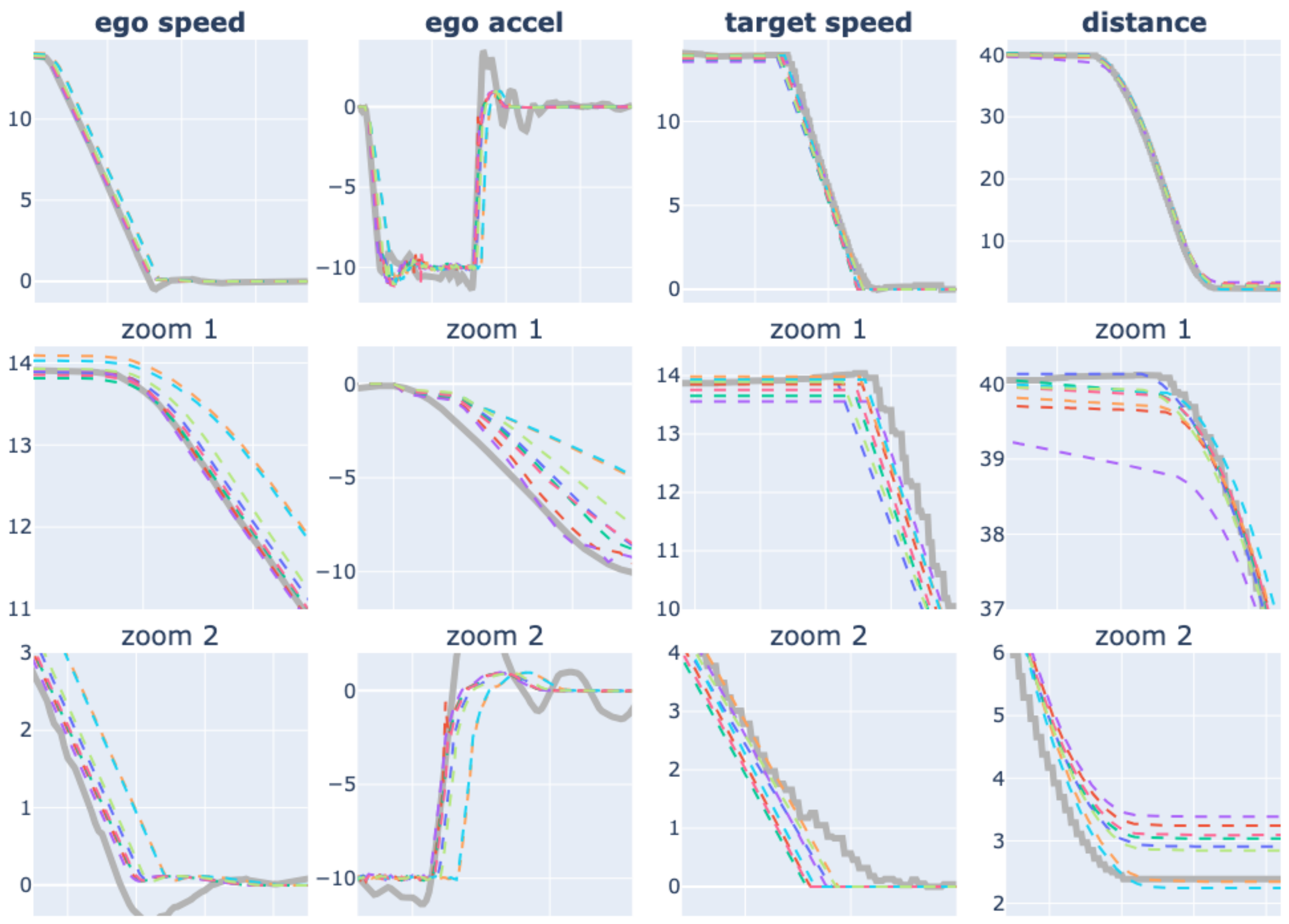
A - Loss function study

How do we decide which is best?

- For each loss ℓ, among $y_1, \ldots, y_n$, we will determine the best y_i such that
$$\underset{y_i,\ i=1,\ldots,n}{\arg\min}\ \ell(y_i, y_\varphi)$$
- We look at the eight selected scenarios and compare them

Overall best choice: scaled RMSE

- By pre-processing the time series to give each one the same weight
- Each one has the same importance in decision making



B - Solving the inverse problem

Mathematical description

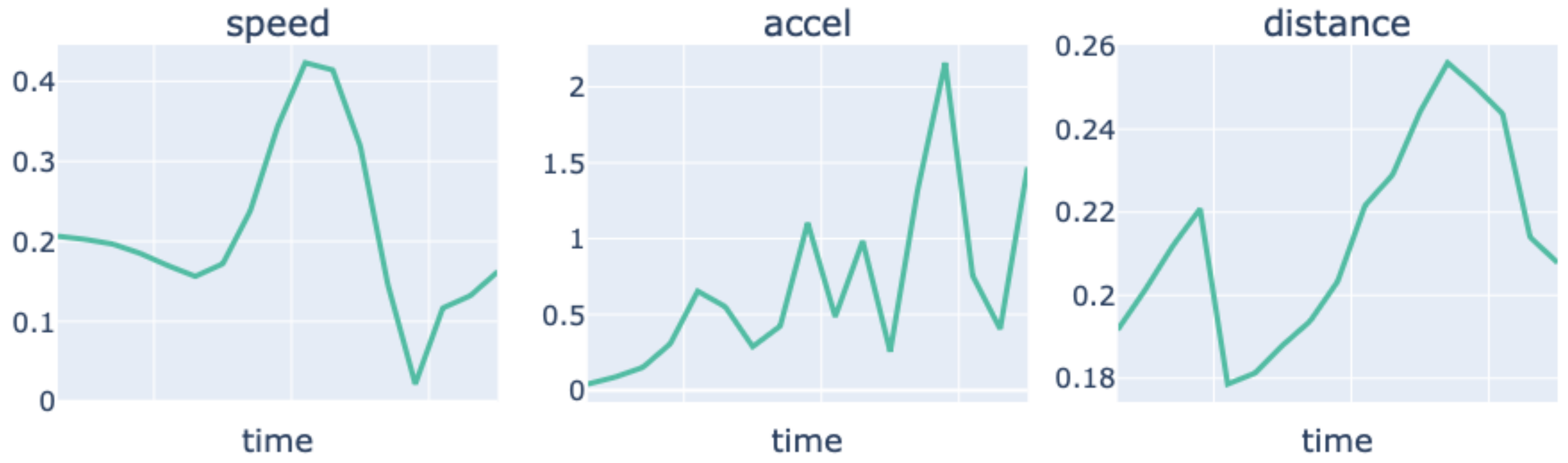
Two hypotheses about the nature of the statistical model noise

B - Solving the inverse problem

Mathematical description

Two hypotheses about the nature of the statistical model noise

1) Gaussian noise assumption

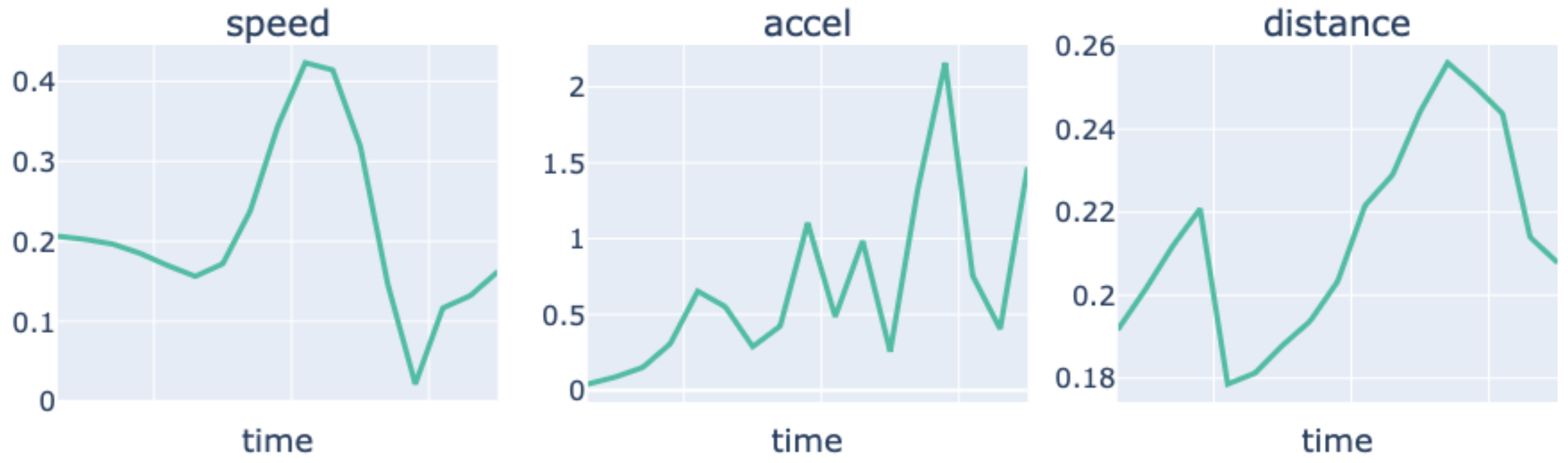


B - Solving the inverse problem

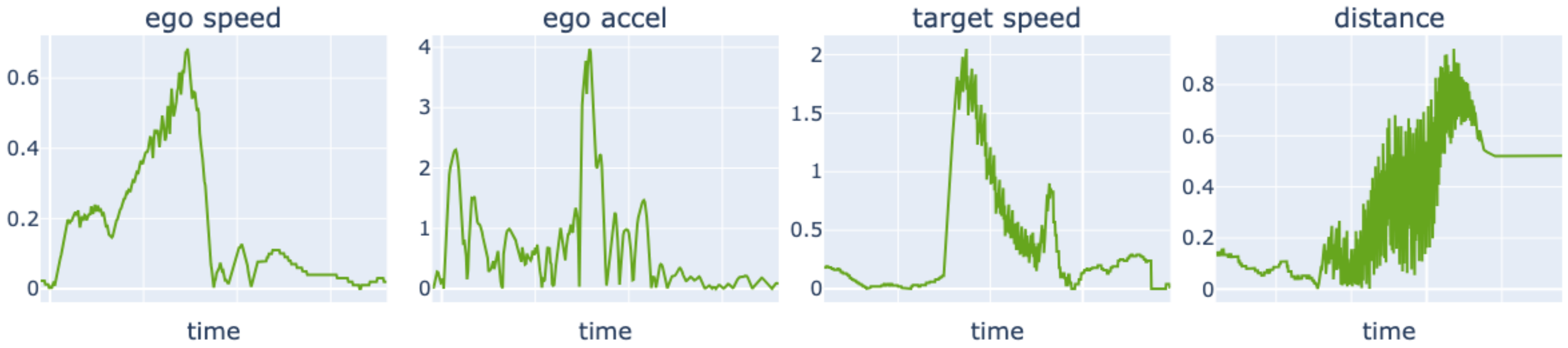
Mathematical description

Two hypotheses about the nature of the statistical model noise

1) Gaussian noise assumption



2) Assumption free



B - Solving the inverse problem

Mathematical description

1) **Gaussian noise assumption**

Let assume that the noise follows a Gaussian distribution

$$\varepsilon \sim \mathcal{N}(0,\Sigma)$$

B - Solving the inverse problem

Mathematical description

1) **Gaussian noise assumption**

Let assume that the noise follows a Gaussian distribution

$$\varepsilon \sim \mathcal{N}(0,\Sigma)$$

(b) If the observed time series is a concatenation of R time series (speed, acceleration, distance, …), we can multiply each squared error by different weights to take into account possible heterogeneity

$$\text{arg min}_{\theta\in\Theta} \left\{ \sum_{i=1}^{n} \text{s-MSE}\big(Y_i, S(\theta)\big) \right\}$$

B - Solving the inverse problem

Mathematical description

1) **Gaussian noise assumption**

Let assume that the noise follows a Gaussian distribution

$$\varepsilon \sim \mathcal{N}(0,\Sigma)$$

(b) If the observed time series is a concatenation of R time series (speed, acceleration, distance, …), we can multiply each squared error by different weights to take into account possible heterogeneity

$$\text{arg min}_{\theta\in\Theta} \left\{ \sum_{i=1}^{n} \text{s-MSE}\big(Y_i, S(\theta)\big) \right\}$$

(c) If Σ is positive semi-definite, we will consider generic methods as Bayesian synthetic likelihood (BSL) by introducing prior knowledge on θ and Σ

B - Solving the inverse problem

Mathematical description

1) Gaussian noise assumption

Let assume that the noise follows a Gaussian distribution

$$\varepsilon \sim \mathcal{N}(0,\Sigma)$$

(a) If $\Sigma = \sigma^2 I_d$, then computing the maximum likelihood estimator is the ordinary least-squares estimator

$$\hat{\theta}_{\text{MLE}} = \text{arg min}_{\theta \in \Theta} \left\{ \sum_{i=1}^{n} \|Y_i - S(\theta)\|_{2,T}^2 \right\} = \text{arg min}_{\theta \in \Theta} \left\{ \sum_{i=1}^{n} \text{MSE}\big(Y_i, S(\theta)\big) \right\}$$

(b) If the observed time series is a concatenation of R time series (speed, acceleration, distance, …), we can multiply each squared error by different weights to take into account possible heterogeneity

$$\text{arg min}_{\theta \in \Theta} \left\{ \sum_{i=1}^{n} \text{s-MSE}\big(Y_i, S(\theta)\big) \right\}$$

(c) If Σ is positive semi-definite, we will consider generic methods as Bayesian synthetic likelihood (BSL) by introducing prior knowledge on θ and Σ

B - Solving the inverse problem

Bayesian approaches - *Bayesian synthetic likelihood (BSL)*

$$\pi(\theta \,|\, y_\varphi) \propto p(y_\varphi \,|\, \theta)\pi_0(\theta)$$

- Likelihood distribution $p(y_\varphi \,|\, \theta)$ is assumed to be Gaussian $\mathcal{N}\left(y_\varphi; S(\theta), \Sigma\right)$

B - Solving the inverse problem

Bayesian approaches - *Bayesian synthetic likelihood (BSL)*

$$\pi(\theta \,|\, y_\varphi) \propto p(y_\varphi \,|\, \theta)\pi_0(\theta)$$

- Likelihood distribution $p(y_\varphi \,|\, \theta)$ is assumed to be Gaussian $\mathcal{N}\left(y_\varphi; S(\theta), \Sigma\right)$
- Parameters $S(\theta)$ and Σ are estimated by estimators

$$S_n = \frac{1}{n}\sum_{i=1}^{n} y_i \qquad \text{and} \qquad \Sigma_n = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - S_n)(y_i - S_n)^\mathsf{T}$$

B - Solving the inverse problem

Bayesian approaches - *Bayesian synthetic likelihood (BSL)*

$$\pi(\theta \mid y_\varphi) \propto p(y_\varphi \mid \theta)\pi_0(\theta)$$

- Likelihood distribution $p(y_\varphi \mid \theta)$ is assumed to be Gaussian $\mathcal{N}\left(y_\varphi; S(\theta), \Sigma\right)$
- Parameters $S(\theta)$ and Σ are estimated by estimators

$$S_n = \frac{1}{n}\sum_{i=1}^{n} y_i \qquad \text{and} \qquad \Sigma_n = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - S_n)(y_i - S_n)^\mathsf{T}$$

- The estimation is then given by

$$\pi_{\mathsf{BSL},n}(\theta \mid y_\varphi) \propto p_n(y_\varphi \mid \theta)\pi_0(\theta)$$

$$\text{where } p_n(y_\varphi \mid \theta) = \int \mathcal{N}(y_\varphi; S_n, \Sigma_n)\prod_{i=1}^{n} p(y_i \mid \theta)\,\mathrm{d}y_{1:n}$$

B - Solving the inverse problem

Bayesian approaches - *Bayesian synthetic likelihood (BSL)*

$$\pi(\theta \,|\, y_\varphi) \propto p(y_\varphi \,|\, \theta)\pi_0(\theta)$$

- Likelihood distribution $p(y_\varphi \,|\, \theta)$ is assumed to be Gaussian $\mathcal{N}\left(y_\varphi; S(\theta), \Sigma\right)$
- Parameters $S(\theta)$ and Σ are estimated by estimators

$$S_n = \frac{1}{n}\sum_{i=1}^{n} y_i \qquad \text{and} \qquad \Sigma_n = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - S_n)(y_i - S_n)^\mathsf{T}$$

- The estimation is then given by

$$\pi_{\mathsf{BSL},n}(\theta \,|\, y_\varphi) \propto p_n(y_\varphi \,|\, \theta)\pi_0(\theta)$$

$$\text{where } p_n(y_\varphi \,|\, \theta) = \int \mathcal{N}(y_\varphi; S_n, \Sigma_n)\prod_{i=1}^{n} p(y_i \,|\, \theta)\,\mathrm{d}y_{1:n}$$

- **Algorithm used:** waste-free sequential Monte Carlo sampler with a tempering version

B - Solving the inverse problem

Constructing and evaluating estimators

B - Solving the inverse problem

Constructing and evaluating estimators

How to transform posterior distribution into a point value estimation?

B - Solving the inverse problem

Constructing and evaluating estimators

How to transform posterior distribution into a point value estimation?

Let $\theta_1, \ldots, \theta_N$ be the final population of accepted particles and $\omega_1, \ldots, \omega_N$ their associated weights

1) Weighted mean
2) Coordinate-wise maximum a posteriori

B - Solving the inverse problem

Constructing and evaluating estimators

How to transform posterior distribution into a point value estimation?

Let $\theta_1, \ldots, \theta_N$ be the final population of accepted particles and $\omega_1, \ldots, \omega_N$ their associated weights

1) Weighted mean
2) Coordinate-wise maximum a posteriori

How to evaluate the quality of the obtained results?

B - Solving the inverse problem

Constructing and evaluating estimators

How to transform posterior distribution into a point value estimation?

Let $\theta_1, \ldots, \theta_N$ be the final population of accepted particles and $\omega_1, \ldots, \omega_N$ their associated weights

1) Weighted mean
2) Coordinate-wise maximum a posteriori

How to evaluate the quality of the obtained results?

- Let $\hat{\theta}$ be the point value estimation obtained at the end and d being the s-RMSE
- Look at the time series generated from $\hat{\theta}$ with the surrogate model and the simulator by computing

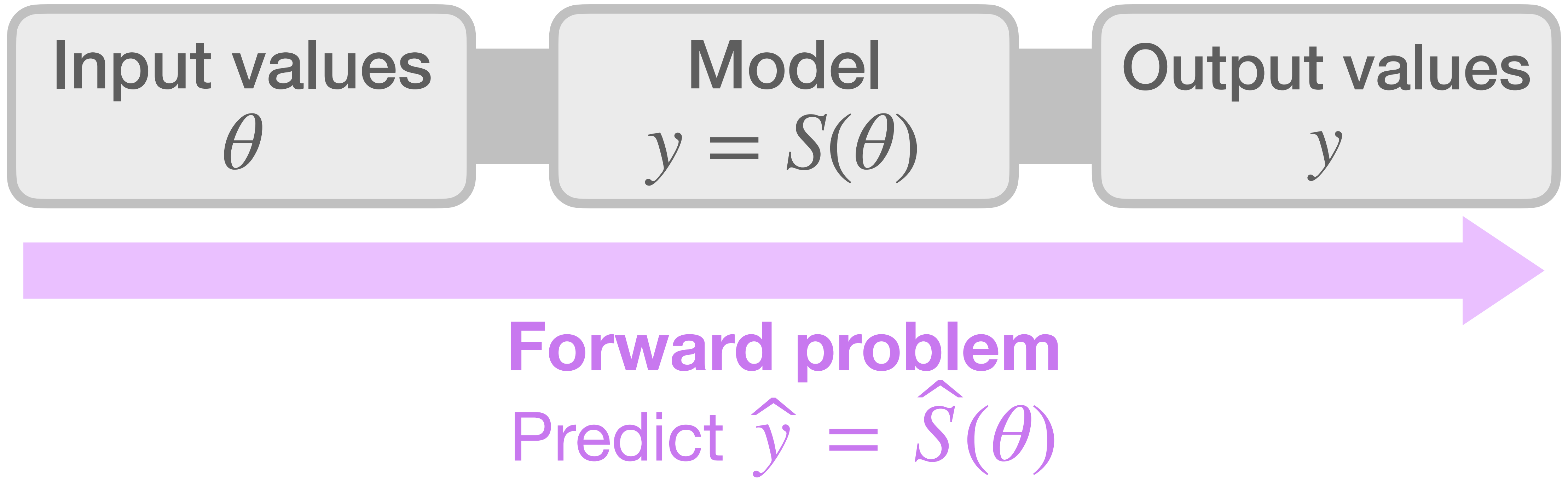
$$d\left(\hat{S}(\hat{\theta}), y_\varphi\right) \quad \text{and} \quad d\left(S(\hat{\theta}), y_\varphi\right)$$

- To better understand why some estimators work better than others, compute

$$\left| d\left(S(\hat{\theta}), y_\varphi\right) - d\left(\hat{S}(\hat{\theta}), y_\varphi\right) \right| \quad \text{and} \quad d\left(S(\hat{\theta}), \hat{S}(\hat{\theta})\right)$$

C - Construction of the surrogate model

Context and objectives of the thesis

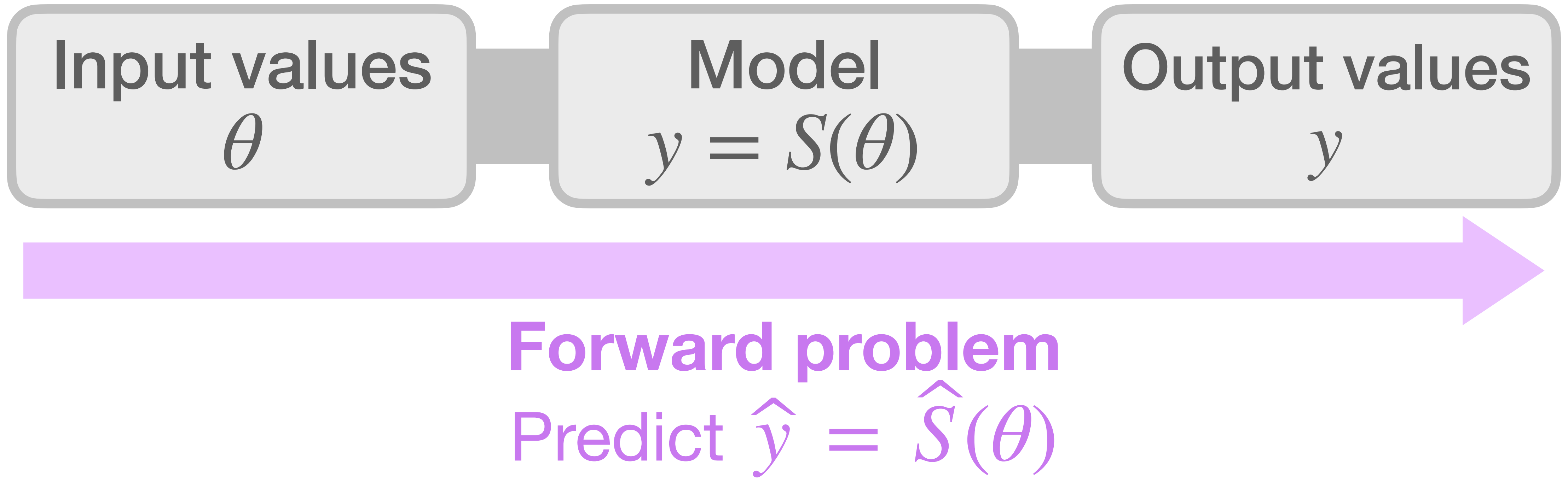


One difficulty araises

- Computing the gradient or the likelihood function is impractical or intractable…
- … but simulating data from the model is feasible

C - Construction of the surrogate model

Context and objectives of the thesis



One difficulty araises

- Computing the gradient or the likelihood function is impractical or intractable…
- … but simulating data from the model is feasible

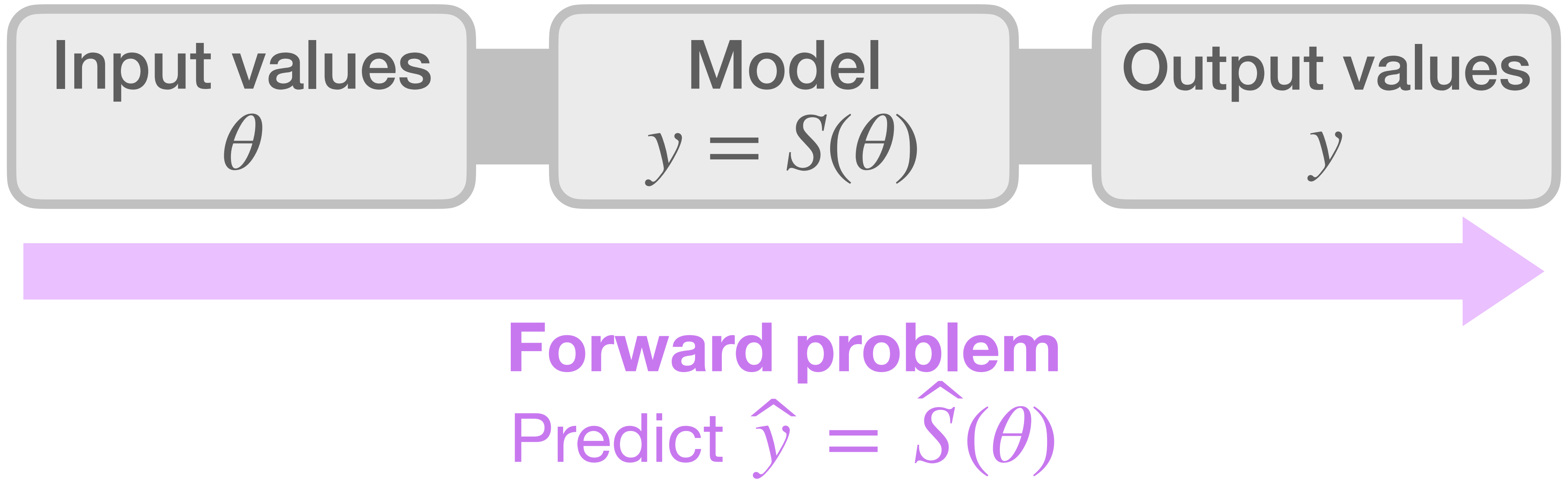
***Gradient-free* or *likelihood-free* methods** are necessaries

- They allowed the estimation of the posterior distribution without explicitly calculating the likelihood
- They require the generation of numerous outputs y linked to candidate inputs θ through the simulator

$$y = S(\theta)$$

C - Construction of the surrogate model

Context and objectives of the thesis



One difficulty araises

- Computing the gradient or the likelihood function is impractical or intractable…
- … but simulating data from the model is feasible

***Gradient-free* or *likelihood-free* methods** are necessaries

- They allowed the estimation of the posterior distribution without explicitly calculating the likelihood
- They require the generation of numerous outputs y linked to candidate inputs θ through the simulator

$$y = S(\theta)$$

How do we generate large amounts of time series?

- Using the simulator S would be optimal but it requires excessive computational resources
- We substitute the simulator S with a pre-trained surrogate model $\widehat{S}$ which mimics it

Predicts y *for a given* θ *using* $\widehat{y} = \widehat{S}(\theta)$

We need to construct a surrogate model using supervised statistical learning to solve a forward problem

C - Construction of the surrogate model

Different data formats

What are the data? How to use it?

C - Construction of the surrogate model

Different data formats

What are the data? How to use it?

Considered outputs are time series

- Time dependency must be taken into account when building our model
- Choose a format that's not too expensive and not too large

C - Construction of the surrogate model

Different data formats

What are the data? How to use it?

Considered outputs are time series

- Time dependency must be taken into account when building our model
- Choose a format that's not too expensive and not too large

Identify two simple formats

C - Construction of the surrogate model

Different data formats

What are the data? How to use it?

Considered outputs are time series

- Time dependency must be taken into account when building our model
- Choose a format that's not too expensive and not too large

Identify two simple formats

- Vector format

$$\begin{pmatrix} \theta_{1,1} & \dots & \theta_{1,p} \\ \theta_{2,1} & \dots & \theta_{2,p} \\ \vdots & & \vdots \\ \theta_{n,1} & \dots & \theta_{n,p} \end{pmatrix} \left(\begin{array}{ccc|c|ccc} y^{(1)}_{1,1} & \dots & y^{(1)}_{1,T_1} & \dots & y^{(R)}_{1,1} & \dots & y^{(R)}_{1,T_R} \\ y^{(1)}_{2,1} & \dots & y^{(1)}_{2,T_1} & \dots & y^{(R)}_{2,1} & \dots & y^{(R)}_{2,T_R} \\ \vdots & & \vdots & & \vdots & & \vdots \\ y^{(1)}_{n,1} & \dots & y^{(1)}_{n,T_1} & \dots & y^{(R)}_{n,1} & \dots & y^{(R)}_{n,T_R} \end{array}\right)$$

C - Construction of the surrogate model

Different data formats

What are the data? How to use it?

Considered outputs are time series

- Time dependency must be taken into account when building our model
- Choose a format that's not too expensive and not too large

Identify two simple formats

- Vector format

$$
\begin{pmatrix}
\theta_{1,1} & \cdots & \theta_{1,p} \\
\theta_{2,1} & \cdots & \theta_{2,p} \\
\vdots & & \vdots \\
\theta_{n,1} & \cdots & \theta_{n,p}
\end{pmatrix}
\left(\begin{array}{ccc|c|ccc}
y^{(1)}_{1,1} & \cdots & y^{(1)}_{1,T_1} & \cdots & y^{(R)}_{1,1} & \cdots & y^{(R)}_{1,T_R} \\
y^{(1)}_{2,1} & \cdots & y^{(1)}_{2,T_1} & \cdots & y^{(R)}_{2,1} & \cdots & y^{(R)}_{2,T_R} \\
\vdots & & \vdots & & \vdots & & \vdots \\
y^{(1)}_{n,1} & \cdots & y^{(1)}_{n,T_1} & \cdots & y^{(R)}_{n,1} & \cdots & y^{(R)}_{n,T_R}
\end{array}\right)
$$

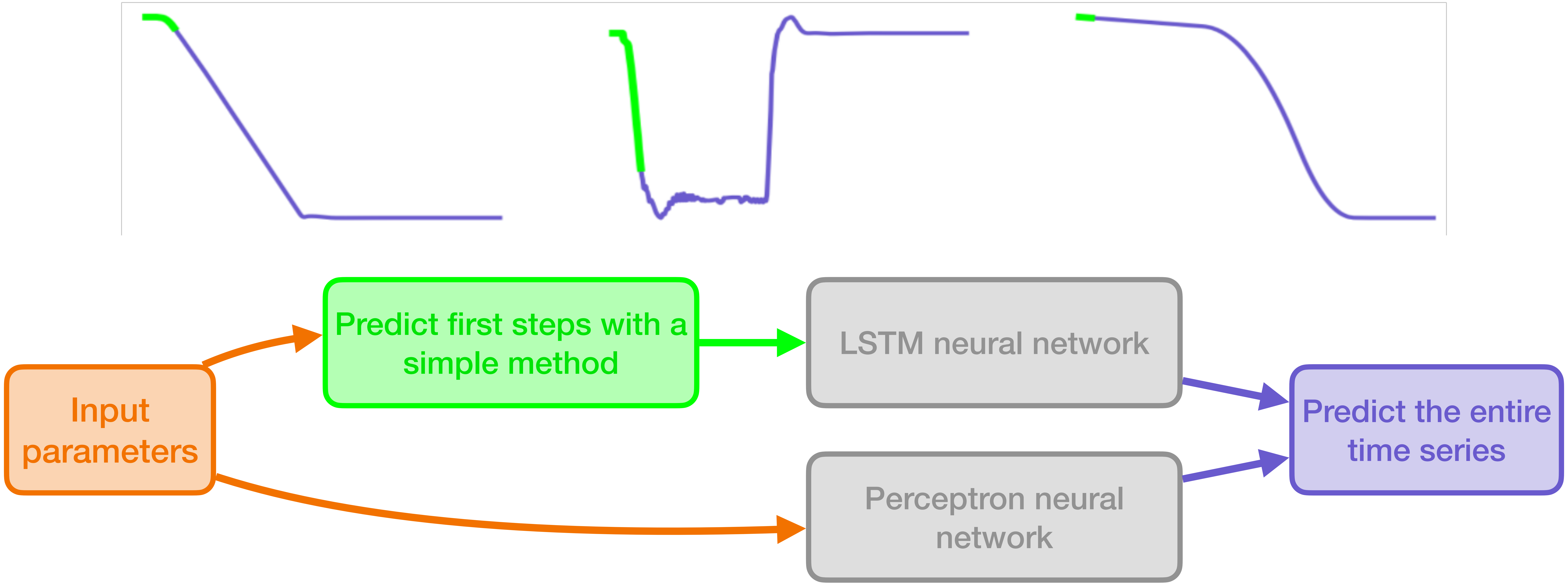
- Matrix format

$$
\left(\begin{array}{ccccc}
\theta_{1,1} & \theta_{1,2} & \cdots & \theta_{1,p} & 1 \\
'' & '' & \cdots & '' & 2 \\
\vdots & \vdots & & \vdots & \vdots \\
'' & '' & \cdots & '' & T \\
\hline
 & & \cdots & & \\
\hline
\theta_{n,1} & \theta_{n,2} & \cdots & \theta_{n,p} & 1 \\
'' & '' & \cdots & '' & 2 \\
\vdots & \vdots & & \vdots & \vdots \\
'' & '' & \cdots & '' & T
\end{array}\right)
\left(\begin{array}{cccc}
y^{(1)}_{1,1} & y^{(2)}_{1,1} & \cdots & y^{(R)}_{1,1} \\
\vdots & \vdots & & \vdots \\
\vdots & \vdots & & \vdots \\
y^{(1)}_{1,T} & y^{(2)}_{1,T} & \cdots & y^{(R)}_{1,T} \\
\hline
 & \cdots & \cdots & \\
\hline
y^{(1)}_{n,1} & y^{(2)}_{n,1} & \cdots & y^{(R)}_{n,1} \\
\vdots & \vdots & & \vdots \\
\vdots & \vdots & & \vdots \\
y^{(1)}_{n,T} & y^{(2)}_{n,T} & \cdots & y^{(R)}_{n,T}
\end{array}\right)
$$

C - Construction of the surrogate model

Neural networks

- Long short-term memory (LSTM)
  - Handles sequential data
  - Maintains a hidden space which acts as a memory of previous inputs



C - Construction of the surrogate model

Conclusion - *Numerical results*

RMSE ($\times 10^{-1}$)	1-RF	4-RF	matrix 1-RF	matrix 4-RF	PCA 4-RF	Sparse PCA 4-RF	fPCA 4-RF
train	1.108	0.998	0.310	**0.201**	1.051	1.136	1.050
validation	2.535	2.243	2.688	2.692	2.572	2.598	2.583
test	3.308	3.047	4.012	3.999	3.465	3.481	3.473

RMSE ($\times 10^{-1}$)	k-NN	KRR	DF	CNN	multi LSTM	hybrid	aggre-gated
train	2.693	0.917	1.221	0.500	0.463	$\times$	$\times$
validation	2.966	3.234	1.791	0.500	0.507	**0.422**	0.996
test	3.903	4.009	2.791	**0.558**	0.665	0.607	1.484

C - Construction of the surrogate model

Conclusion - *Numerical results*

RMSE $(\times 10^{-1})$	1-RF	4-RF	matrix 1-RF	matrix 4-RF	PCA 4-RF	Sparse PCA 4-RF	fPCA 4-RF
train	1.108	0.998	0.310	**0.201**	1.051	1.136	1.050
validation	2.535	2.243	2.688	2.692	2.572	2.598	2.583
test	3.308	3.047	4.012	3.999	3.465	3.481	3.473

RMSE $(\times 10^{-1})$	k-NN	KRR	DF	CNN	multi LSTM	hybrid	aggre-gated
train	2.693	0.917	1.221	0.500	0.463	$\times$	$\times$
validation	2.966	3.234	1.791	0.500	0.507	**0.422**	0.996
test	3.903	4.009	2.791	**0.558**	0.665	0.607	1.484

Traditional ML

C - Construction of the surrogate model

Conclusion - *Numerical results*

Principal components analysis

RMSE $(\times 10^{-1})$	1-RF	4-RF	matrix 1-RF	matrix 4-RF	PCA 4-RF	Sparse PCA 4-RF	fPCA 4-RF
train	1.108	0.998	0.310	**0.201**	1.051	1.136	1.050
validation	2.535	2.243	2.688	2.692	2.572	2.598	2.583
test	3.308	3.047	4.012	3.999	3.465	3.481	3.473

RMSE $(\times 10^{-1})$	k-NN	KRR	DF	CNN	multi LSTM	hybrid	aggre-gated
train	2.693	0.917	1.221	0.500	0.463	$\times$	$\times$
validation	2.966	3.234	1.791	0.500	0.507	**0.422**	0.996
test	3.903	4.009	2.791	**0.558**	0.665	0.607	1.484

Traditional ML

C - Construction of the surrogate model

Conclusion - *Numerical results*

Principal components analysis

RMSE $(\times 10^{-1})$	1-RF	4-RF	matrix 1-RF	matrix 4-RF	PCA 4-RF	Sparse PCA 4-RF	fPCA 4-RF
train	1.108	0.998	0.310	**0.201**	1.051	1.136	1.050
validation	2.535	2.243	2.688	2.692	2.572	2.598	2.583
test	3.308	3.047	4.012	3.999	3.465	3.481	3.473

RMSE $(\times 10^{-1})$	k-NN	KRR	DF	CNN	multi LSTM	hybrid	aggre-gated
train	2.693	0.917	1.221	0.500	0.463	$\times$	$\times$
validation	2.966	3.234	1.791	0.500	0.507	**0.422**	0.996
test	3.903	4.009	2.791	**0.558**	0.665	0.607	1.484

Traditional ML

Neural networks

C - Construction of the surrogate model

Conclusion - *Numerical results*

Principal components analysis

RMSE $(\times 10^{-1})$	1-RF	4-RF	matrix 1-RF	matrix 4-RF	PCA 4-RF	Sparse PCA 4-RF	fPCA 4-RF
train	1.108	0.998	0.310	**0.201**	1.051	1.136	1.050
validation	2.535	2.243	2.688	2.692	2.572	2.598	2.583
test	3.308	3.047	4.012	3.999	3.465	3.481	3.473

RMSE $(\times 10^{-1})$	k-NN	KRR	DF	CNN	multi LSTM	hybrid	aggre-gated
train	2.693	0.917	1.221	0.500	0.463	$\times$	$\times$
validation	2.966	3.234	1.791	0.500	0.507	**0.422**	0.996
test	3.903	4.009	2.791	**0.558**	0.665	0.607	1.484

Traditional ML

Neural networks

Hybrid and aggregated

C - Construction of the surrogate model

Conclusion - *Computation times*

time	1-RF	4-RF	matrix 1-RF	matrix 4-RF	PCA 4-RF	Sparse PCA 4-RF	fPCA 4-RF
training	2 min	2 min	30 min	2 hours	$<$ 1 min	16 min	$<$ 1 min
prediction	5 ms	8 ms	144 ms	484 ms	44 ms	412 ms	436 ms

time	k-NN	KRR	DF	CNN	multi LSTM	hybrid	aggre-gated
training	1 sec	1 sec	21 min	6 hours	33 min	5 sec (+)	4 min (+)
prediction	1 ms	$<$ 1 ms	26 ms	3 ms	27 ms	40 ms	6 sec

D - Theoretical guarantees for functional expert aggregation

Framework overall description

By defining an appropriate framework and under certain conditions, it permits to

D - Theoretical guarantees for functional expert aggregation

Framework overall description

By defining an appropriate framework and under certain conditions, it permits to

- Use the existing optimization tools to compute the best W such as

$$W^* = \operatorname*{arg\,min}_{W \in \mathscr{C}} \left\{ \mathscr{R}(W) := \mathbb{E}_P[\ell(W, X)] \right\}$$

D - Theoretical guarantees for functional expert aggregation

Framework overall description

By defining an appropriate framework and under certain conditions, it permits to

- Use the existing optimization tools to compute the best W such as

$$W^* = \underset{W \in \mathscr{C}}{\arg\min} \left\{ \mathscr{R}(W) := \mathbb{E}_P[\ell(W, X)] \right\}$$

- Guarantee convergence rates of deterministic and stochastic algorithms with little or no proof modification

D - Theoretical guarantees for functional expert aggregation

Framework overall description

By defining an appropriate framework and under certain conditions, it permits to

- Use the existing optimization tools to compute the best W such as

$$W^* = \underset{W \in \mathscr{C}}{\text{arg min}} \left\{ \mathscr{R}(W) := \mathbb{E}_P[\ell(W, X)] \right\}$$

- Guarantee convergence rates of deterministic and stochastic algorithms with little or no proof modification

Needed conditions

1. Embed the space $\mathscr{W}$ in which W resides into a Hilbert space to permit convex optimization techniques
2. The set of contraints $\mathscr{C}$ needs to be closed
3. The loss function ℓ needs to be μ-strongly convex and ν-smooth

D - Theoretical guarantees for functional expert aggregation

Framework overall description

- **Data**: (x, y) where $x \in \mathbb{R}^p$ is a non-temporal input and $y \in \mathcal{S}$ is a function of time where $\mathcal{S}$ is the set of real-valued twice constinuously differentiable functions on the interval $[0,1]$

$$\mathcal{S} = C^2([0,1], \mathbb{R})$$

D - Theoretical guarantees for functional expert aggregation

Framework overall description

- **Data**: (x, y) where $x \in \mathbb{R}^p$ is a non-temporal input and $y \in \mathcal{S}$ is a function of time where $\mathcal{S}$ is the set of real-valued twice constinuously differentiable functions on the interval $[0,1]$

$$\mathcal{S} = C^2([0,1], \mathbb{R})$$

- **Functionals**: Maps from $\mathbb{R}^p$ to $\mathcal{S}$ denoted f such as f_x is defined by

$$\begin{array}{rccc} f_x : & [0,1] & \to & \mathbb{R} \\ & t & \mapsto & f(x)(t) \end{array}$$

D - Theoretical guarantees for functional expert aggregation

Framework overall description

- **Data**: (x, y) where $x \in \mathbb{R}^p$ is a non-temporal input and $y \in \mathcal{S}$ is a function of time where $\mathcal{S}$ is the set of real-valued twice constinuously differentiable functions on the interval $[0,1]$

$$\mathcal{S} = C^2([0,1], \mathbb{R})$$

- **Functionals**: Maps from $\mathbb{R}^p$ to $\mathcal{S}$ denoted f such as f_x is defined by

$$\begin{array}{cccc} f_x : & [0,1] & \to & \mathbb{R} \\ & t & \mapsto & f(x)(t) \end{array}$$

- **Renault's digital simulator**: Consider an unknown functional defined as follows

$$\begin{array}{cccc} F^* : & \mathbb{R}^p & \to & \mathcal{S} \\ & x & \mapsto & F^*(x) =: F^*_x \end{array}$$

which maps non-temporal input values x to output time series F^*_x

D - Theoretical guarantees for functional expert aggregation

Framework overall description

To estimate F^*, we aggregate experts with weights

- Let the experts $f_1, \ldots, f_M$ be M functionals
- $\mathscr{W}$ be a subspace of twice continuously differentiable maps $[0,1] \to \Delta^M$ where Δ^M denotes the simplex of $\mathbb{R}^M$

D - Theoretical guarantees for functional expert aggregation

Framework overall description

To estimate F^*, we aggregate experts with weights

- Let the experts $f_1, \ldots, f_M$ be M functionals
- $\mathscr{W}$ be a subspace of twice continuously differentiable maps $[0,1] \to \Delta^M$ where Δ^M denotes the simplex of $\mathbb{R}^M$
- Let $\mathscr{F} = \{F_W,\ W \in \mathscr{W}\}$ denote the set of aggregates defined by the functionals F_W such as

$$F_{W,x} : t \in [0,1] \mapsto \sum_{j=1}^{M} w_j(t) f_{j,x}(t)$$

where $W(t) = (w_1(t), \ldots, w_M(t)) \in \mathscr{W}$ and $x \in \mathbb{R}^p$

D - Theoretical guarantees for functional expert aggregation

Framework overall description

To estimate F^*, we aggregate experts with weights

- Let the experts $f_1, \ldots, f_M$ be M functionals
- $\mathscr{W}$ be a subspace of twice continuously differentiable maps $[0,1] \to \Delta^M$ where Δ^M denotes the simplex of $\mathbb{R}^M$
- Let $\mathscr{F} = \{F_W,\ W \in \mathscr{W}\}$ denote the set of aggregates defined by the functionals F_W such as

$$F_{W,x} : t \in [0,1] \mapsto \sum_{j=1}^{M} w_j(t) f_{j,x}(t)$$

where $W(t) = (w_1(t), \ldots, w_M(t)) \in \mathscr{W}$ and $x \in \mathbb{R}^p$

Non-constant weights
$\neq$ prevents all linearity arguments

D - Theoretical guarantees for functional expert aggregation

Framework overall description

To estimate F^*, we aggregate experts with weights

- Let the experts $f_1, \ldots, f_M$ be M functionals
- $\mathscr{W}$ be a subspace of twice continuously differentiable maps $[0,1] \to \Delta^M$ where Δ^M denotes the simplex of $\mathbb{R}^M$
- Let $\mathscr{F} = \{F_W,\ W \in \mathscr{W}\}$ denote the set of aggregates defined by the functionals F_W such as

$$F_{W,x} : t \in [0,1] \mapsto \sum_{j=1}^{M} w_j(t) f_{j,x}(t)$$

where $W(t) = (w_1(t), \ldots, w_M(t)) \in \mathscr{W}$ and $x \in \mathbb{R}^p$

Non-constant weights
$\neq$ prevents all linearity arguments

We need to take into account this pseudo-linearity and update arguments if needed

D - Theoretical guarantees for functional expert aggregation

Framework overall description

Loss to consider

D - Theoretical guarantees for functional expert aggregation

Framework overall description

Loss to consider

- The set $\mathcal{S} = C^2([0,1], \mathbb{R})$ is endowed with the L^2-norm

$$\forall y \in \mathcal{S} \quad \|y\|_{L^2}^2 = \int_0^1 y(t)^2 \mathrm{d}t$$

D - Theoretical guarantees for functional expert aggregation

Framework overall description

Loss to consider

- The set $\mathcal{S} = C^2([0,1], \mathbb{R})$ is endowed with the L^2-norm

$$\forall y \in \mathcal{S} \quad \|y\|_{L^2}^2 = \int_0^1 y(t)^2 \mathrm{d}t$$

- For any $W \in \mathscr{W}$ and observation $z = (x, y)$, the loss of the aggregate F_W is

$$\ell(W, z) = \left\| F_{W,x} - y \right\|_{L^2}^2 = \left\| \sum_{j=1}^{M} w_j f_{j,x} - y \right\|_{L^2}^2$$

D - Theoretical guarantees for functional expert aggregation

Framework overall description

Loss to consider

- The set $\mathcal{S} = C^2([0,1], \mathbb{R})$ is endowed with the L^2-norm

$$\forall y \in \mathcal{S} \quad \|y\|_{L^2}^2 = \int_0^1 y(t)^2 \mathrm{d}t$$

- For any $W \in \mathcal{W}$ and observation $z = (x, y)$, the loss of the aggregate F_W is

$$\ell(W, z) = \left\| F_{W,x} - y \right\|_{L^2}^2 = \left\| \sum_{j=1}^{M} w_j f_{j,x} - y \right\|_{L^2}^2$$

- The risk function is defined by

$$\mathcal{R}(W) = \mathbb{E}\|F_{W,X} - Y\|_{L^2}^2$$

where the expectation is taken with respect to the joint distribution of the random vector X and random function Y