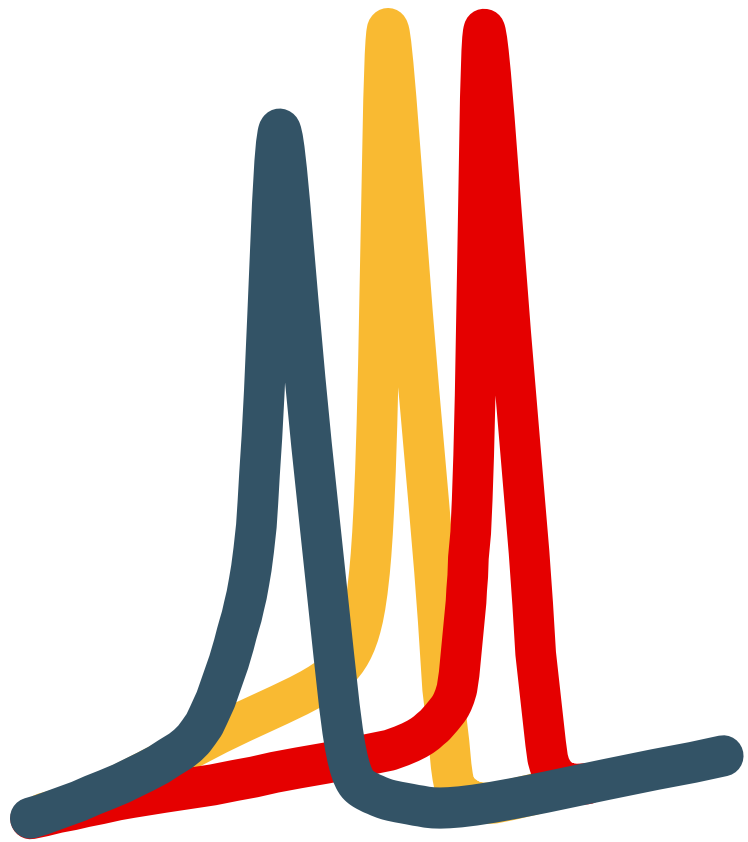
How to model effects of uncertain model parameters with Uncertainpy

Simen Tennøe

University of Oslo, CINPLA



cinpla

Overview of this talk

Why do we need uncertainty quantification and what is it

Overview of this talk

?

Why do we need uncertainty quantification and what is it

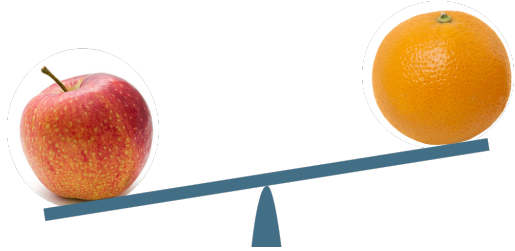
How to perform an uncertainty quantification

Overview of this talk

?

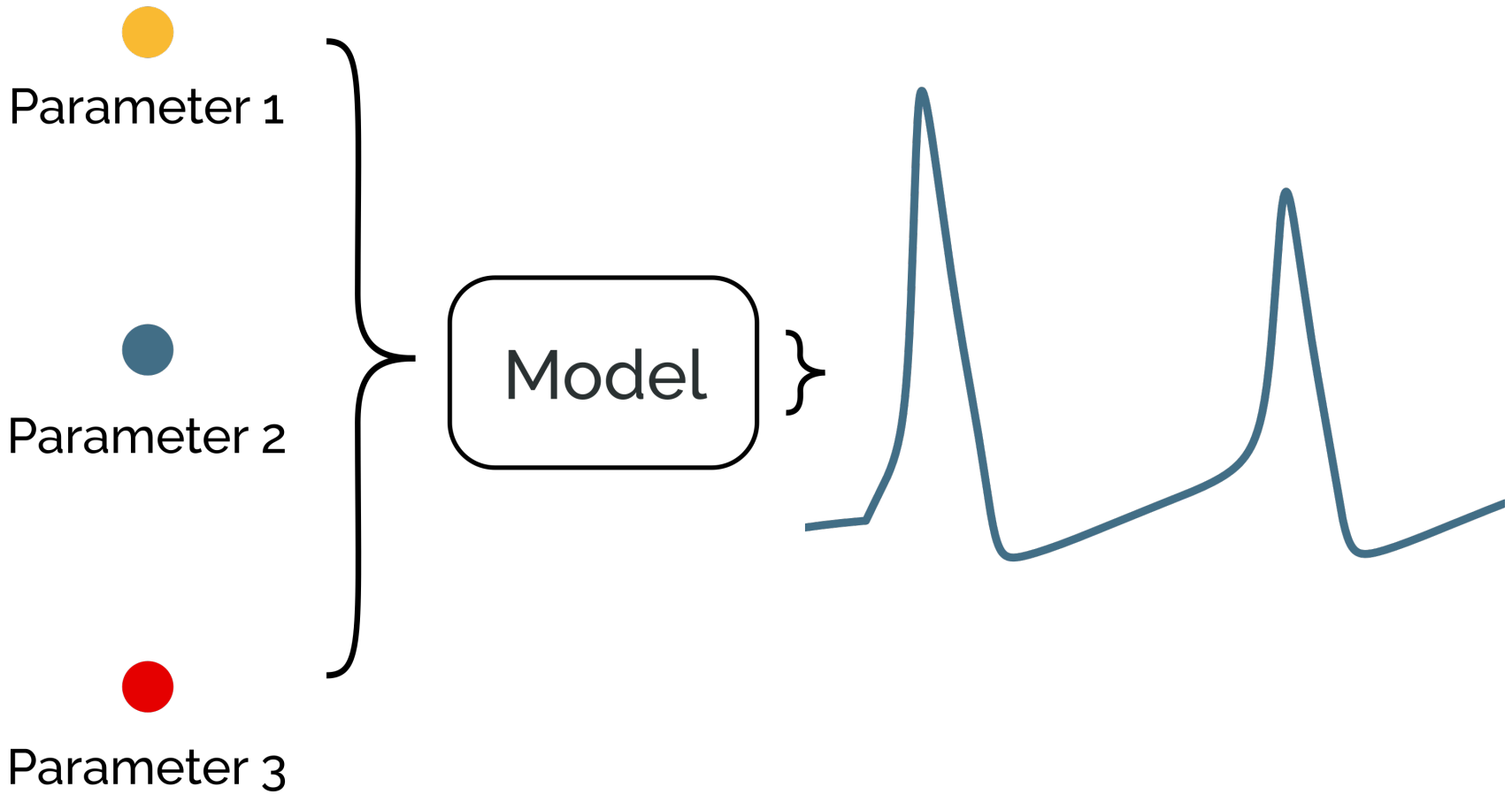
Why do we need uncertainty quantification and what is it

How to perform an uncertainty quantification



Comparison problems

Computational models contain parameters



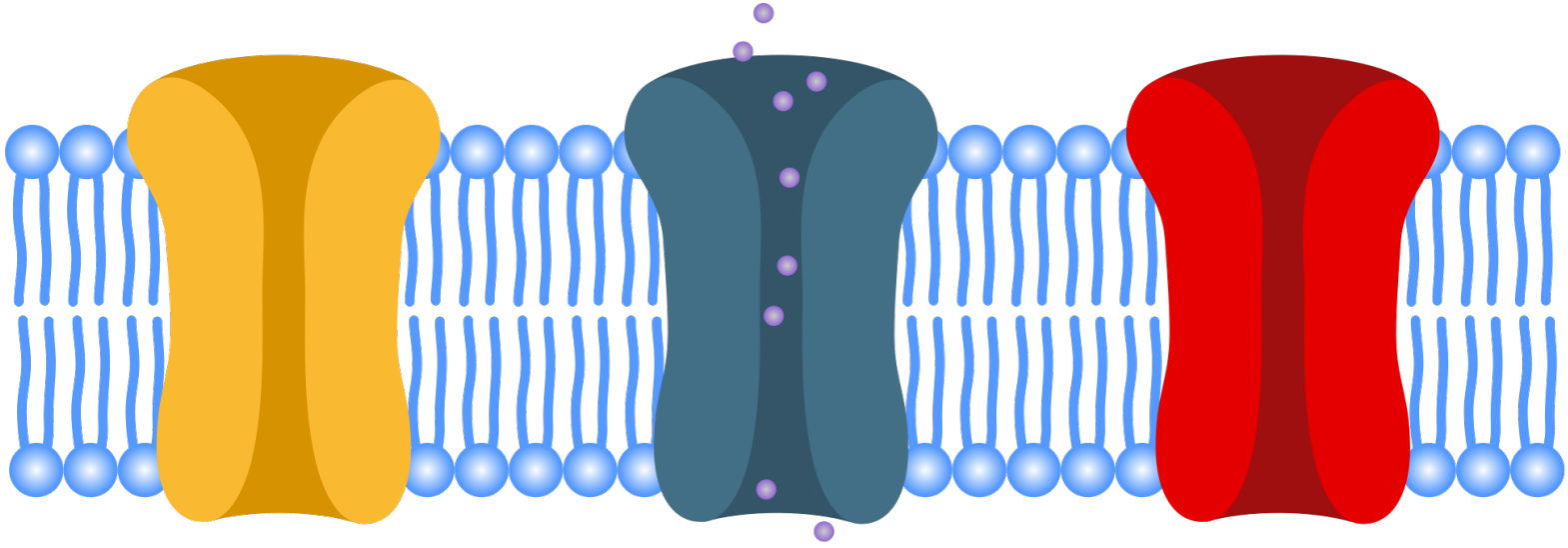
The example we use is the Hodgkin-Huxley model

$$I = C_m \frac{\mathrm{d}V_m}{\mathrm{d}t} + \bar{g}_{\mathrm{K}}(\ldots) + \bar{g}_{\mathrm{Na}}(\ldots) + \bar{g}_{\mathrm{l}}(\ldots)$$

The example we use is the Hodgkin-Huxley model

$$I = C_m \frac{\mathrm{d}V_m}{\mathrm{d}t} + \overbrace{\bar{g}_K(\ldots) + \bar{g}_{Na}(\ldots) + \bar{g}_l(\ldots)}^{\text{Current through the ion channels}}$$

The Hodgkin-Huxley model has three types of ion channels



$\overline{g}_{\mathrm{Na}}$

Potassium conductance

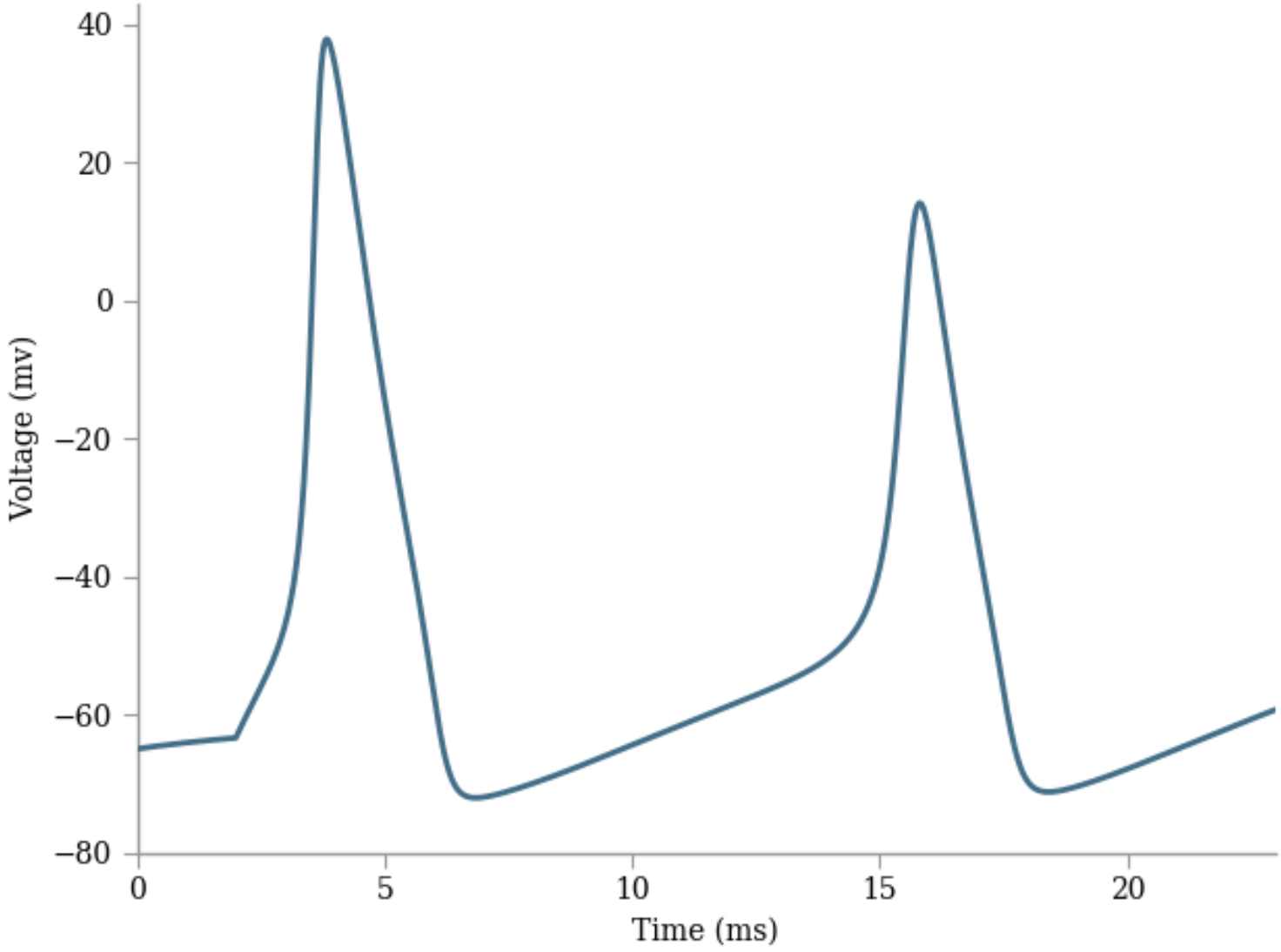
$\overline{g}_{\mathrm{K}}$

Sodium conductance

$\overline{g}_{\mathrm{l}}$

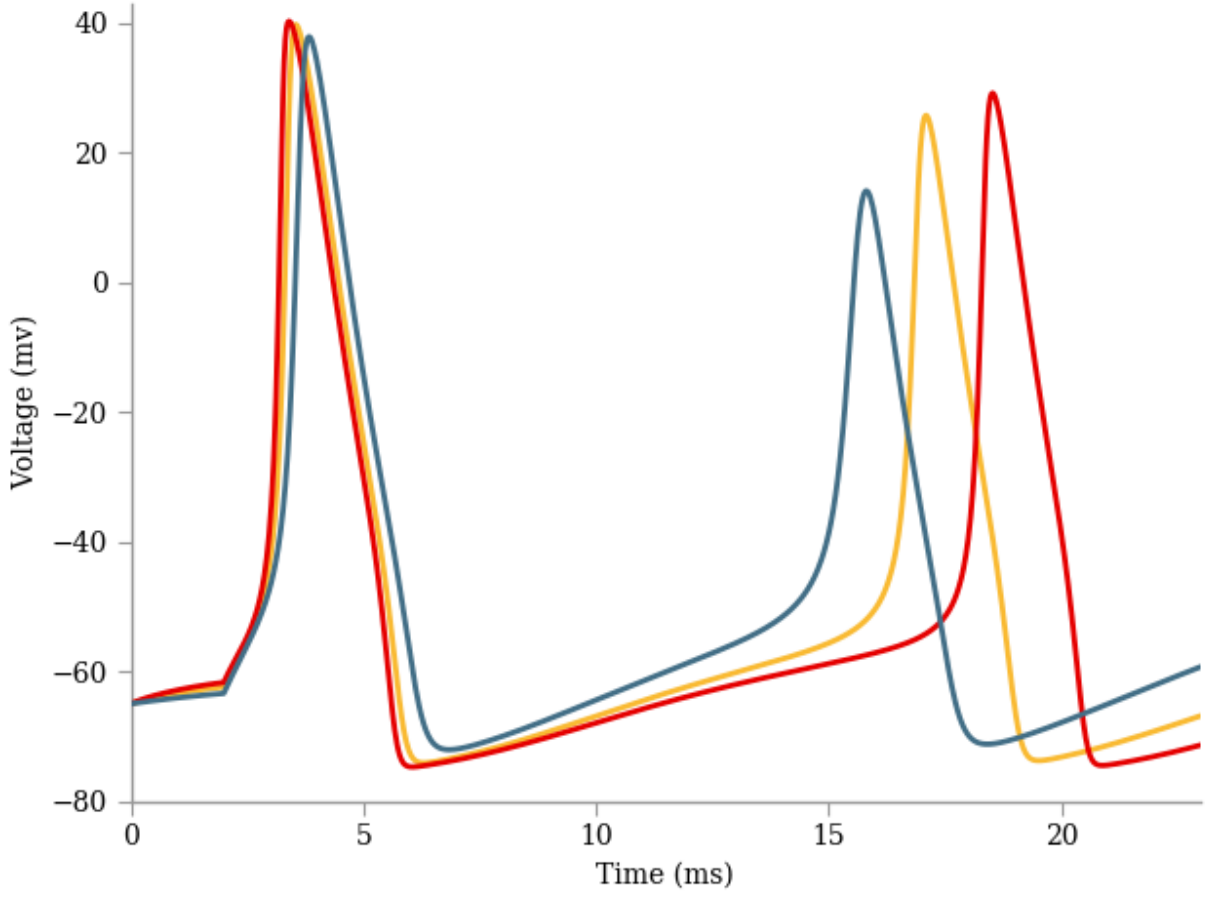
Leak conductance

Membrane potential of the Hodgkin-Huxley model

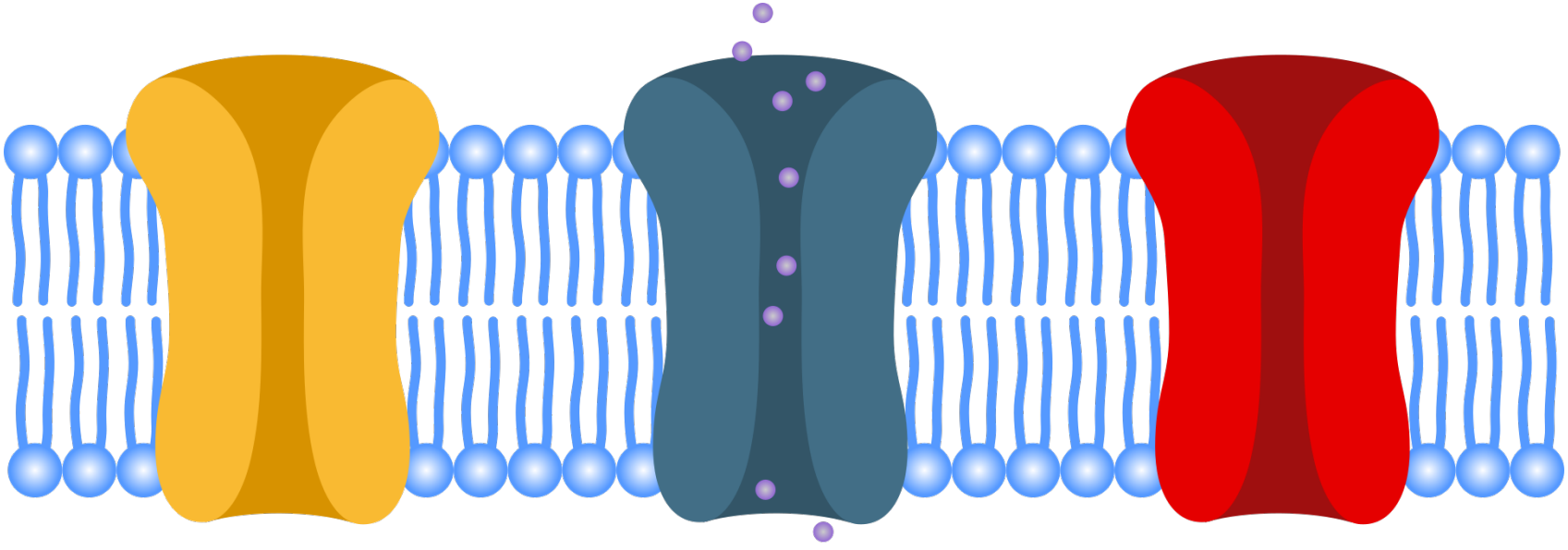


Changing the parameters give different results

$$I = C_m \frac{\mathrm{d}V_m}{\mathrm{d}t} + \overline{g}_{\mathrm{K}}(\ldots) + \overline{g}_{\mathrm{Na}}(\ldots) + \overline{g}_{\mathrm{l}}(\ldots)$$

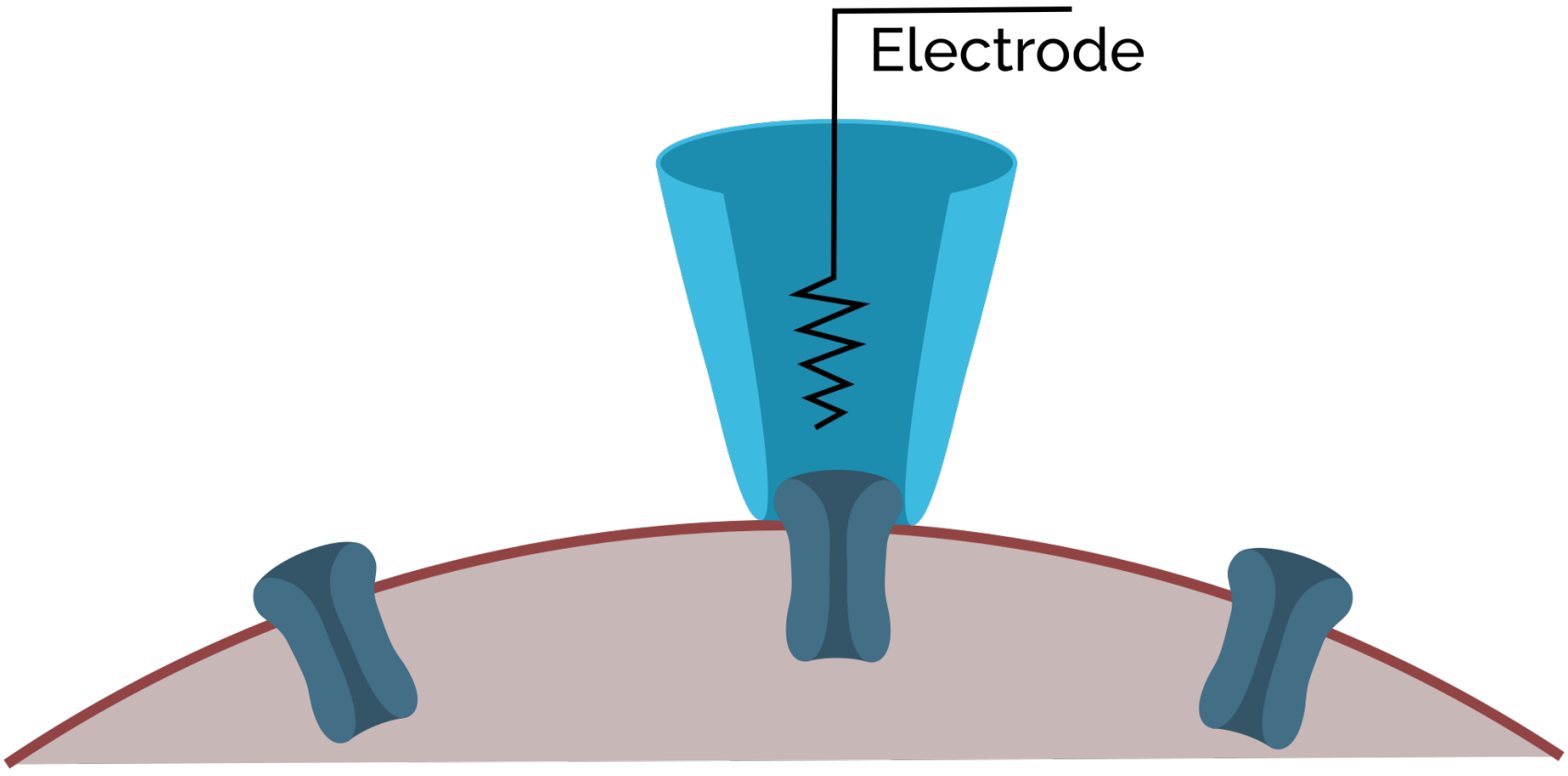


The parameters do not have exact fixed values



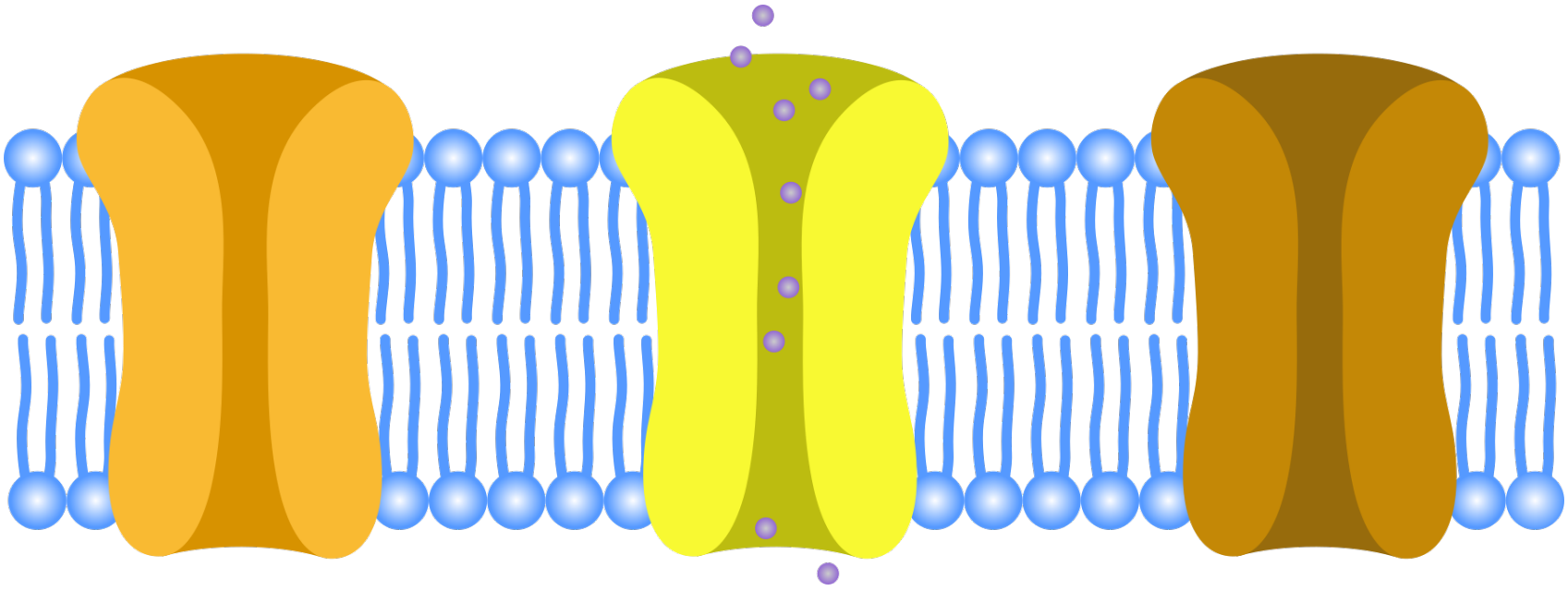
$\overline{g}_{\mathrm{Na}}$ $\overline{g}_{\mathrm{K}}$ $\overline{g}_{\mathrm{l}}$

Measurement uncertainty

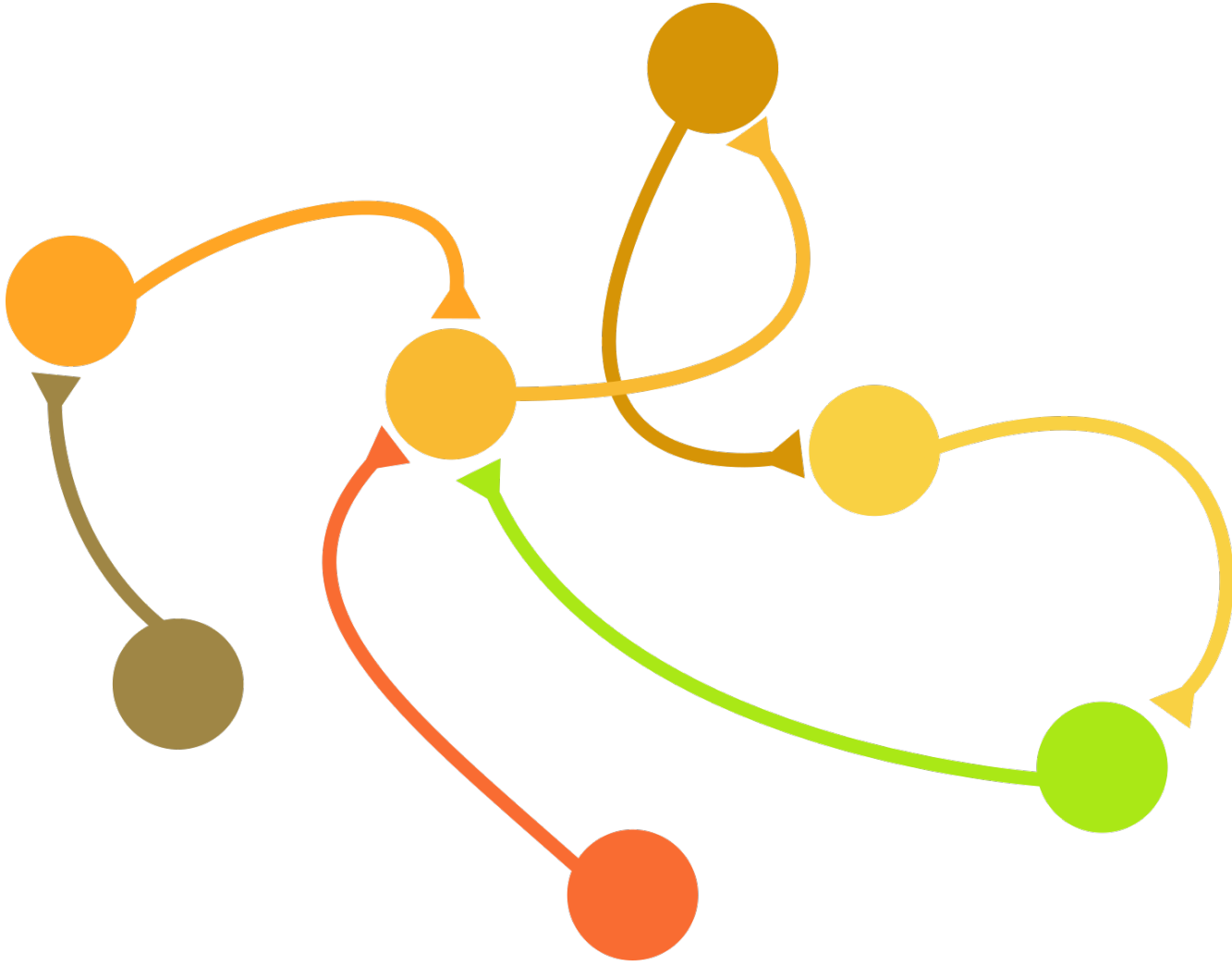


Biological variability: parameters change over time

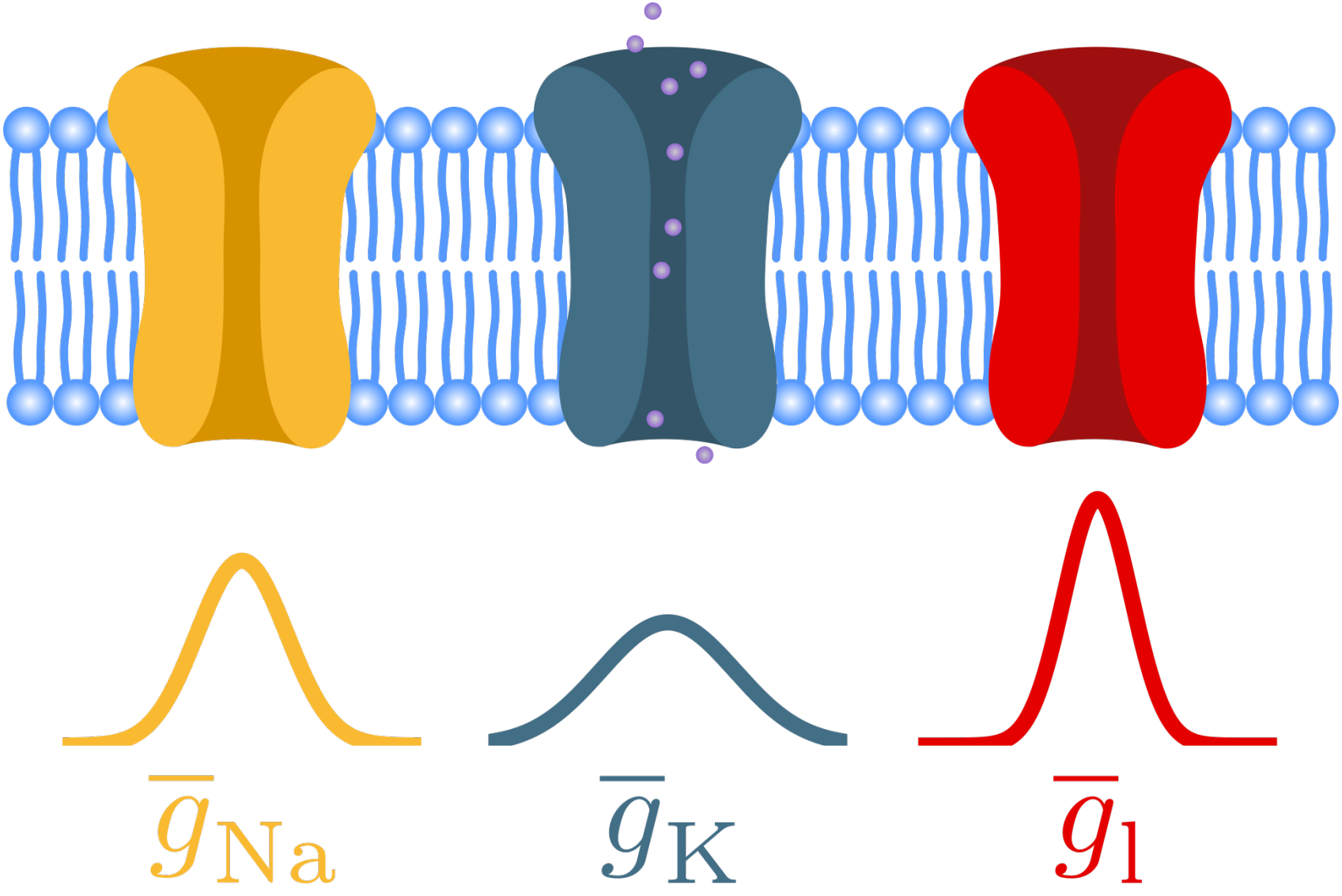
Biological variability: parameters vary within a neuron



Biological variability: parameters vary between several neurons of the same type

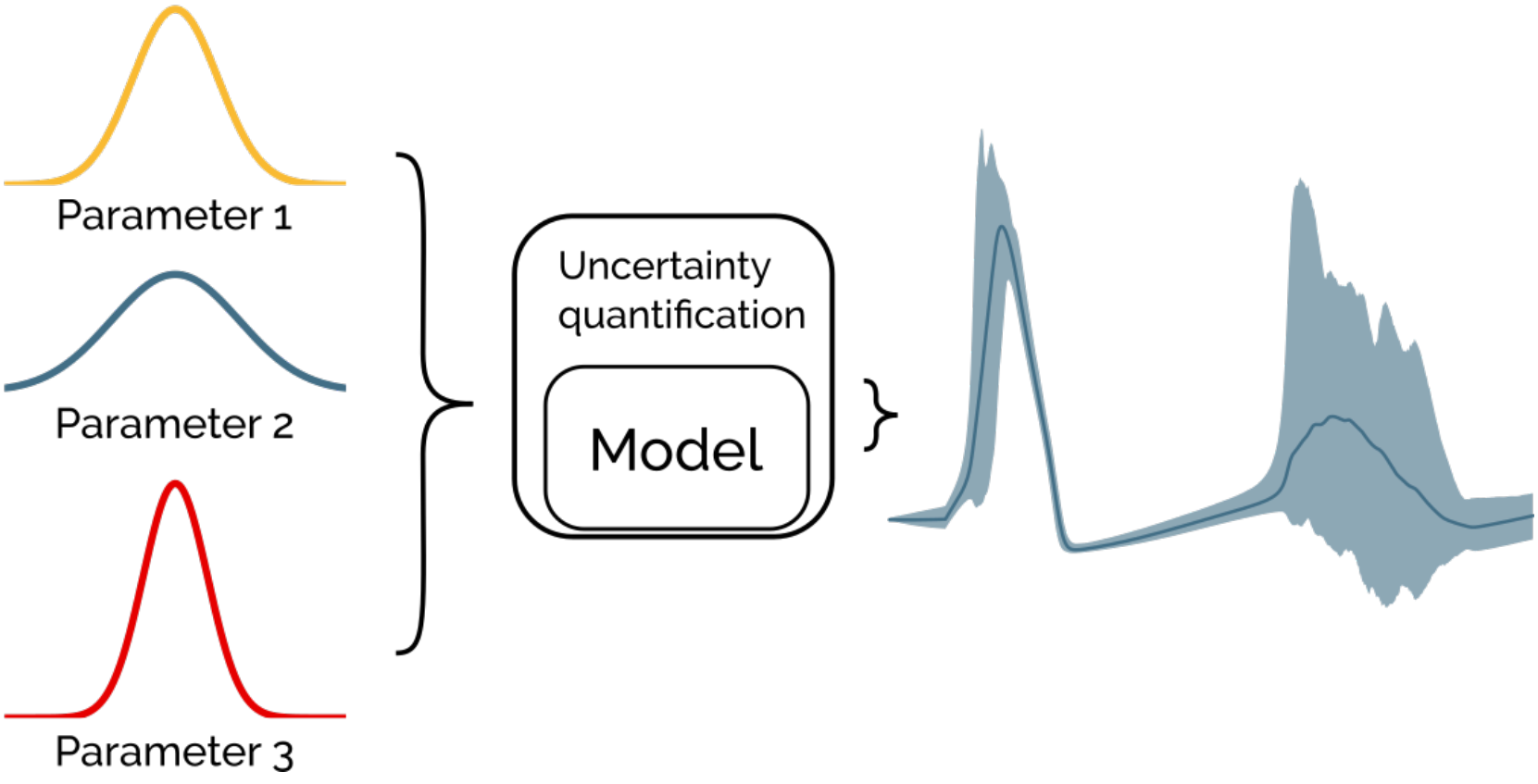


Parameters are best described by distributions

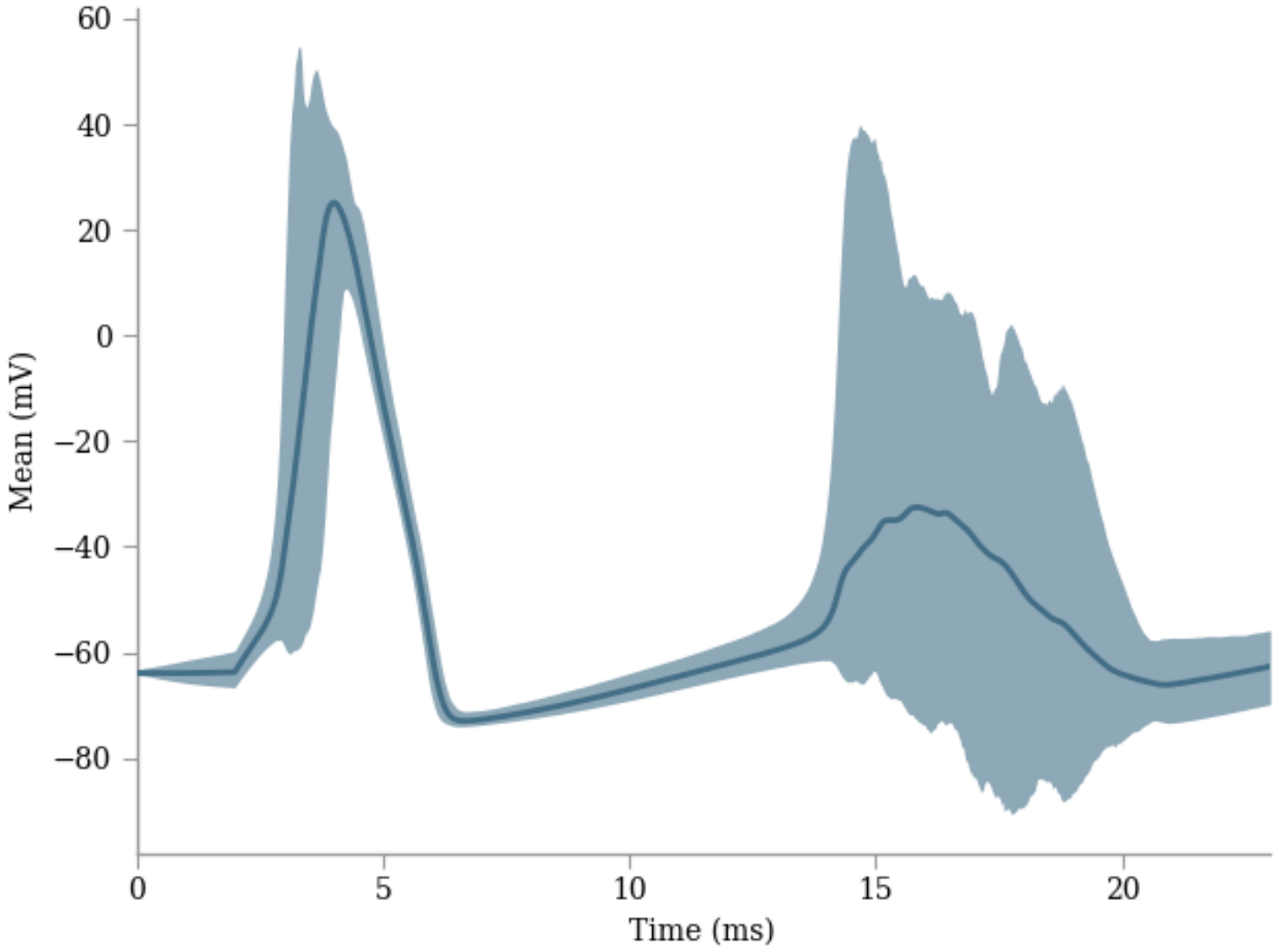


Uncertainty quantification enables us to take the effects of uncertain parameters into account

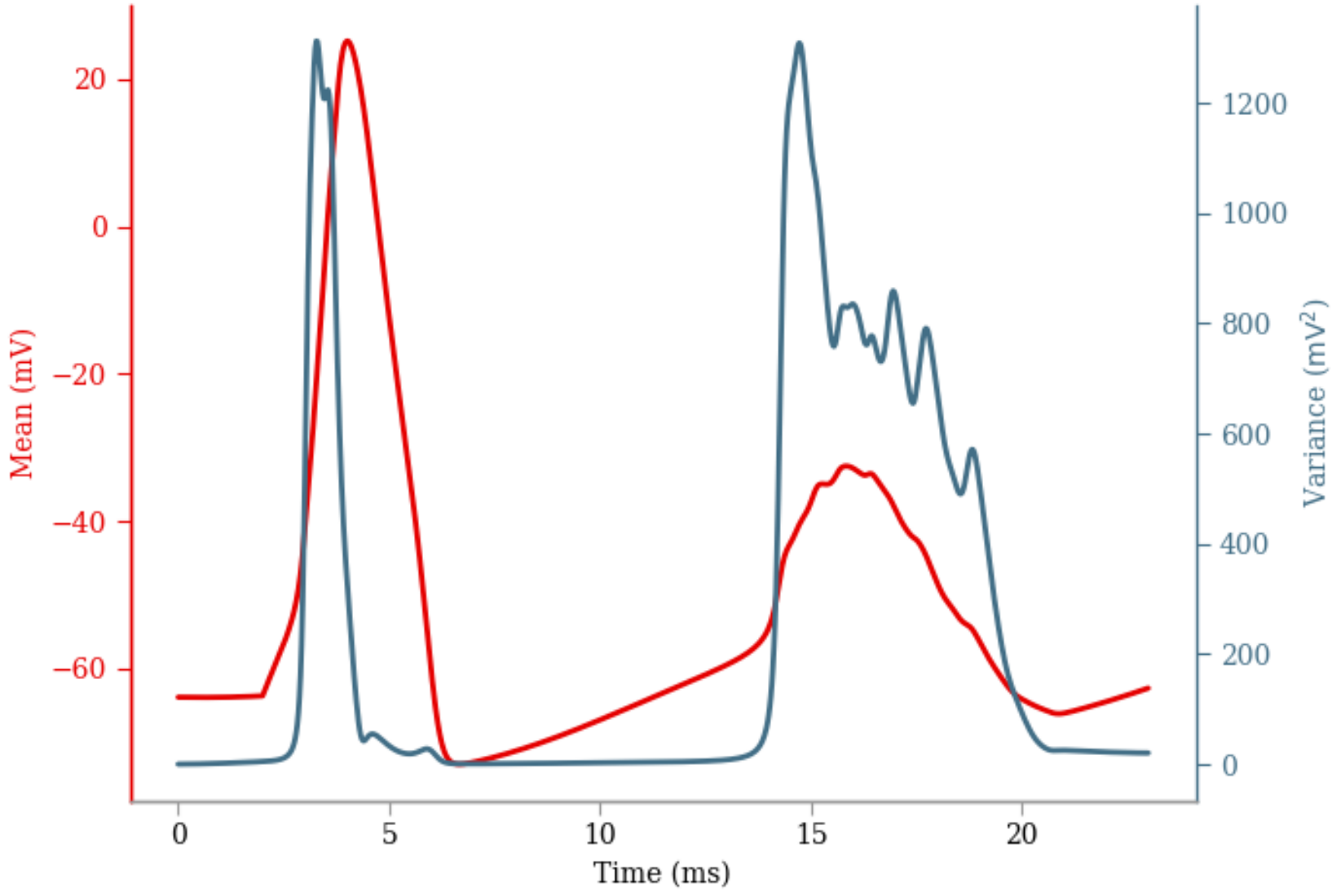
Uncertainty quantification is the process of quantifying the effects of uncertain parameters



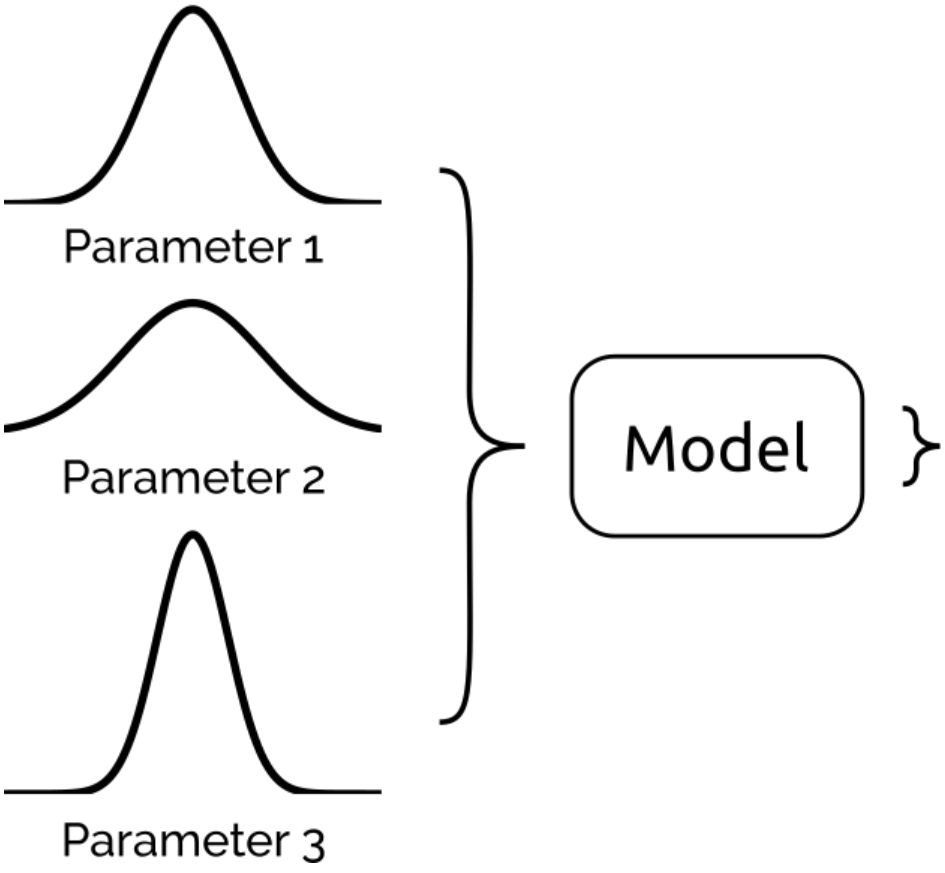
90% prediction interval of the Hodgkin-Huxley model



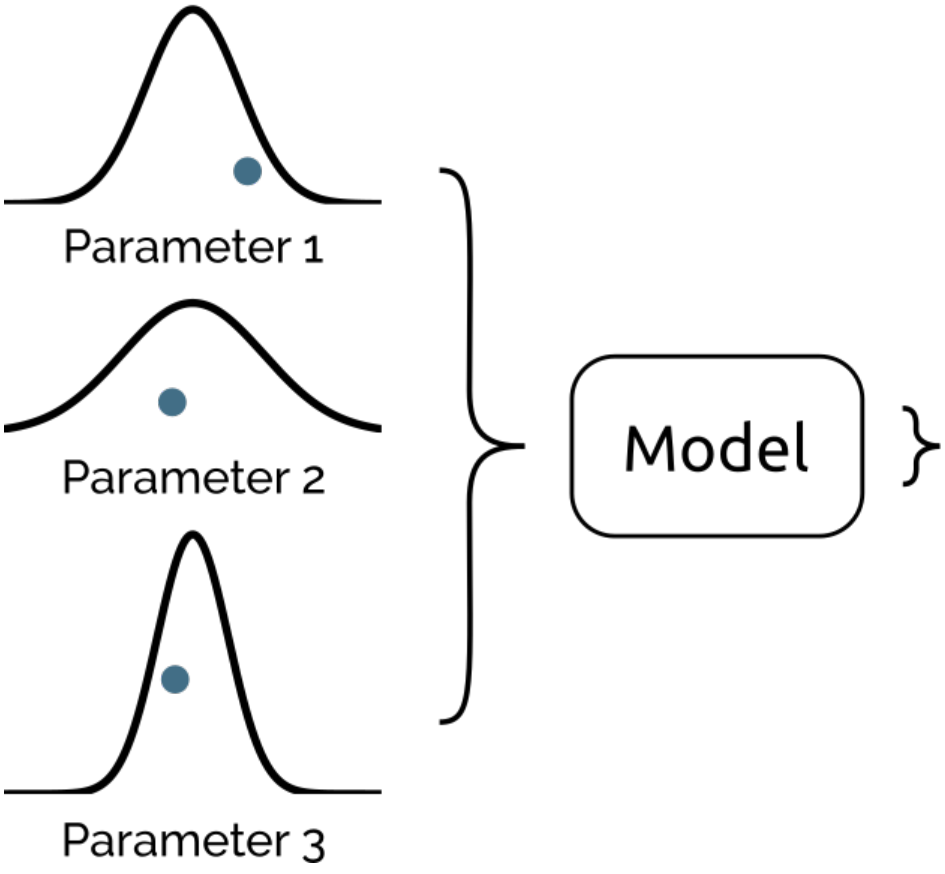
Mean and variance of the Hodgkin-Huxley model



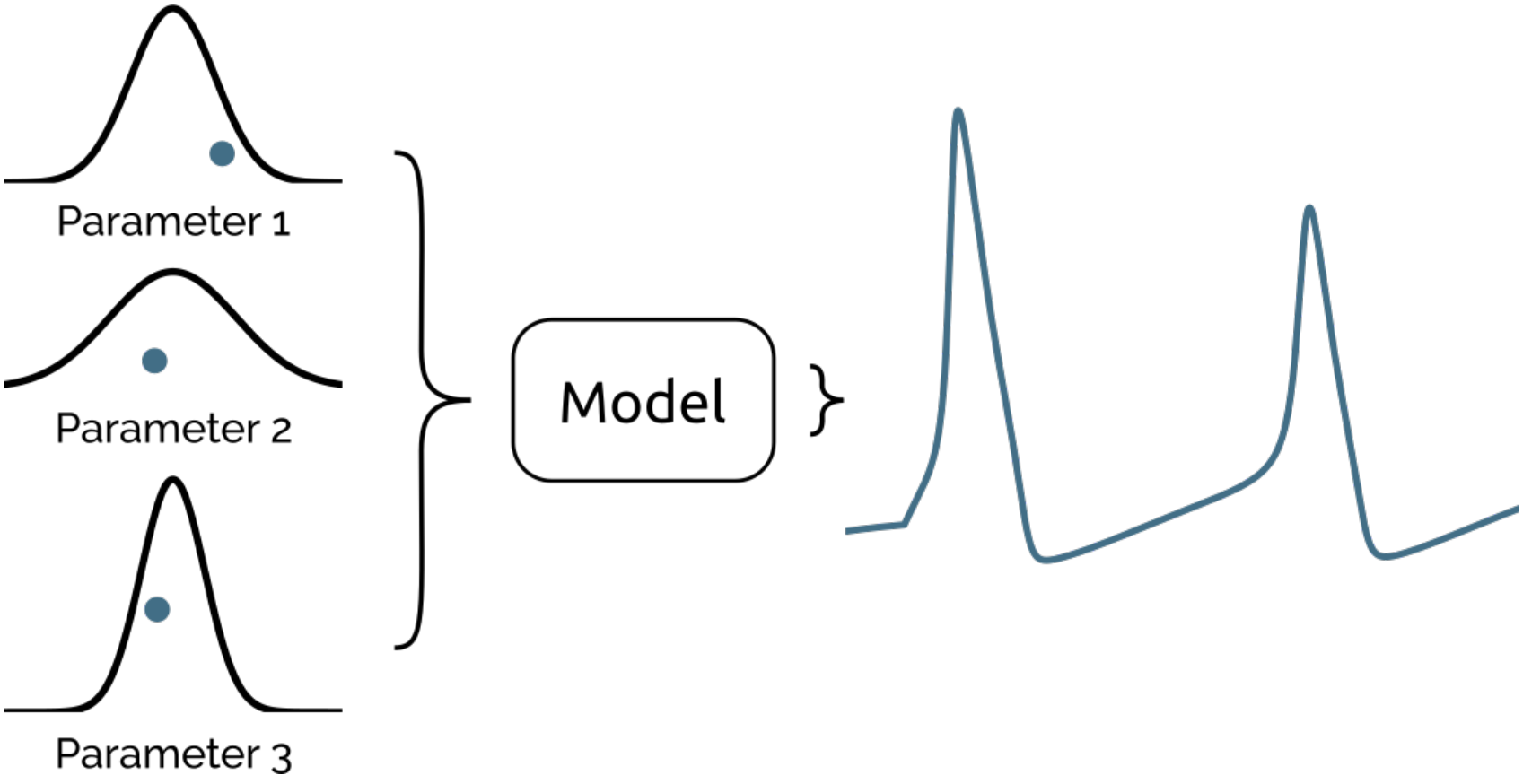
The Monte-Carlo method uses random numbers to perform the uncertainty quantification



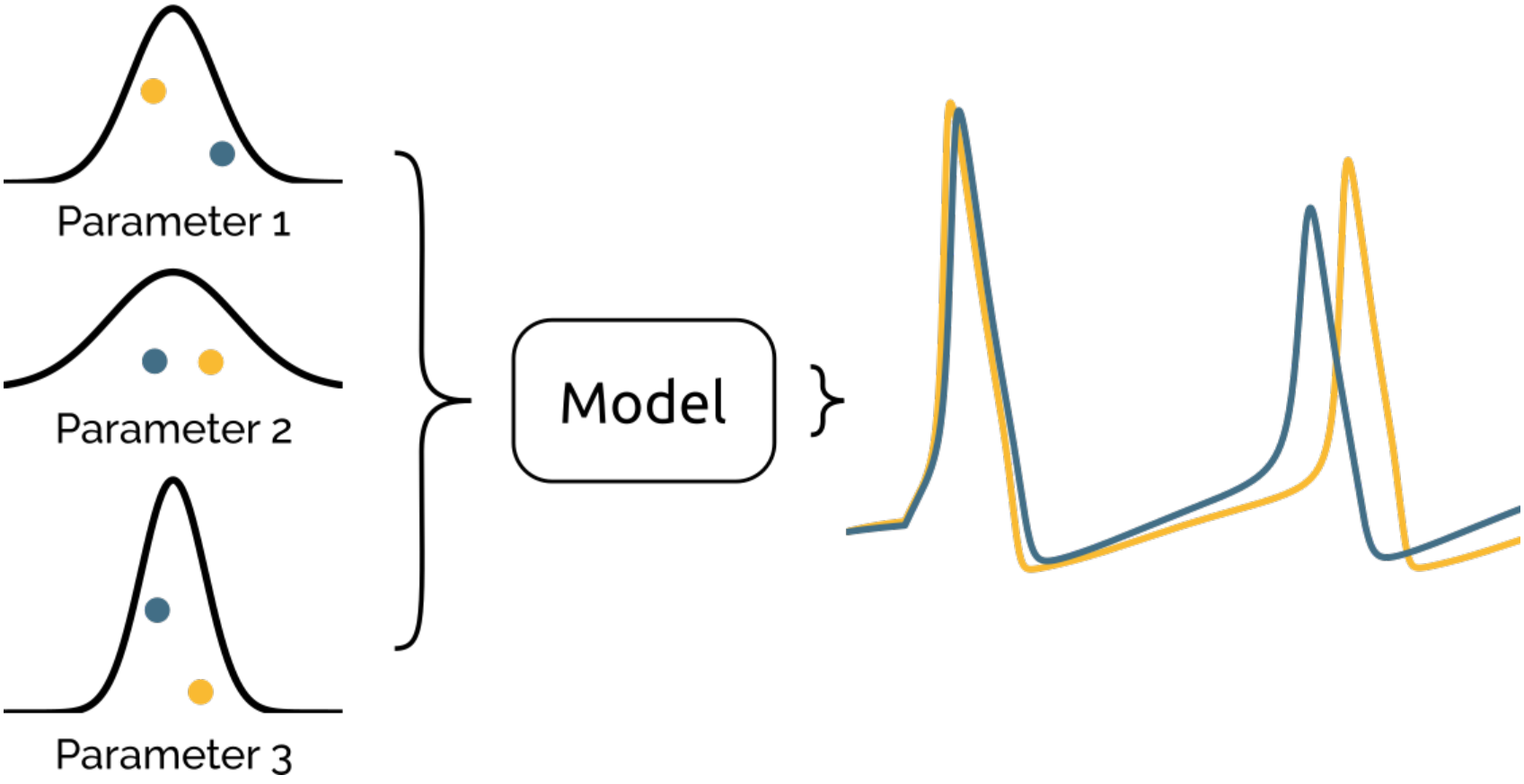
The Monte-Carlo method uses random numbers to perform the uncertainty quantification



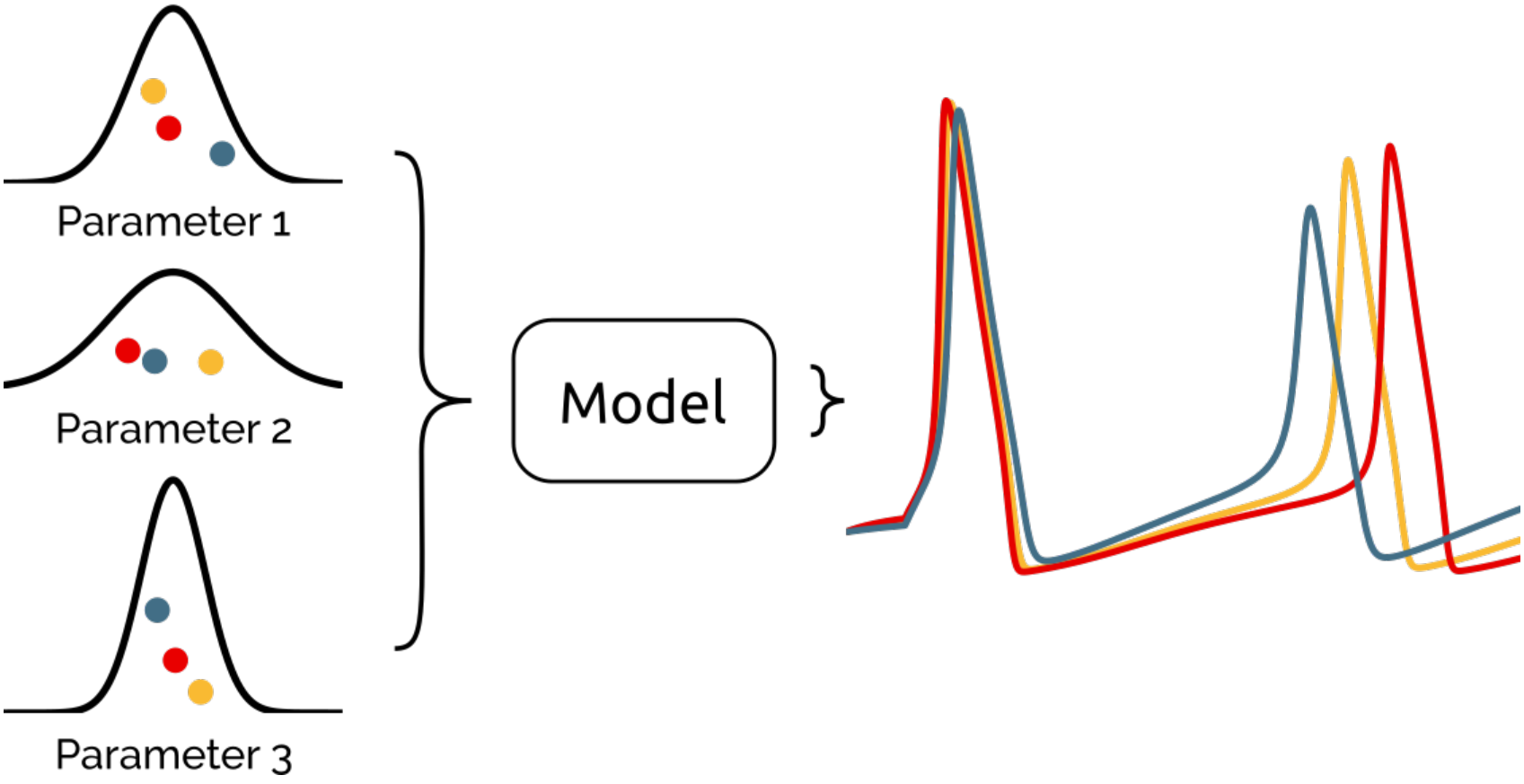
The Monte-Carlo method uses random numbers to perform the uncertainty quantification



The Monte-Carlo method uses random numbers to perform the uncertainty quantification



The Monte-Carlo method uses random numbers to perform the uncertainty quantification



A Python toolbox for uncertainty quantification

Found at:

```
https://github.com/simetenn/uncertainpy
```

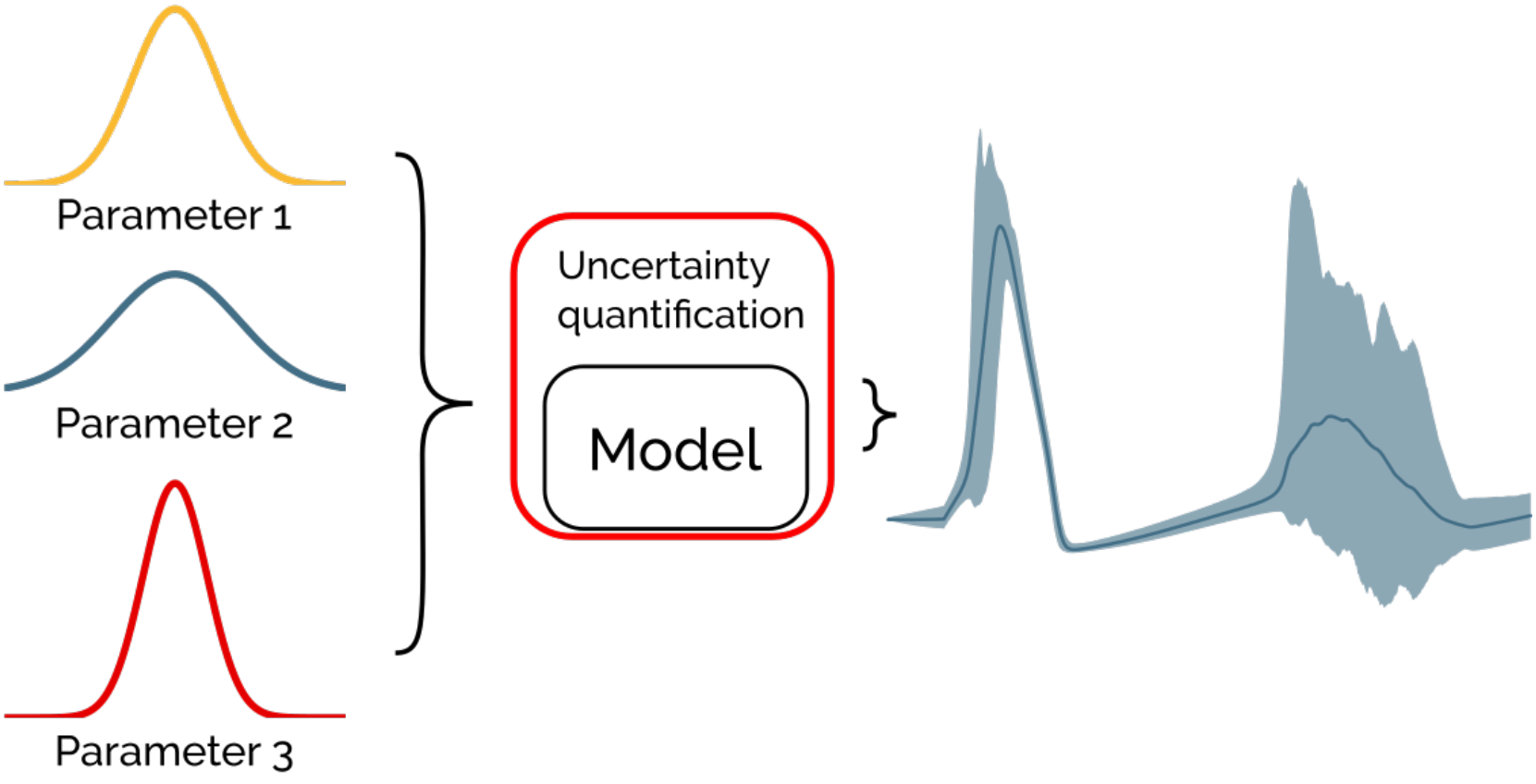
Installation:

```
pip install uncertainpy
```

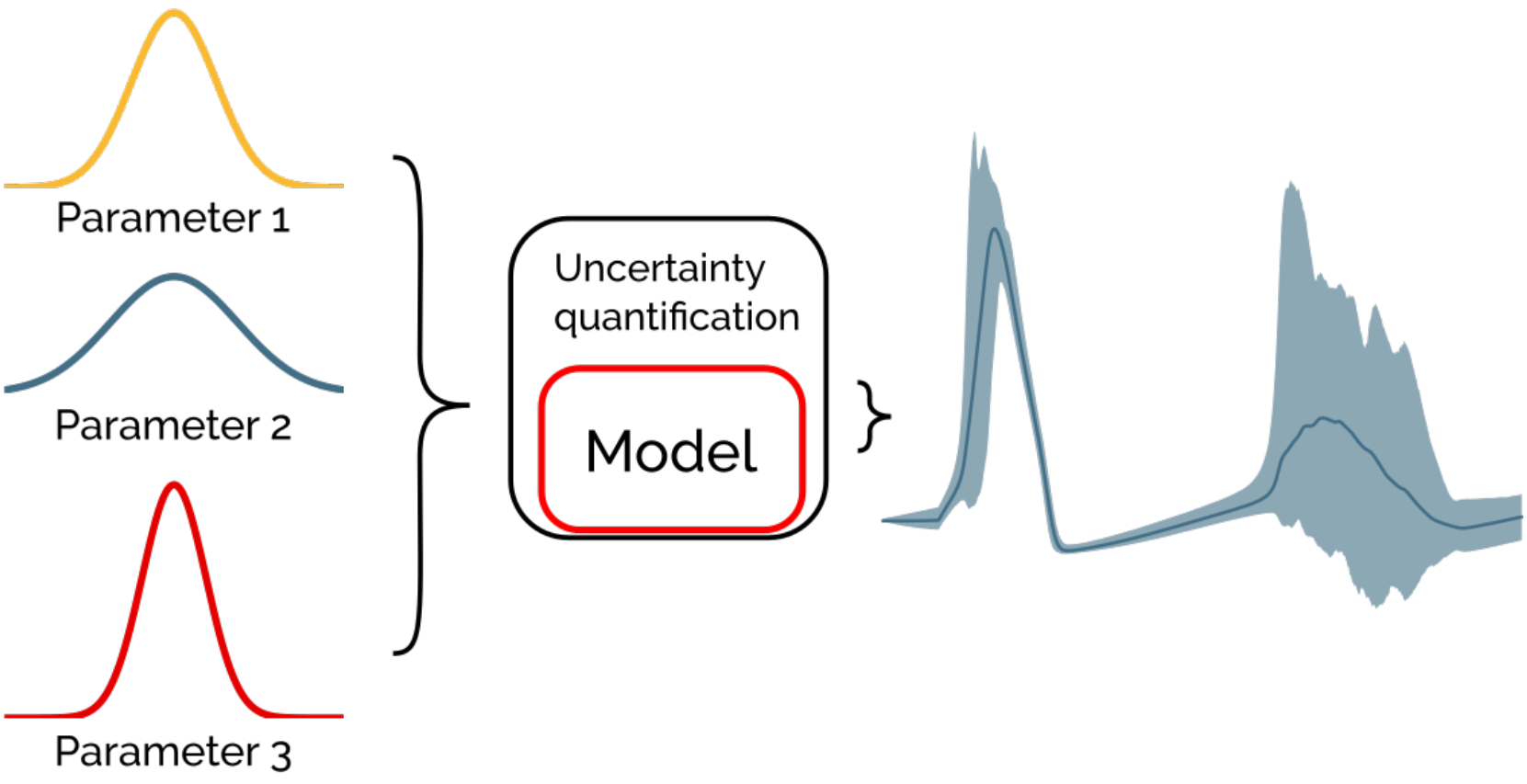
Preprint found at:

```
www.biorxiv.org
```

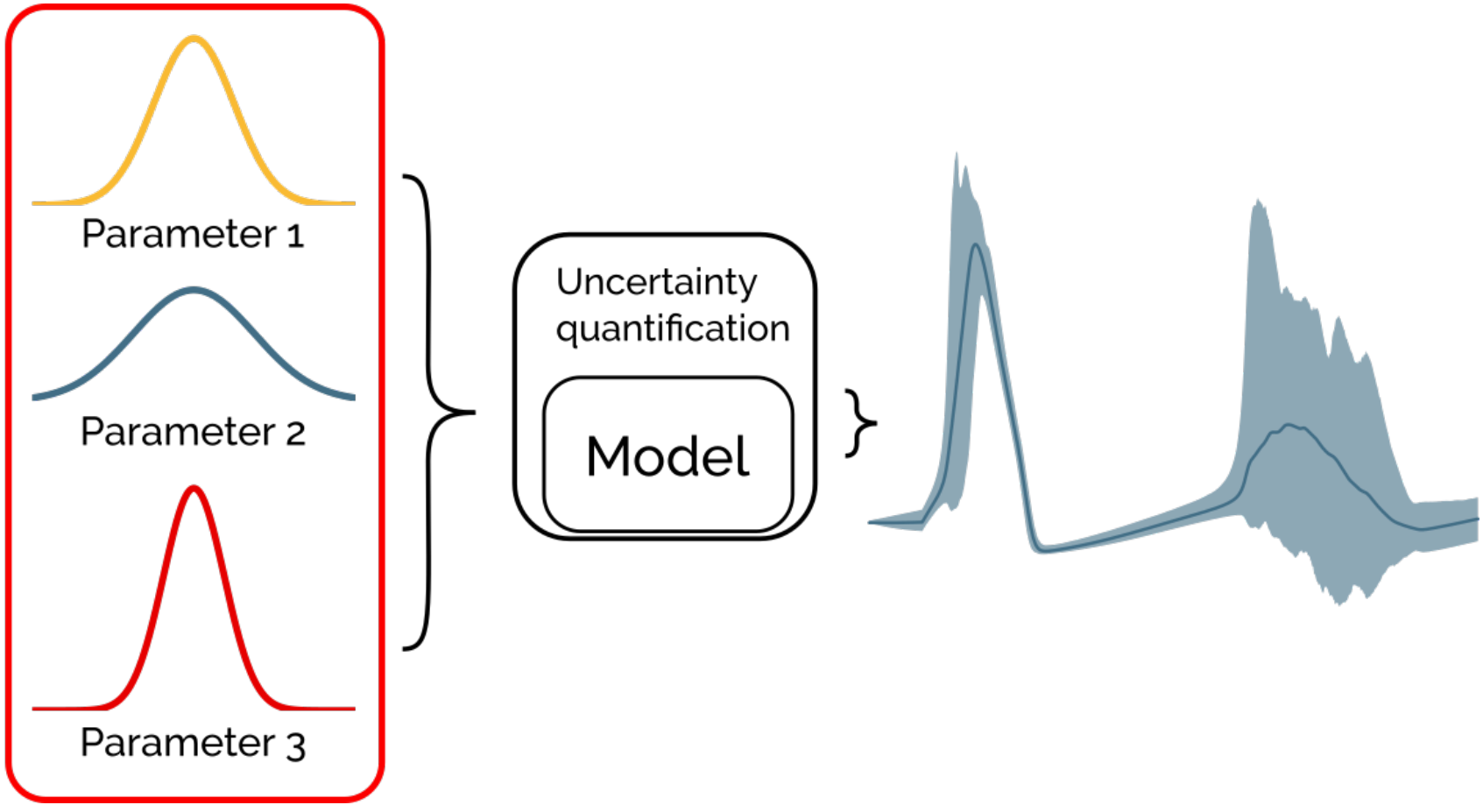
Uncertainpy performs all uncertainty calculations and requires the model and parameters



Uncertainpy performs all uncertainty calculations and requires the model and parameters



Uncertainpy performs all uncertainty calculations and requires the model and parameters



The problem is set up using the `UncertaintyQuantification` class

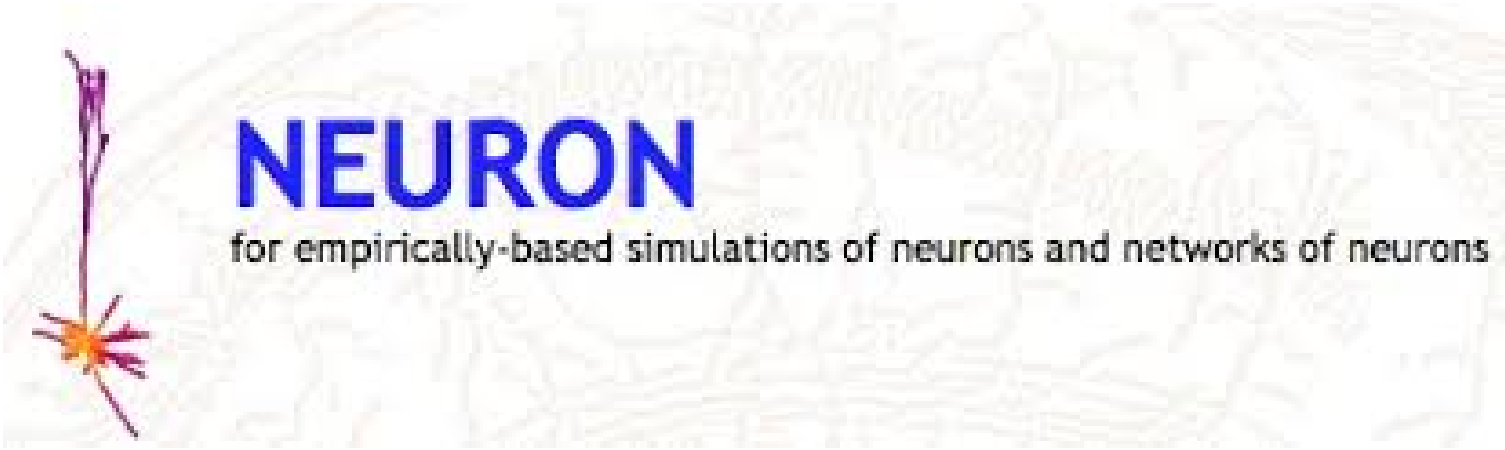
```
import uncertainpy as un

UQ = un.UncertaintyQuantification(
    model=...,
    parameters=...
)
```

Models are created as Python functions

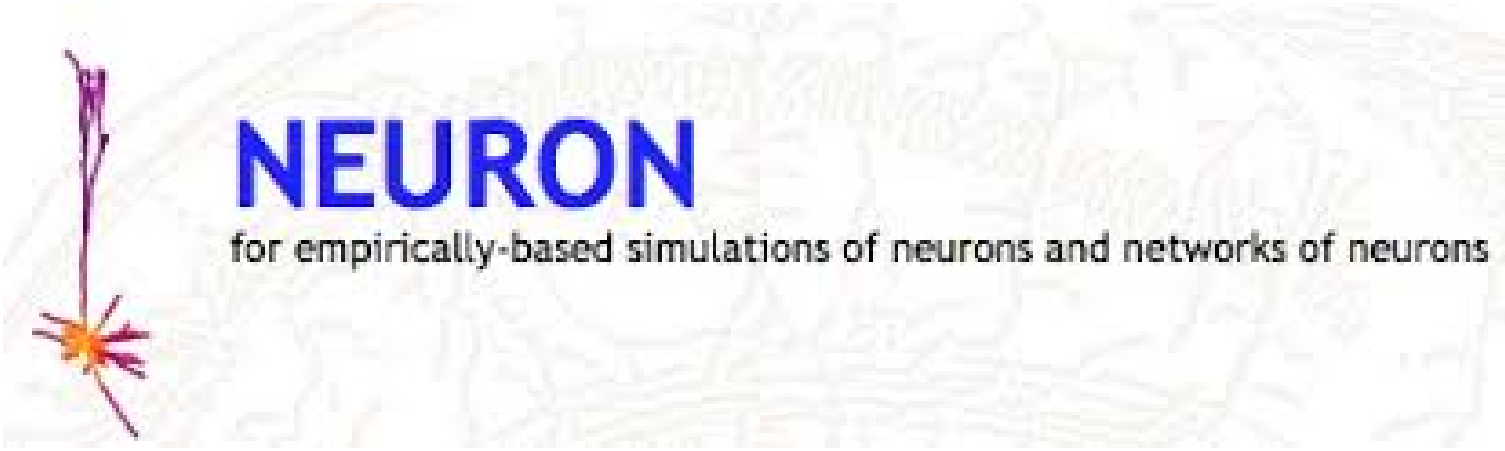
```
def hodgin_huxley(gbar_Na, gbar_K, gbar_l):
    # Calculate the membrane potential using
    # gbar_Na, gbar_K and gbar_l

    ...

    return time, membrane_potential
```

```
UQ = un.UncertaintyQuantification(
    model=hodgin_huxley,
    parameters=...
)
```

Uncertainpy has support for multi-compartmental models



Uncertainpy has support for multi-compartmental models and network models



Parameters are given as a Python dictionary

```
parameters = {"name": distribution}
```

Parameters are given as a Python dictionary

```
import chaospy as cp

uniform_distribution = cp.Uniform(2, 5)
normal_distribution = cp.Normal(12, 2)
```

Parameters are given as a Python dictionary

```
parameters = {                              # Original
  "gbar_Na": cp.Uniform(60, 180),           # 120
  "gbar_K": cp.Uniform(18, 54),             # 36
  "gbar_l": cp.Uniform(0.15, 0.45)          # 0.3
}
```

```
UQ = un.UncertaintyQuantification(
    model=hodgkin_huxley,
    parameters=parameters,
)
```

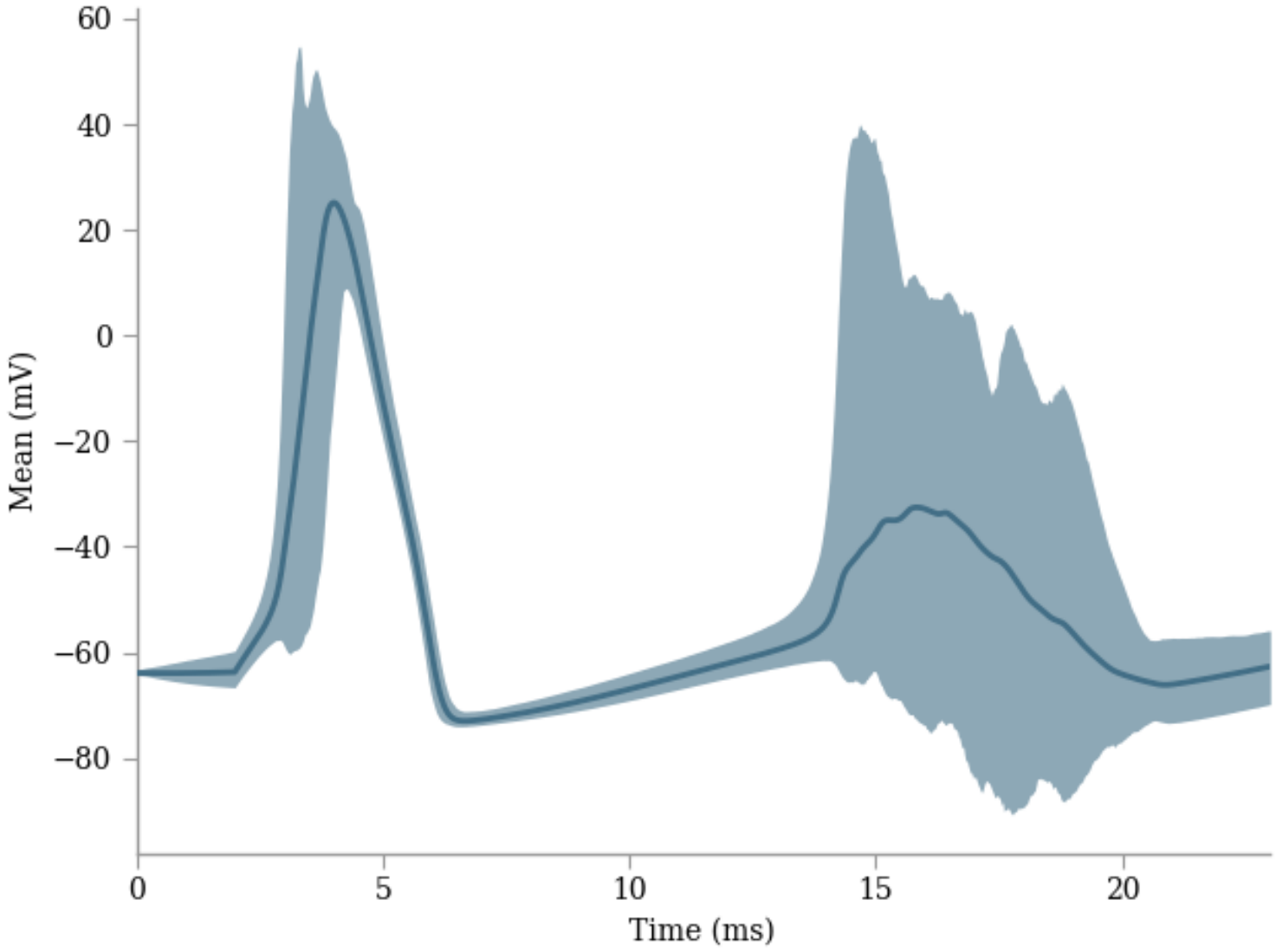
We perform the uncertainty calculations with `quantify()`

```
UQ = un.UncertaintyQuantification(
    model=hodgkin_huxley,
    parameters=parameters
)

results = UQ.quantify()
```

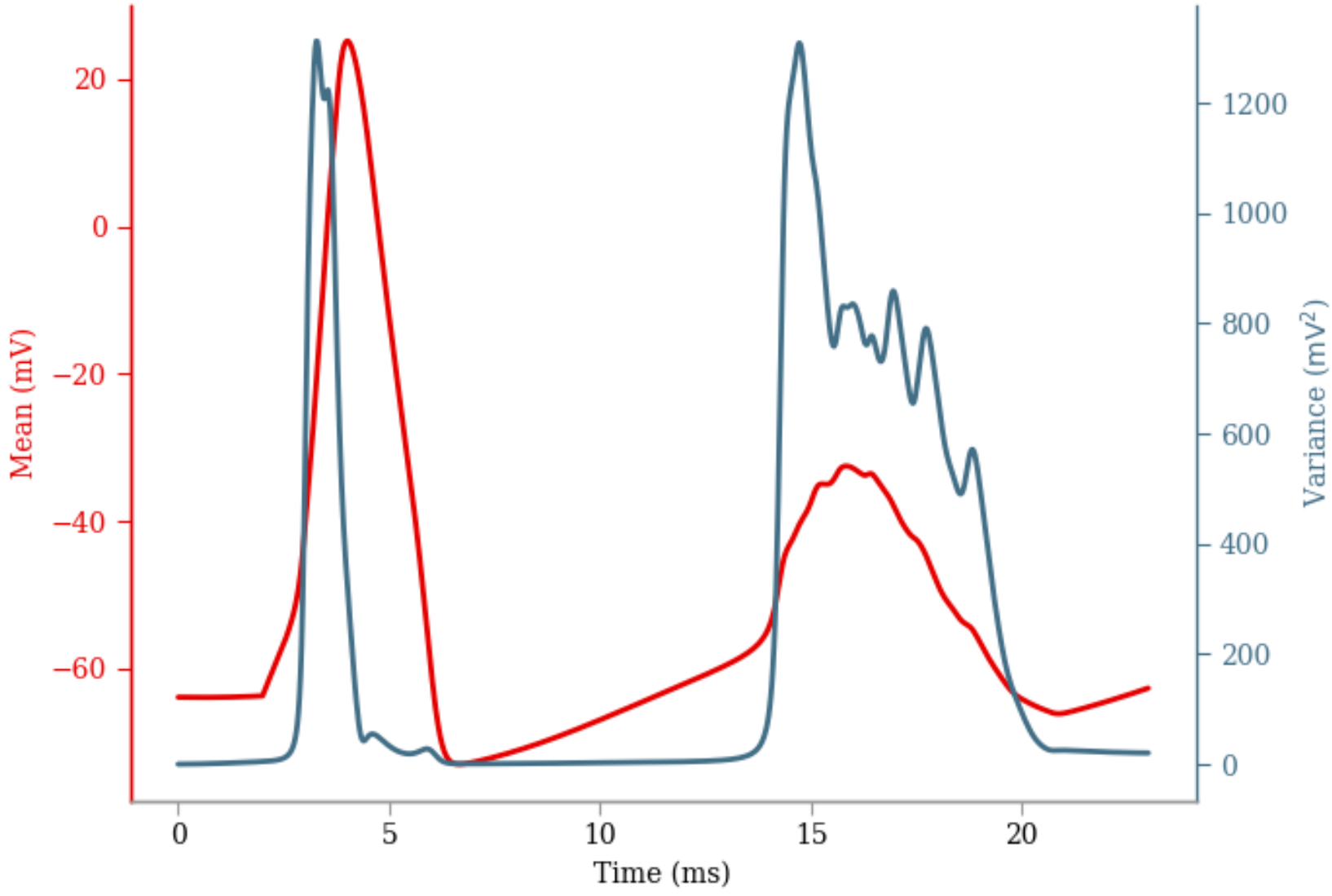
Overview of the Hodgkin-Huxley example

```
def hodgin_huxley(gbar_Na, gbar_K, gbar_l):
    # Calculate the membrane potential using
    # gbar_Na, gbar_K and gbar_l

    return time, membrane_potential

parameters = {                              # Original
  "gbar_Na": cp.Uniform(60, 180),           # 120
  "gbar_K": cp.Uniform(18, 54),             # 36
  "gbar_l": cp.Uniform(0.15, 0.45)}         # 0.3

UQ = un.UncertaintyQuantification(
    model=hodgkin_huxley,
    parameters=parameters)

results = UQ.quantify()
```

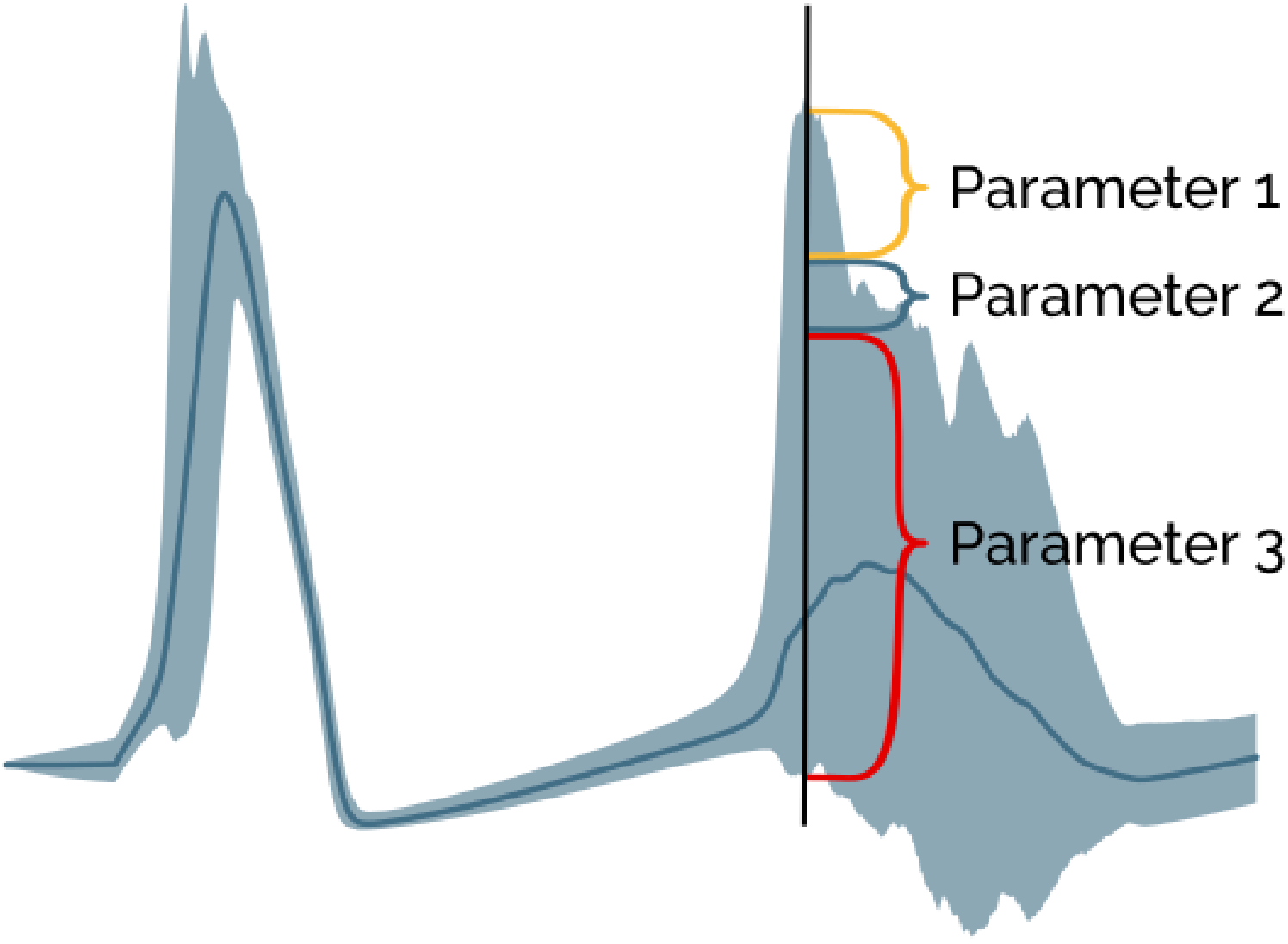
90% prediction interval of the Hodgkin-Huxley model



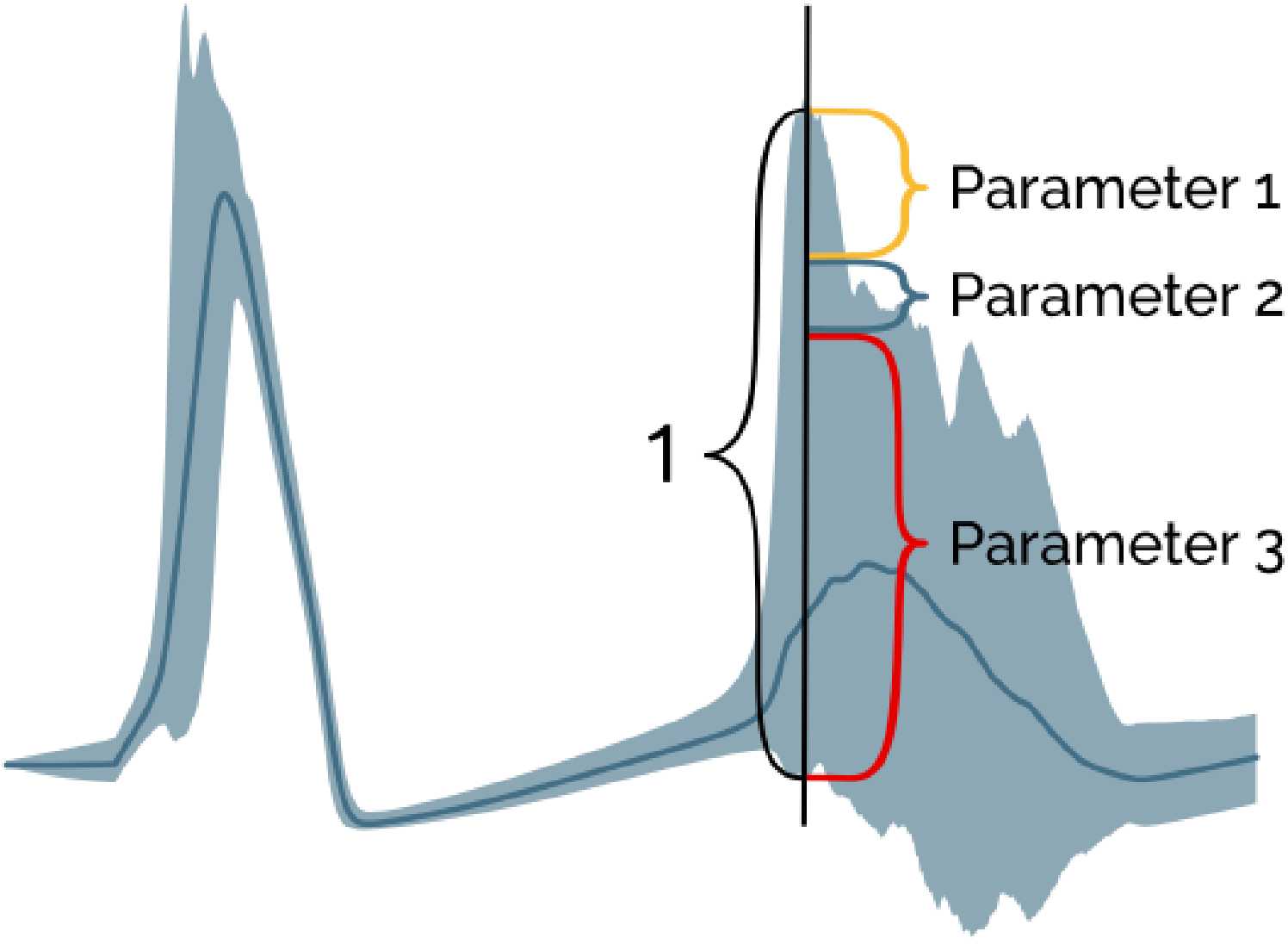
Mean and variance of the Hodgkin-Huxley model



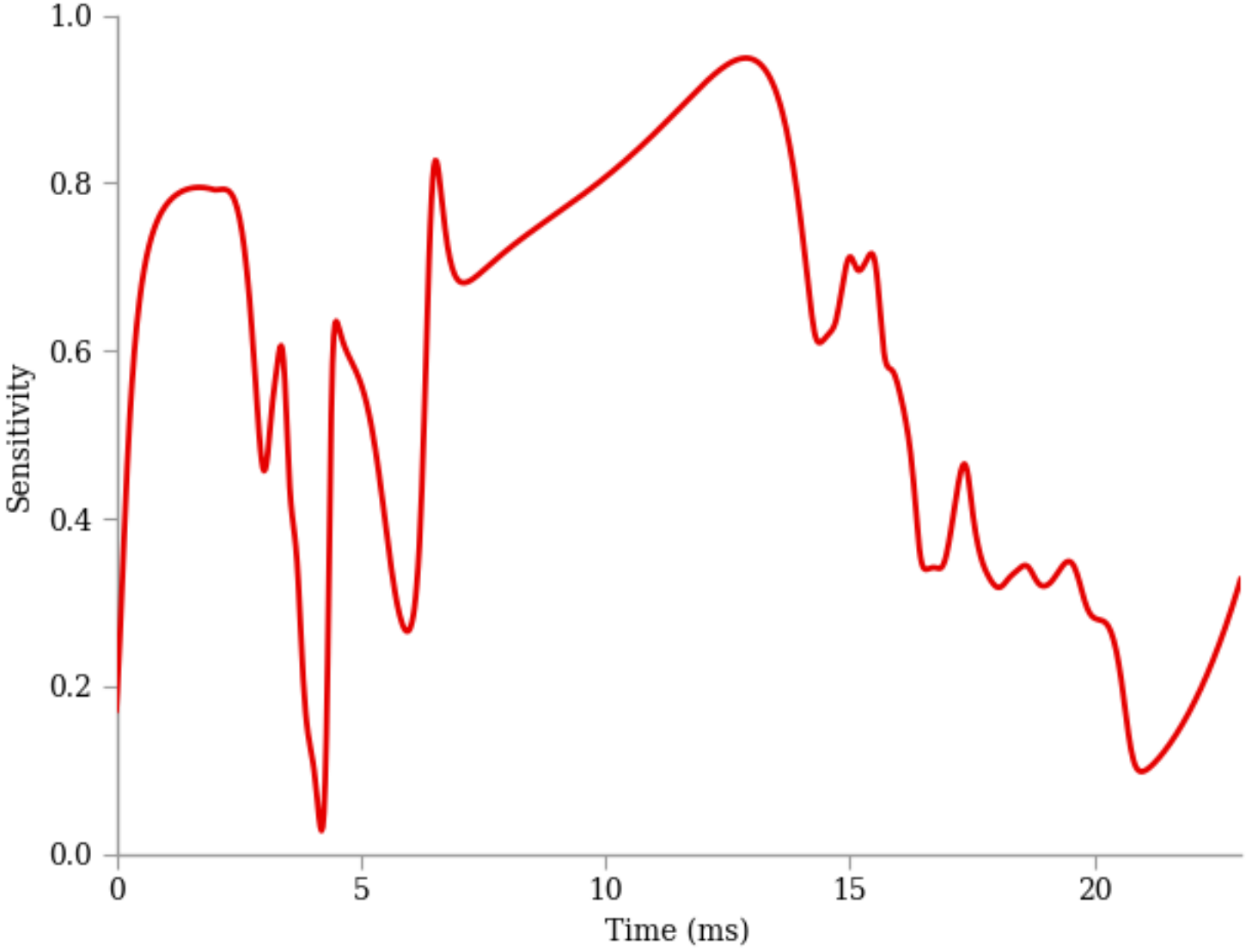
Sensitivity analysis quantifies how much of the uncertainty each parameter is responsible for



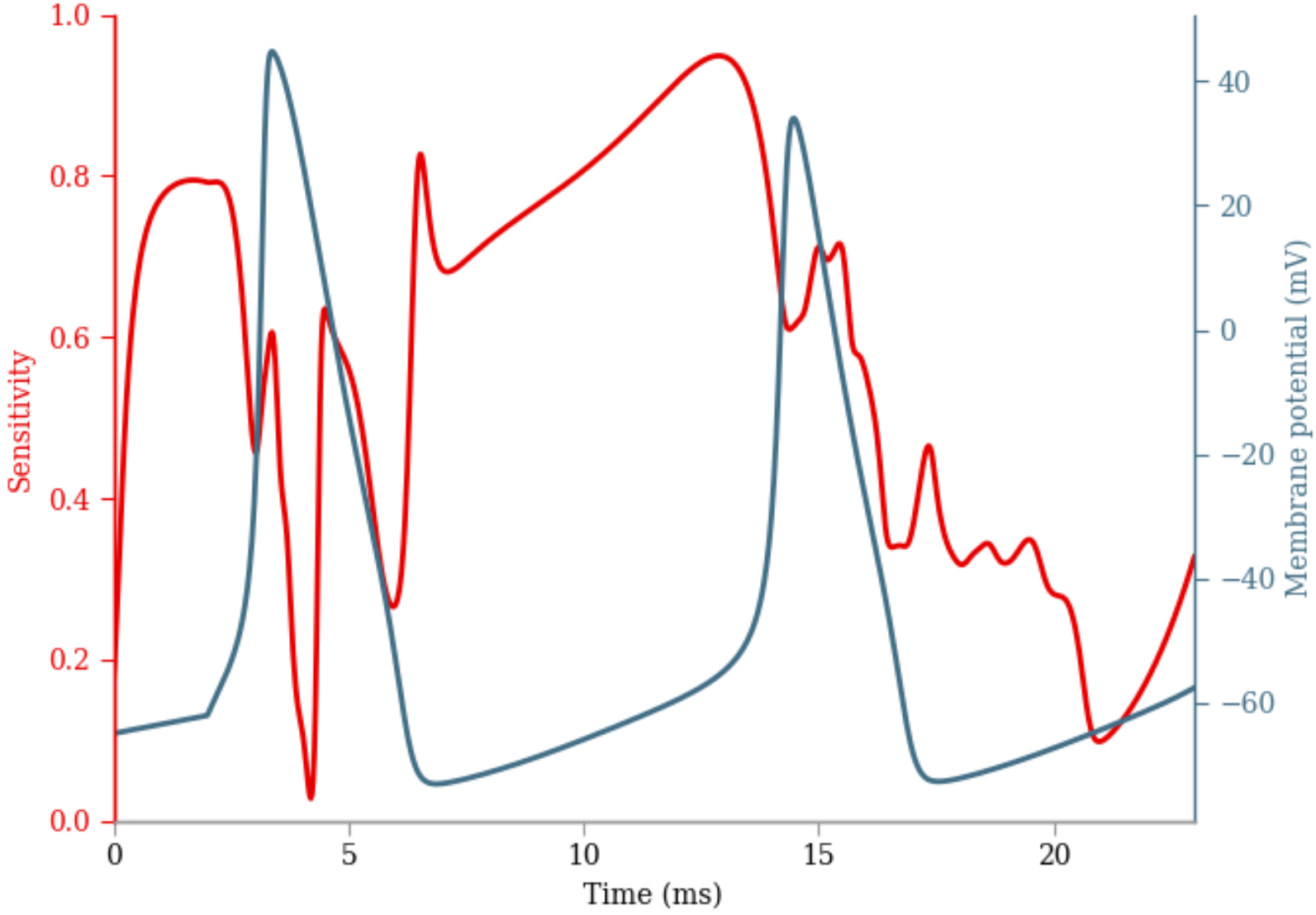
Sensitivity analysis quantifies how much of the uncertainty each parameter is responsible for

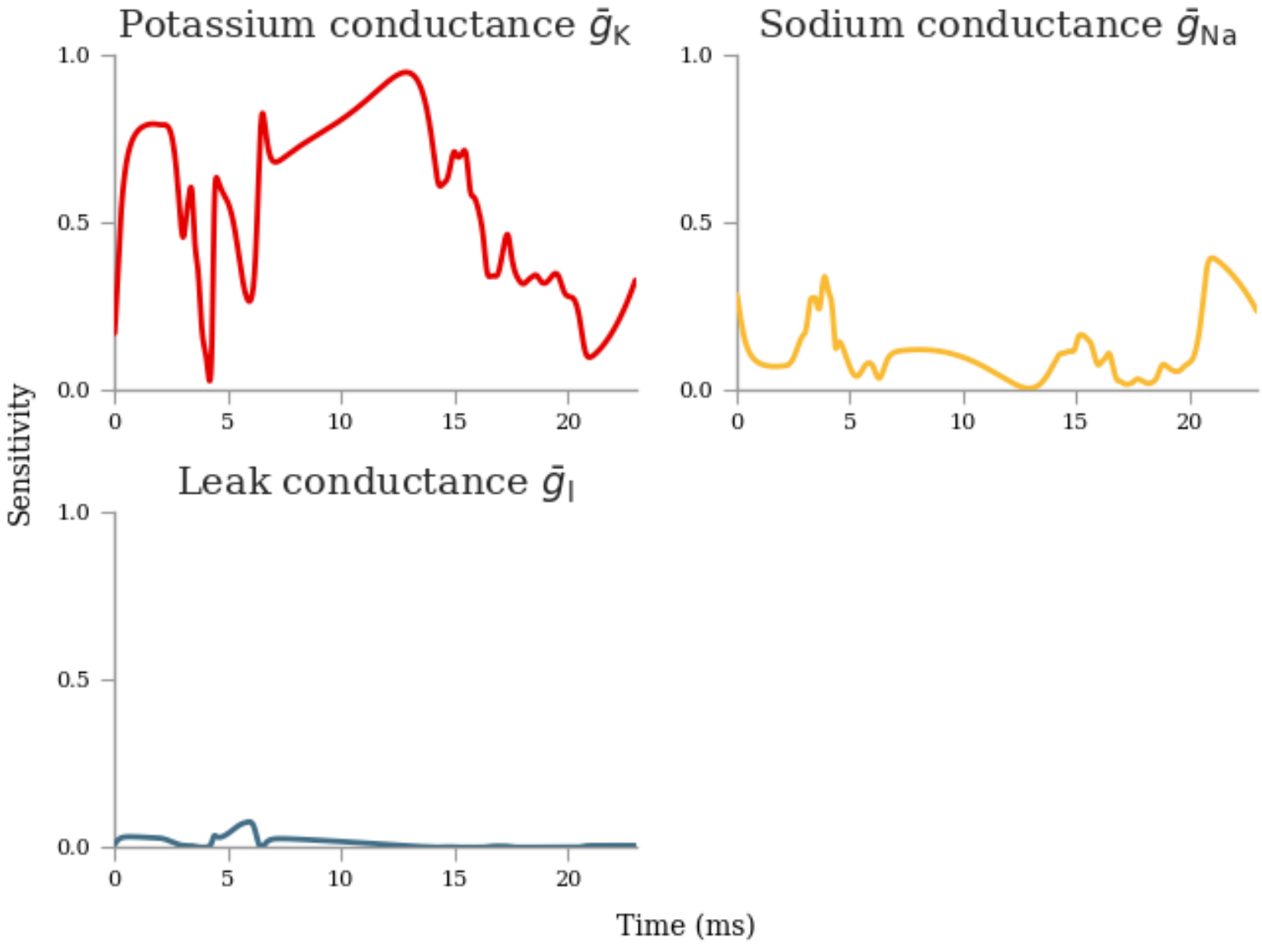


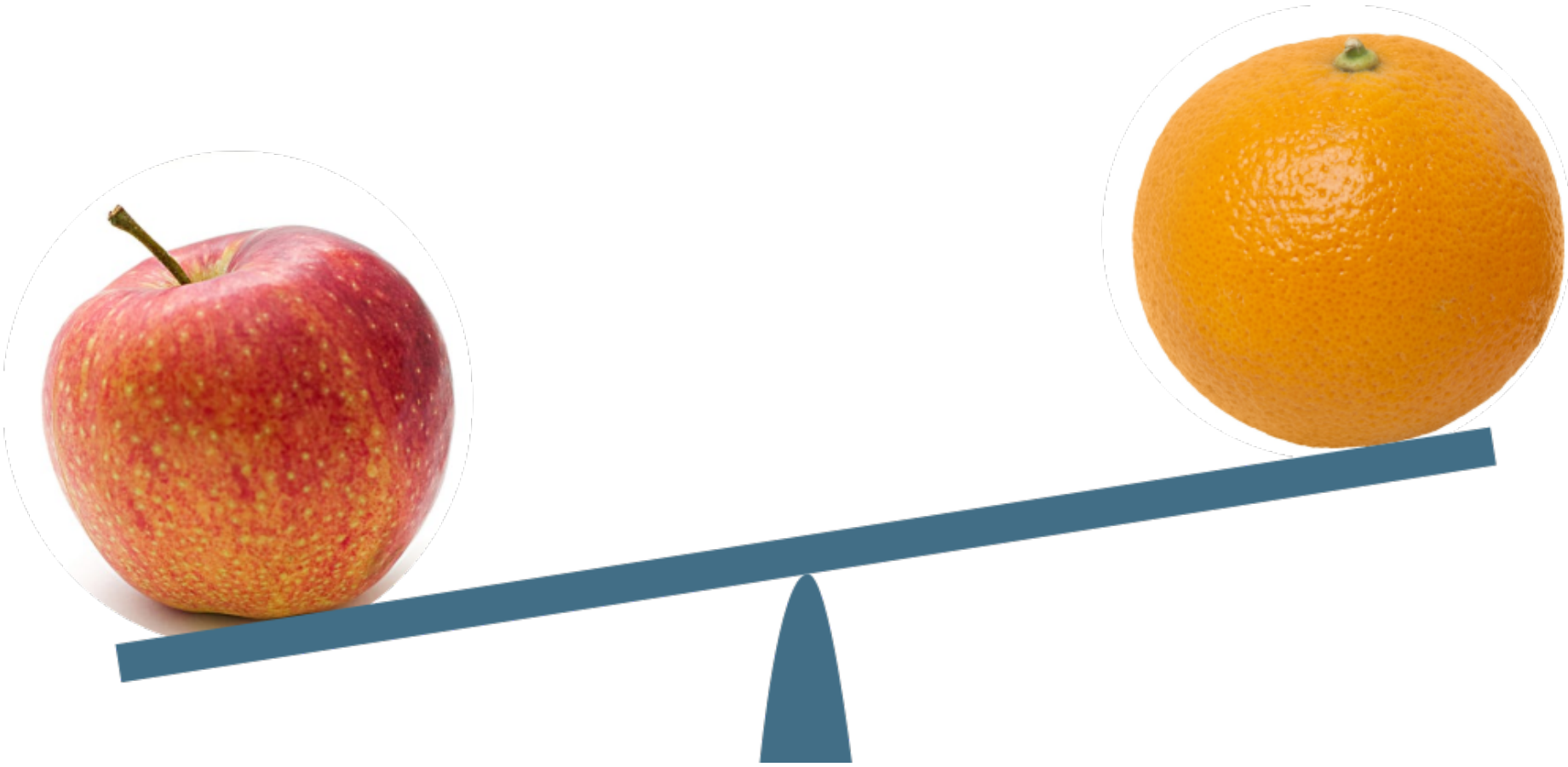
Sensitivity of the potassium conductance $\overline{g}_K$ in the Hodgkin-Huxley model



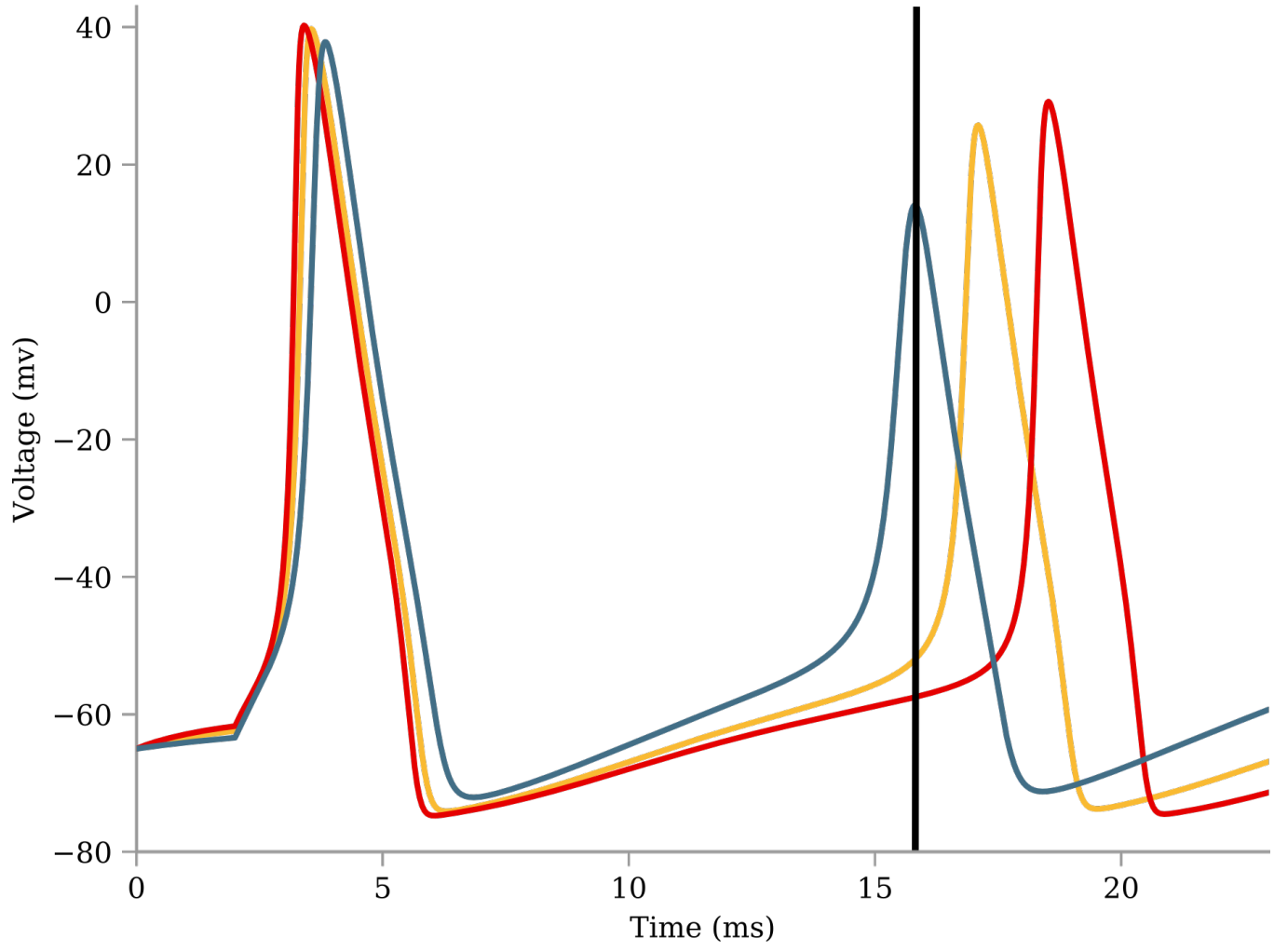
Sensitivity of the potassium conductance $\overline{g}_K$ in the Hodgkin-Huxley model



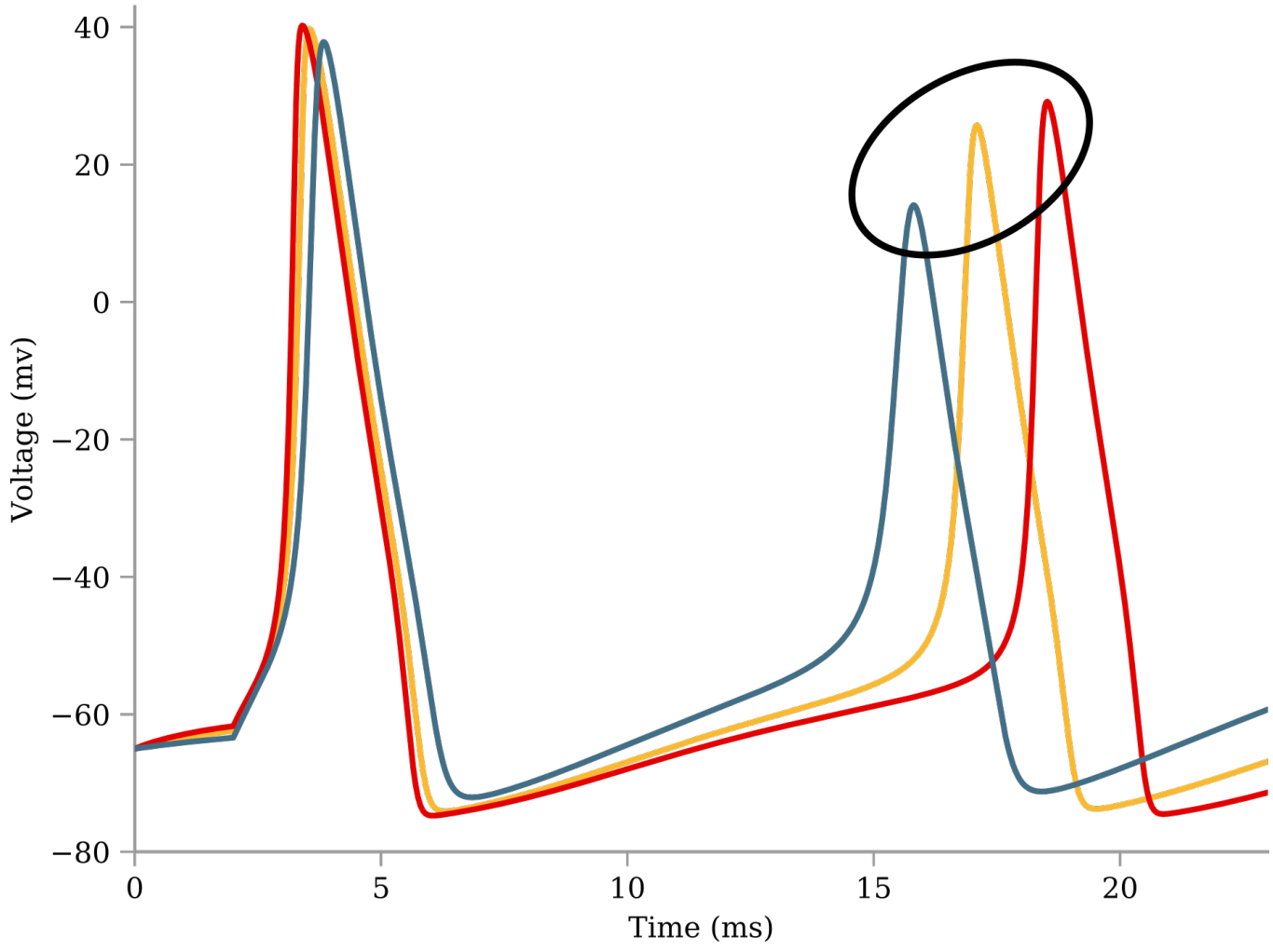

Potassium conductance $\bar{g}_K$
Sodium conductance $\bar{g}_{Na}$
Leak conductance $\bar{g}_l$
Sensitivity
1.0
0.5
0.0
0
5
10
15
20
Time (ms)



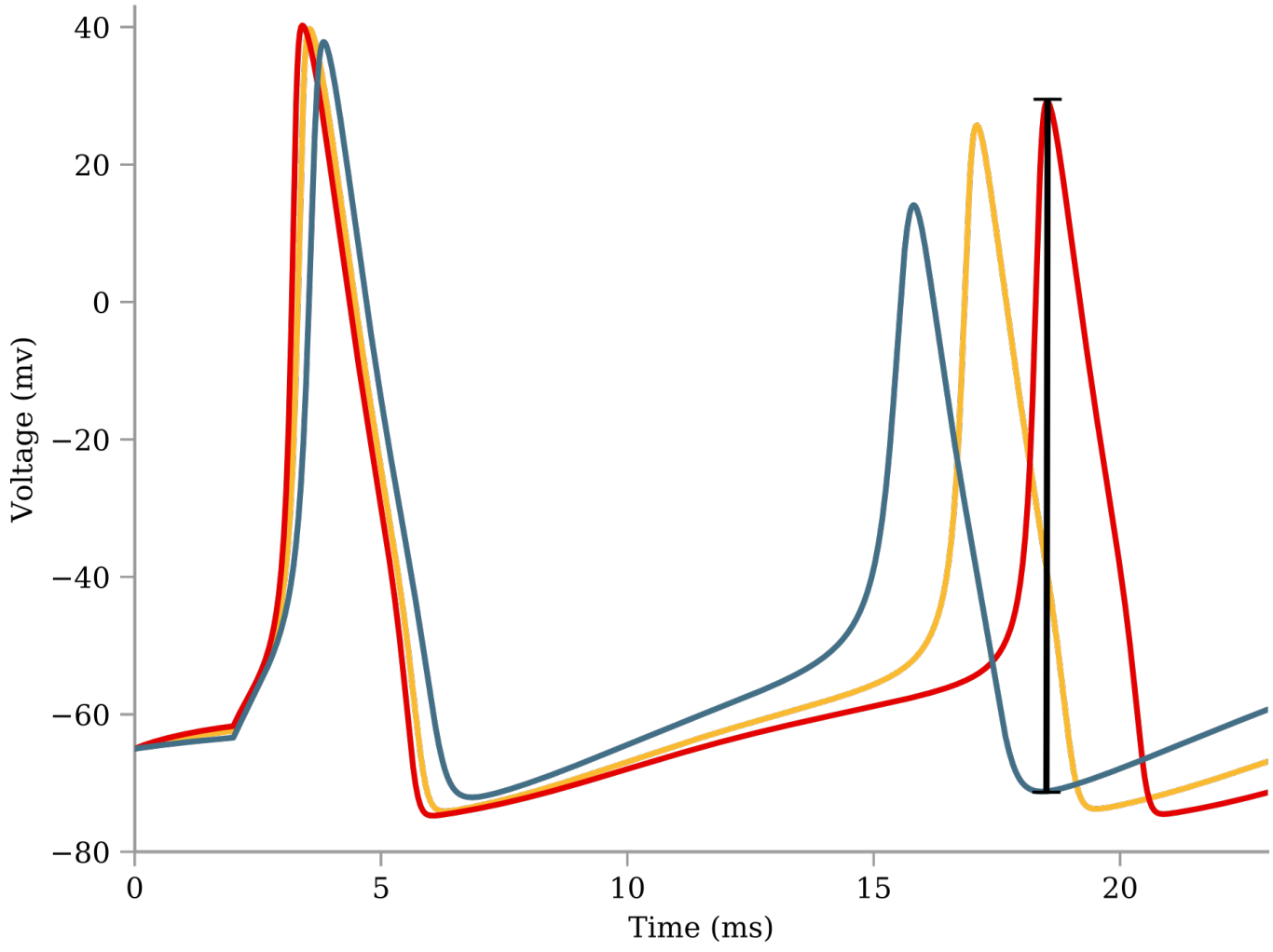
Pointwise comparison of model results give large differences due to small time shifts in spikes



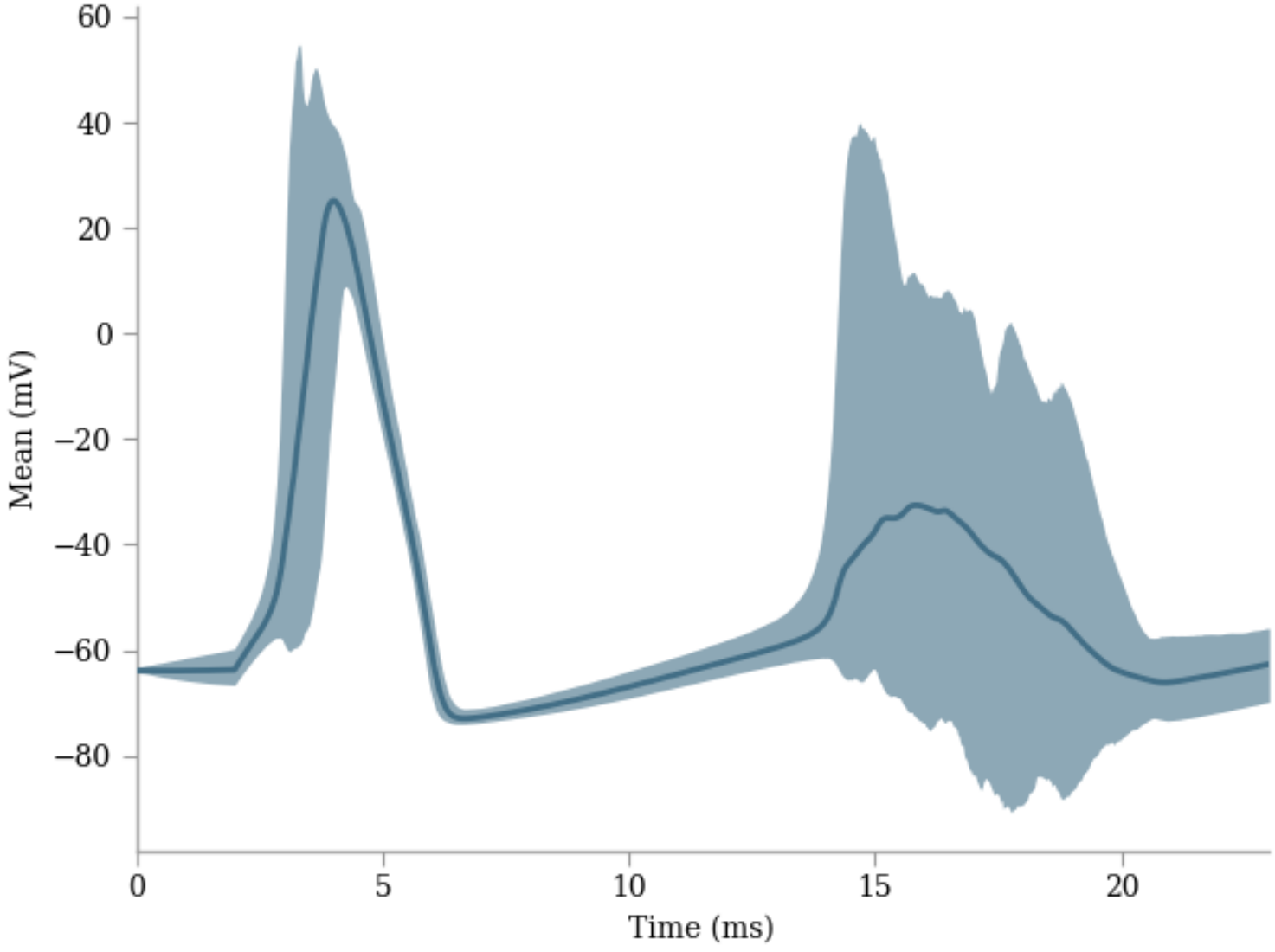
Pointwise comparison of model results give large differences due to small time shifts in spikes



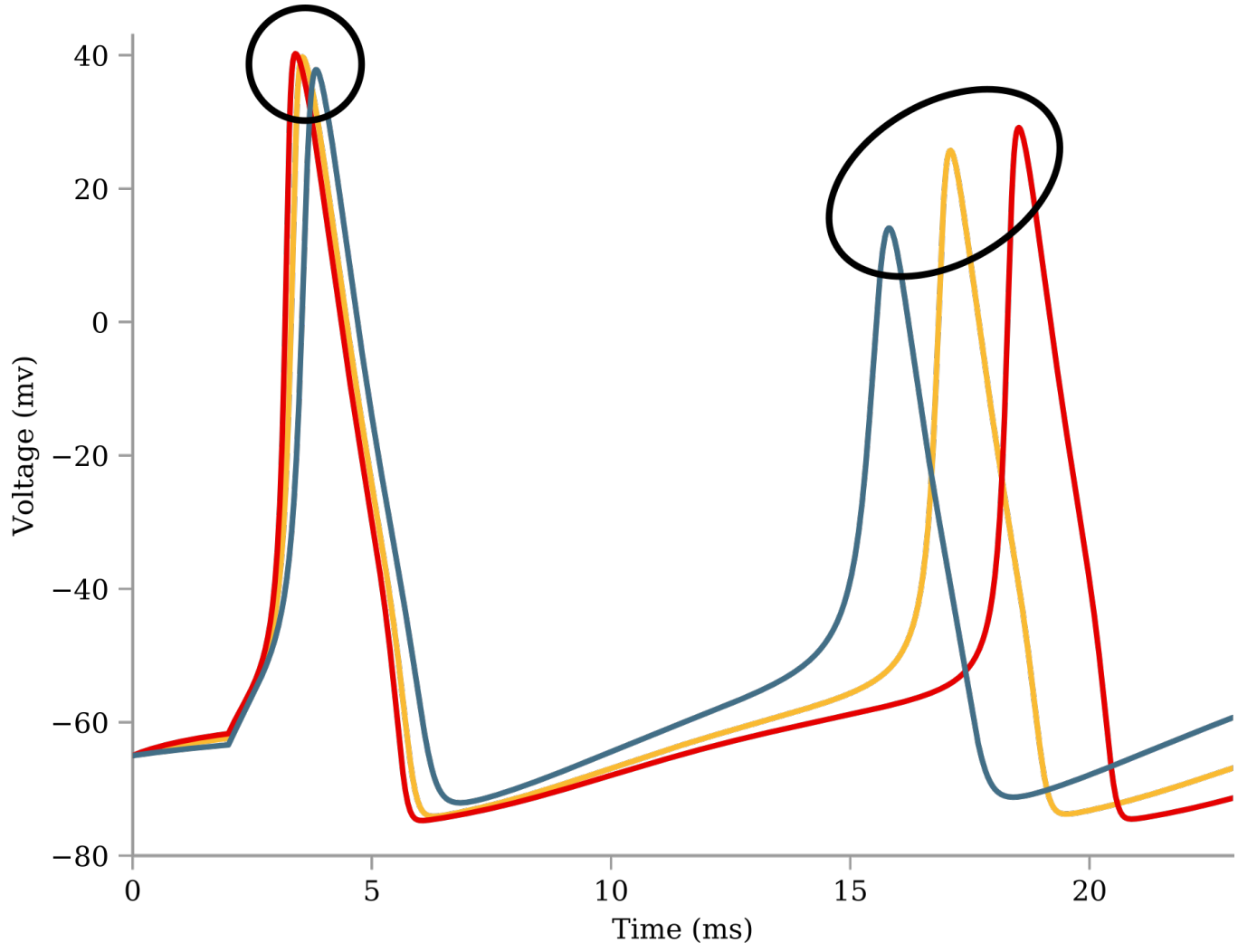
Pointwise comparison of model results give large differences due to small time shifts in spikes



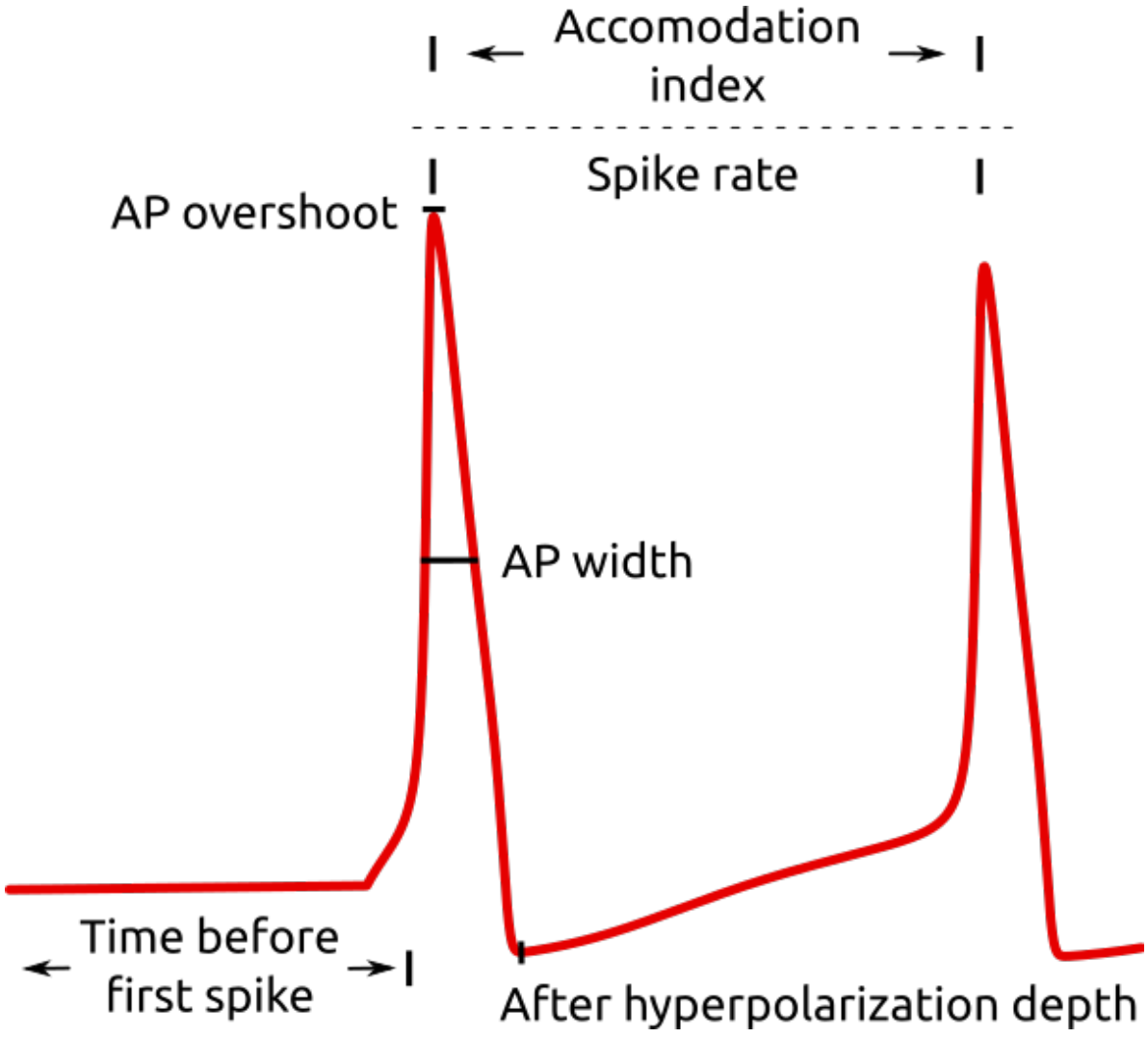
The time shifted spikes cause large variance



The solution is to compare features of the model, such as the number of spikes



Uncertainpy calculates the uncertainty for a user selected set of features of the model



Uncertainpy comes with pre-defined features that can be directly used with the `features` argument

```
UQ = un.UncertaintyQuantification(
    model=hodgkin_huxley,
    parameters=parameters,
    features=un.SpikingFeatures()
)
```

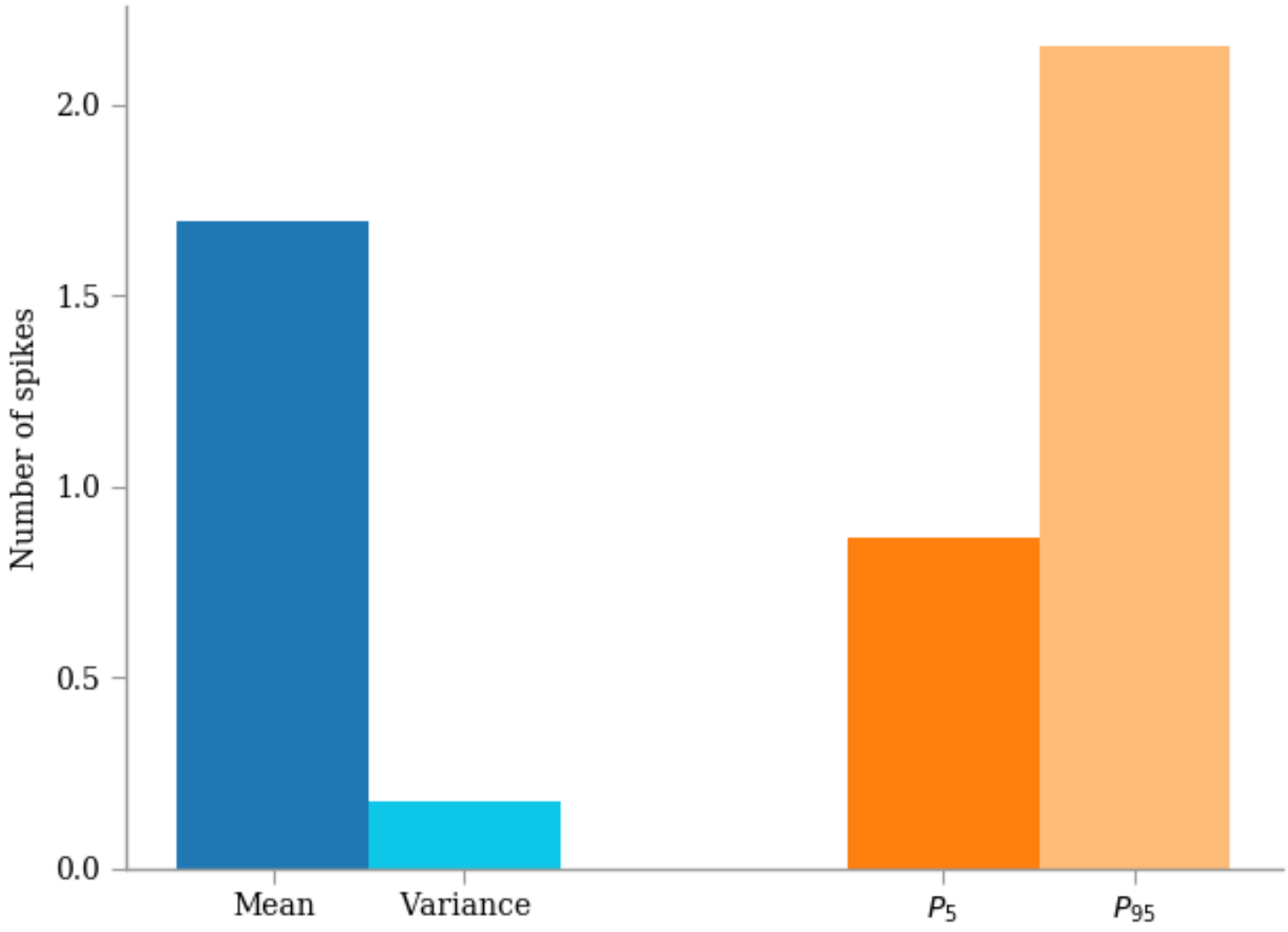
Features are defined as Python functions

```
def nr_spikes(time, membrane_potential):
    # Calculate the feature using
    # time and membrane_potential

    ...

    return time, nr_spikes
```

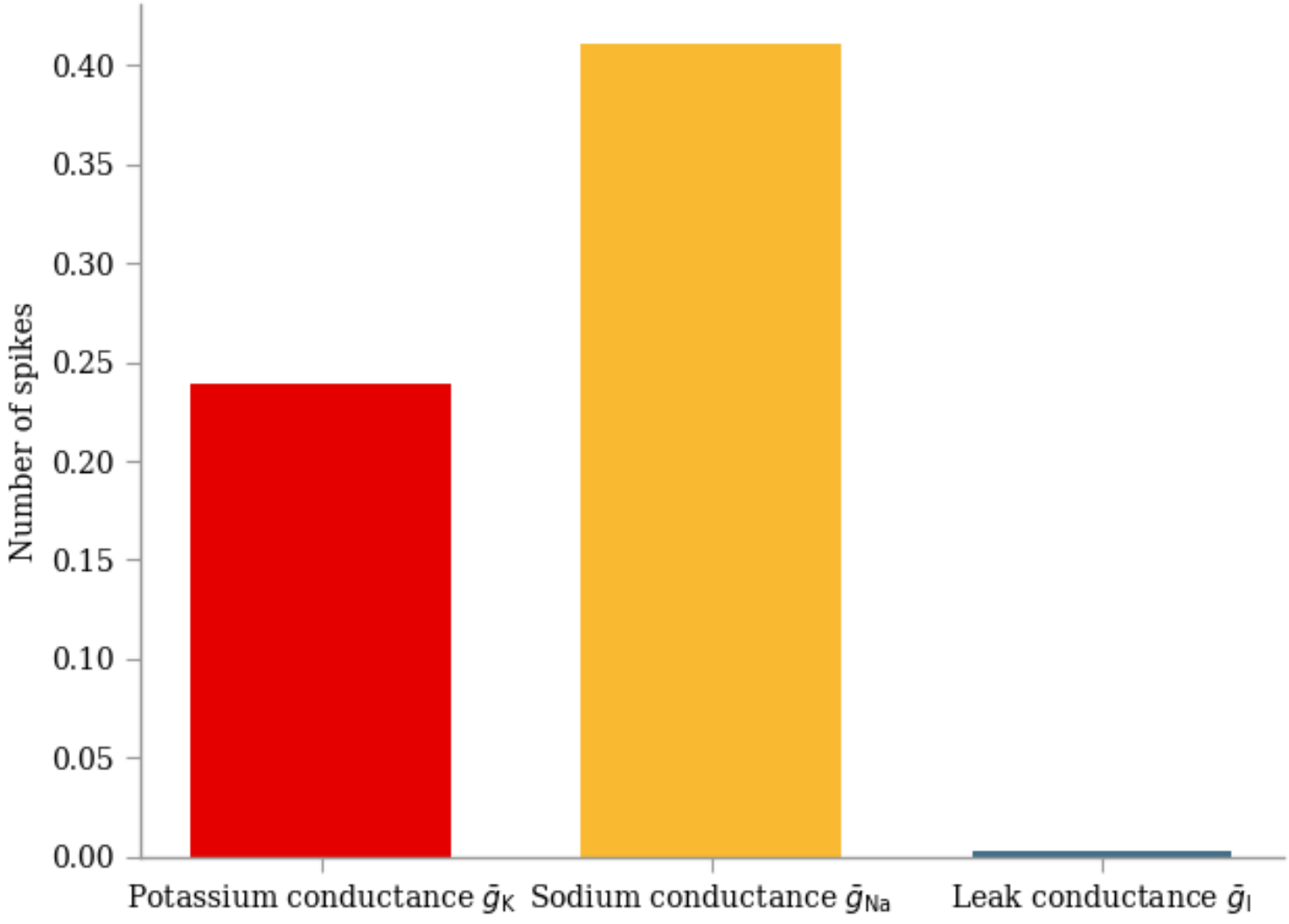
A list of feature functions is given with the `features` argument

```
UQ = un.UncertaintyQuantification(
    model=hodgkin_huxley,
    parameters=parameters,
    features=[nr_spikes, spike_rate]
)
```

The number of spikes of the Hodgkin-Huxley model



Sensitivity of the number of spikes of the Hodgkin-Huxley model



Summary:

Uncertainty quantification enables us to take the effects of uncertain parameters into account

Summary:

Uncertainty quantification enables us to take the effects of uncertain parameters into account

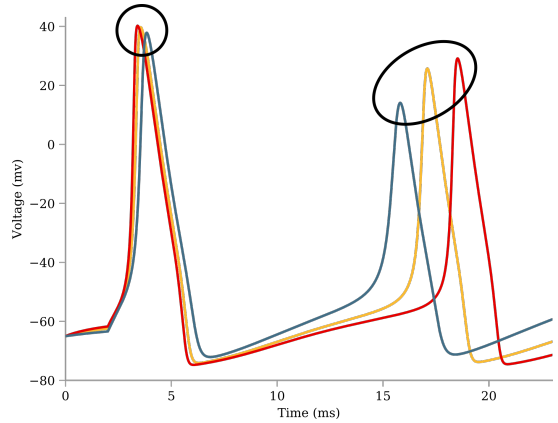
performs all calculations without the need for detailed knowledge

Summary:

Uncertainty quantification enables us to take the effects of uncertain parameters into account

Uncertainpy

performs all calculations without the need for detailed knowledge



Features are useful when comparing spiking results

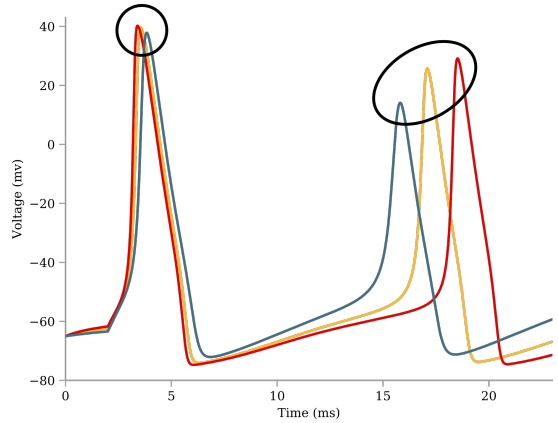
cinpla

Summary:

Uncertainty quantification enables us to take the effects of uncertain parameters into account

Uncertainpy

performs all calculations without the need for detailed knowledge

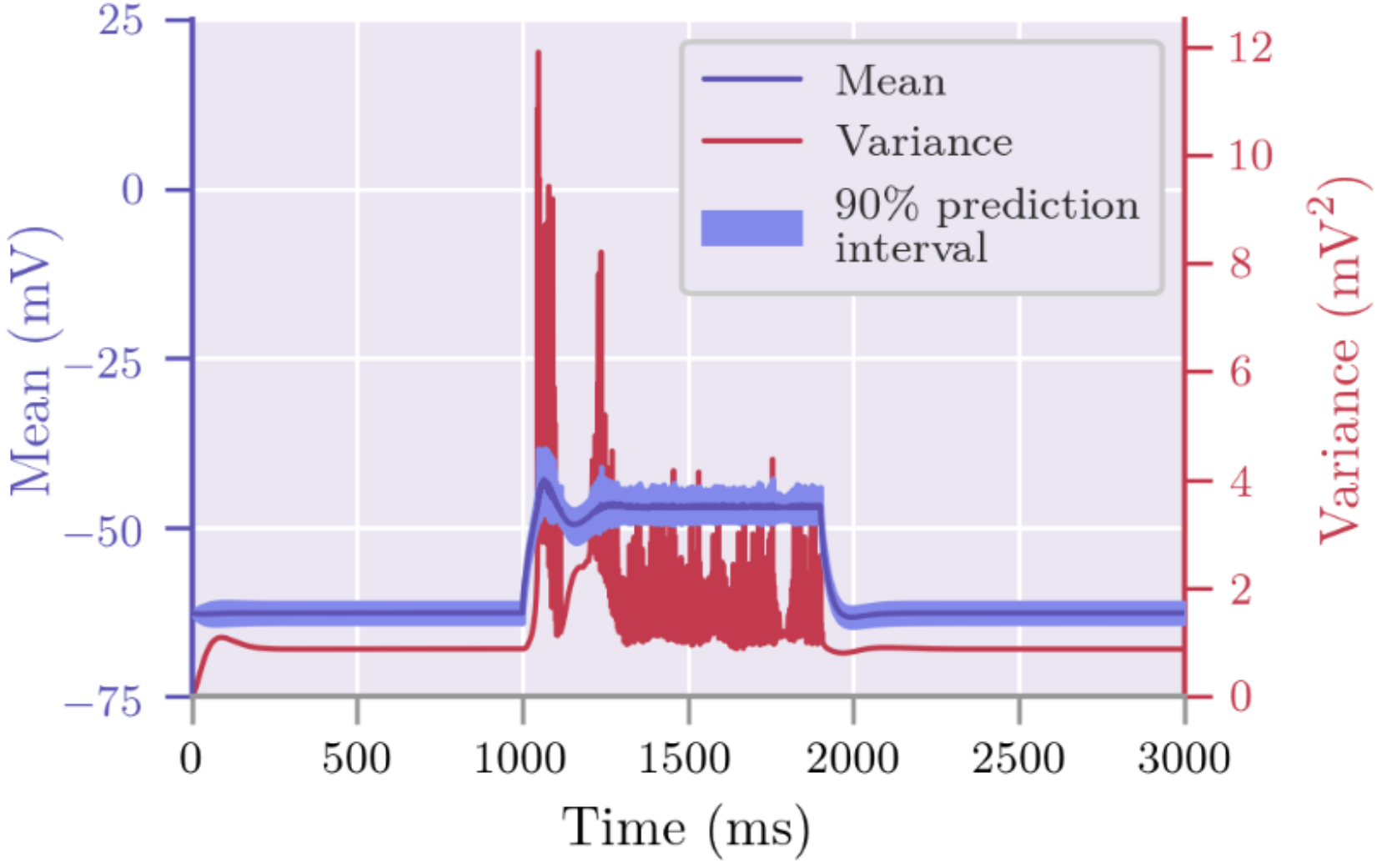


Features are useful when comparing spiking results

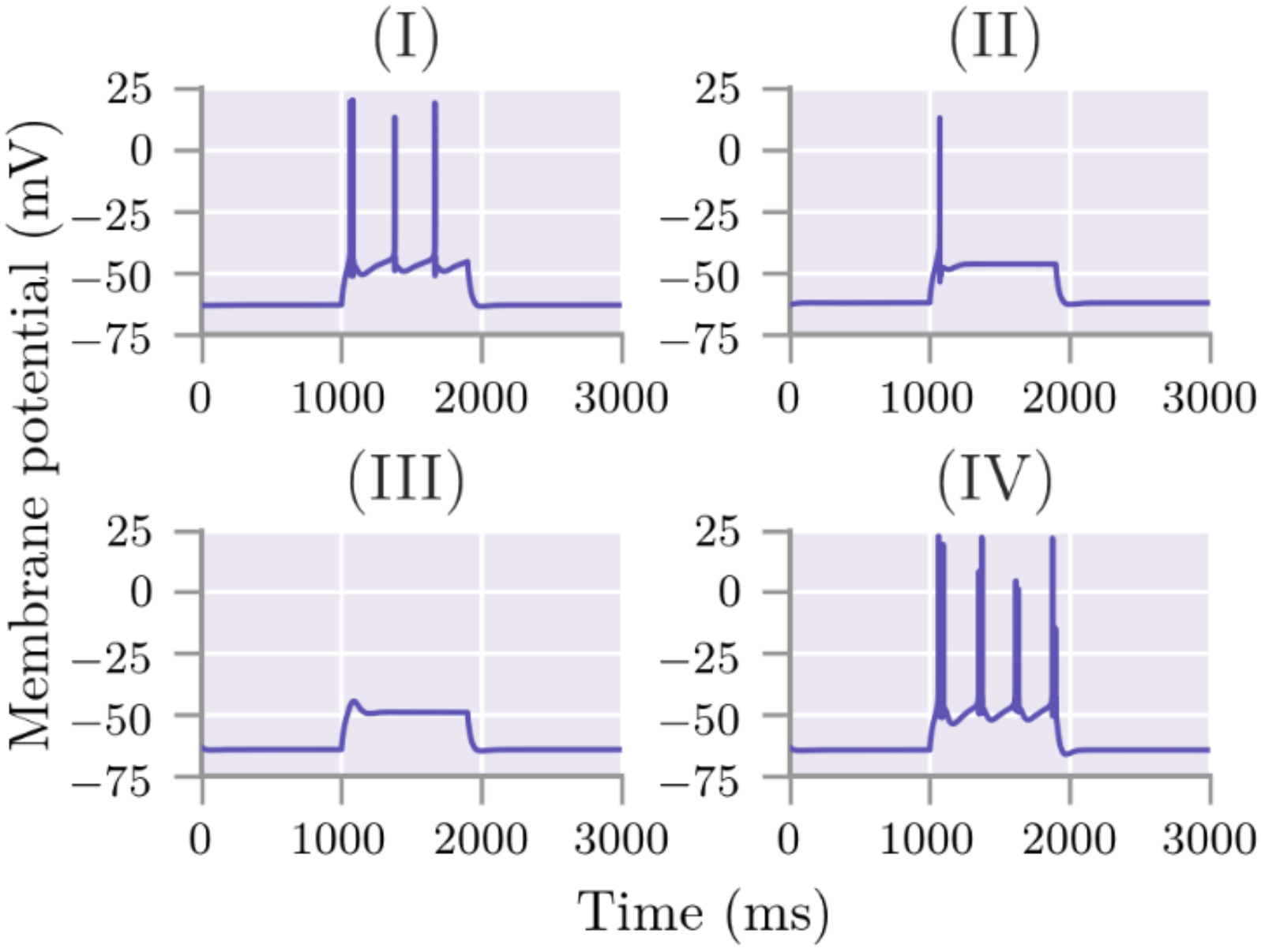
Questions?

cinpla

Membrane potential of a thalamic interneuron



Membrane potential of a thalamic interneuron



Results are saved and loaded with the `Data` class

```
results = un.Data()
results.load("filename")

mean = results["nr_spikes"].mean
variance = results["nr_spikes"].variance
```

NEURON models can be used with the `NeuronModel` class

```
model = un.NeuronModel(
    path="path/to/neuron_model",
    adaptive=True,
    stimulus_start=1000,
    stimulus_end=1900
)
```