Simulation-based Inference

Inverting Simulators

Talk by Stefan Wezel

mlcolab @ Tübingen University Cluster of Excellence

May 3, 2021

Overview

- The problem setting
- Traditional approaches and their issues
- Using Neural Nets to alleviate them
- A worked example
- Outlook

What and Why?

An Example

- Inverting simulators?
- What is a simulator?
  - Forward, generative model with parameters and stochasticity
  - Produces observations
  - In context of this talk computer program, but can be electrical circuit (Hodgkin–Huxley model)
- Used by scientists to model empirical observed data
- Used in particle physics, population genetics, epidemiology
- Encode knowledge about systems gathered over centuries

What and Why?

An Example

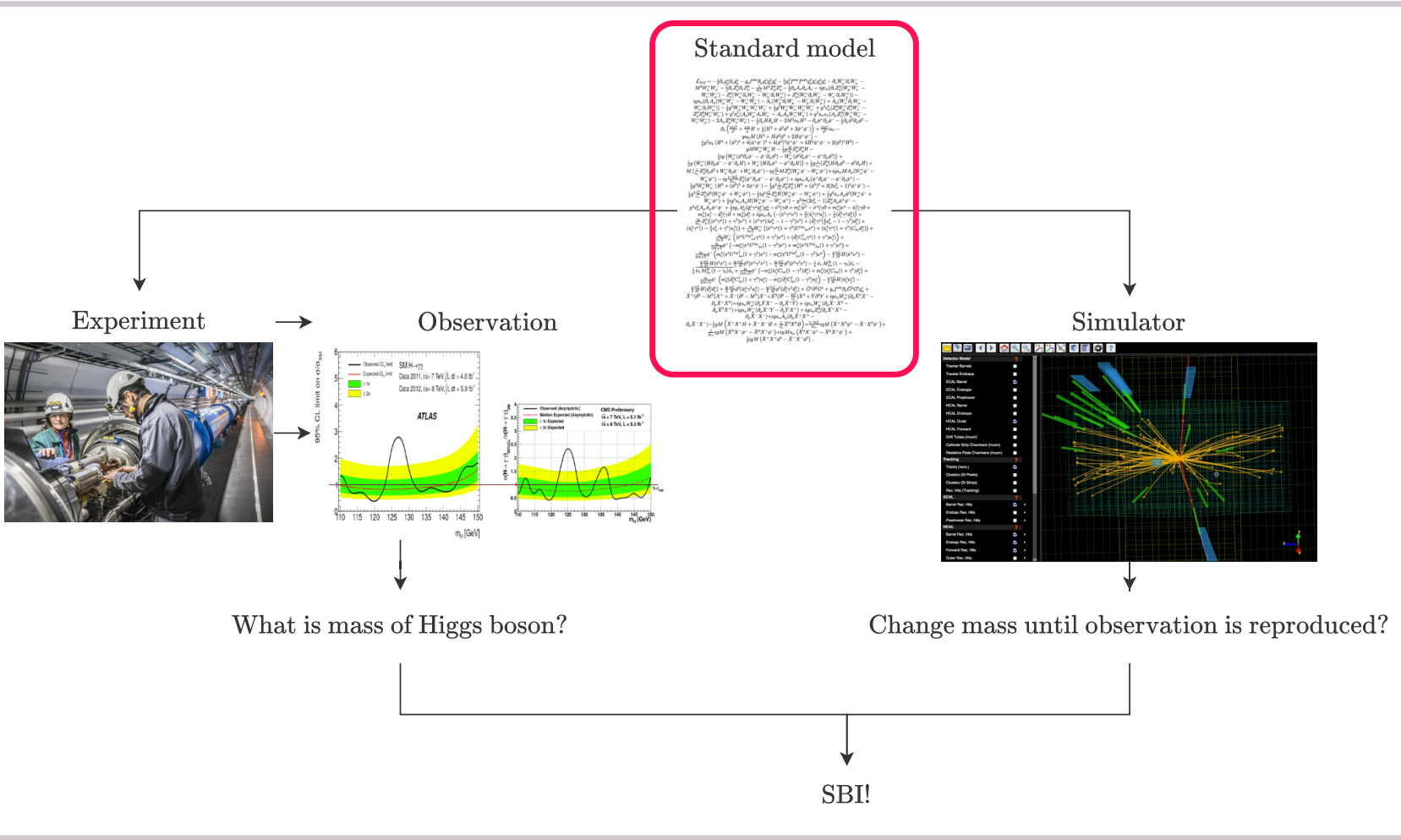


Figure derived from cern.ch, Achintya Rao and Tom McCauley, sciencealert.com

What and Why?

An Example

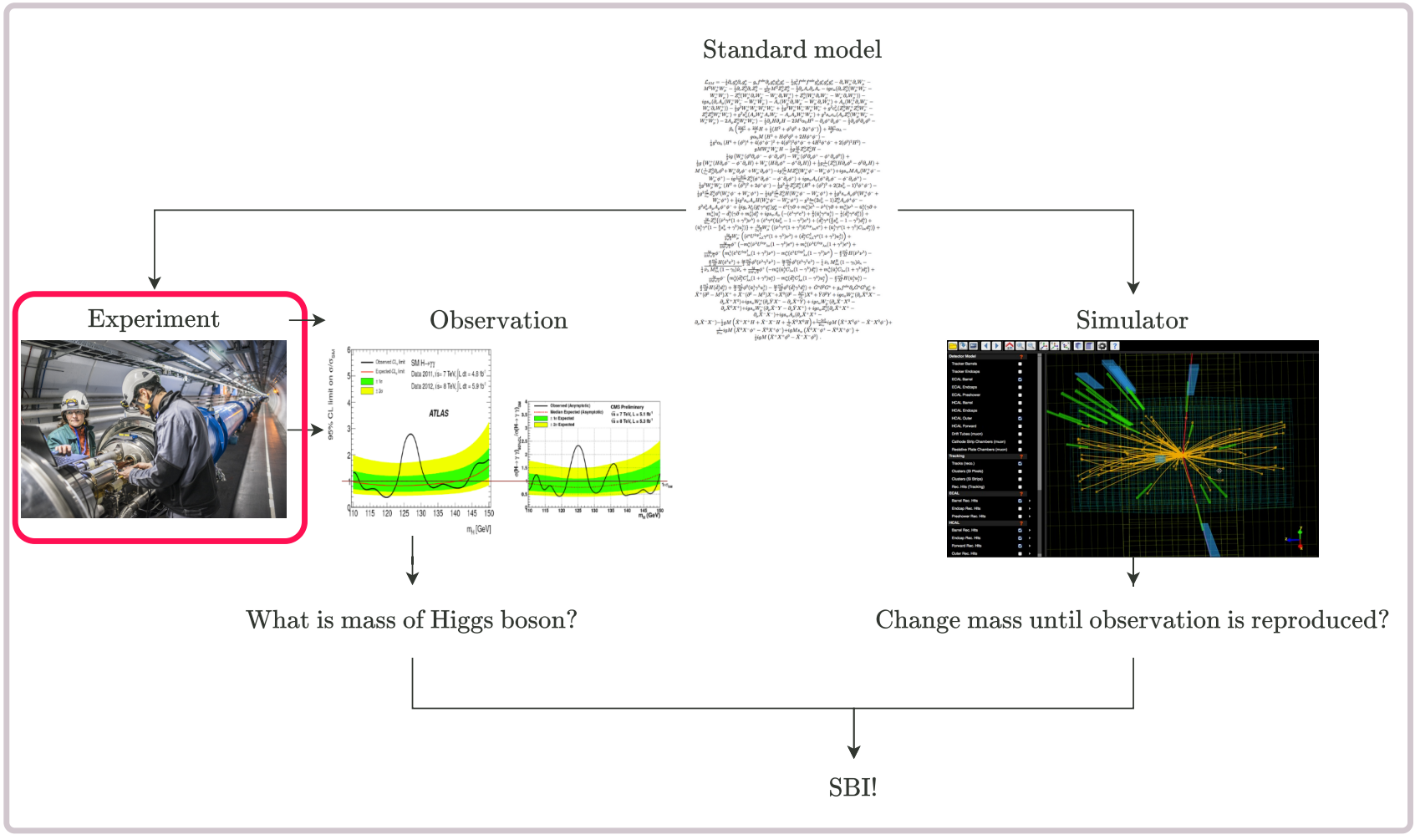


Figure derived from cern.ch, Achintya Rao and Tom McCauley, sciencealert.com

What and Why?

An Example

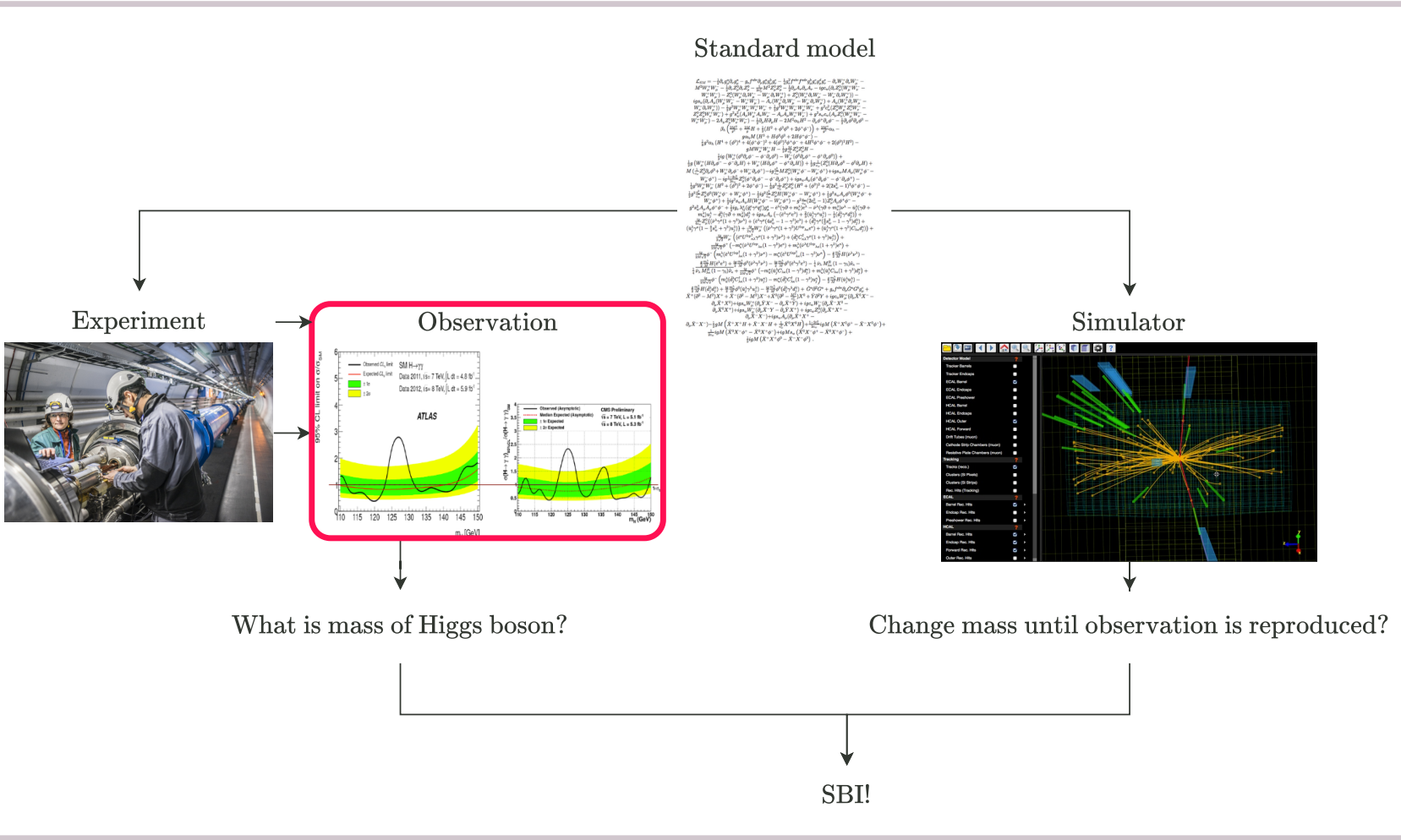


Figure derived from cern.ch, Achintya Rao and Tom McCauley, sciencealert.com

What and Why?

An Example

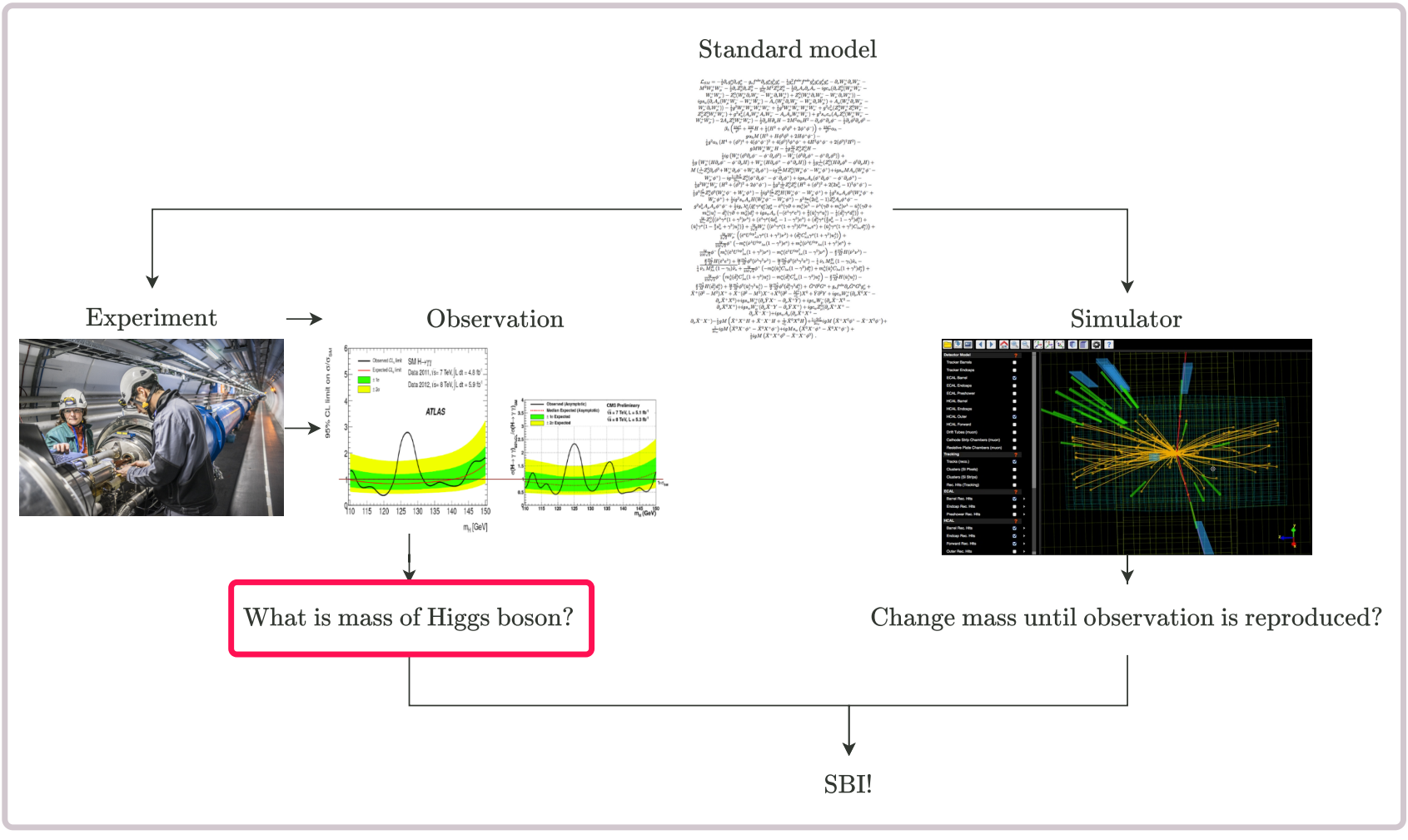


Figure derived from cern.ch, Achintya Rao and Tom McCauley, sciencealert.com

What and Why?

An Example

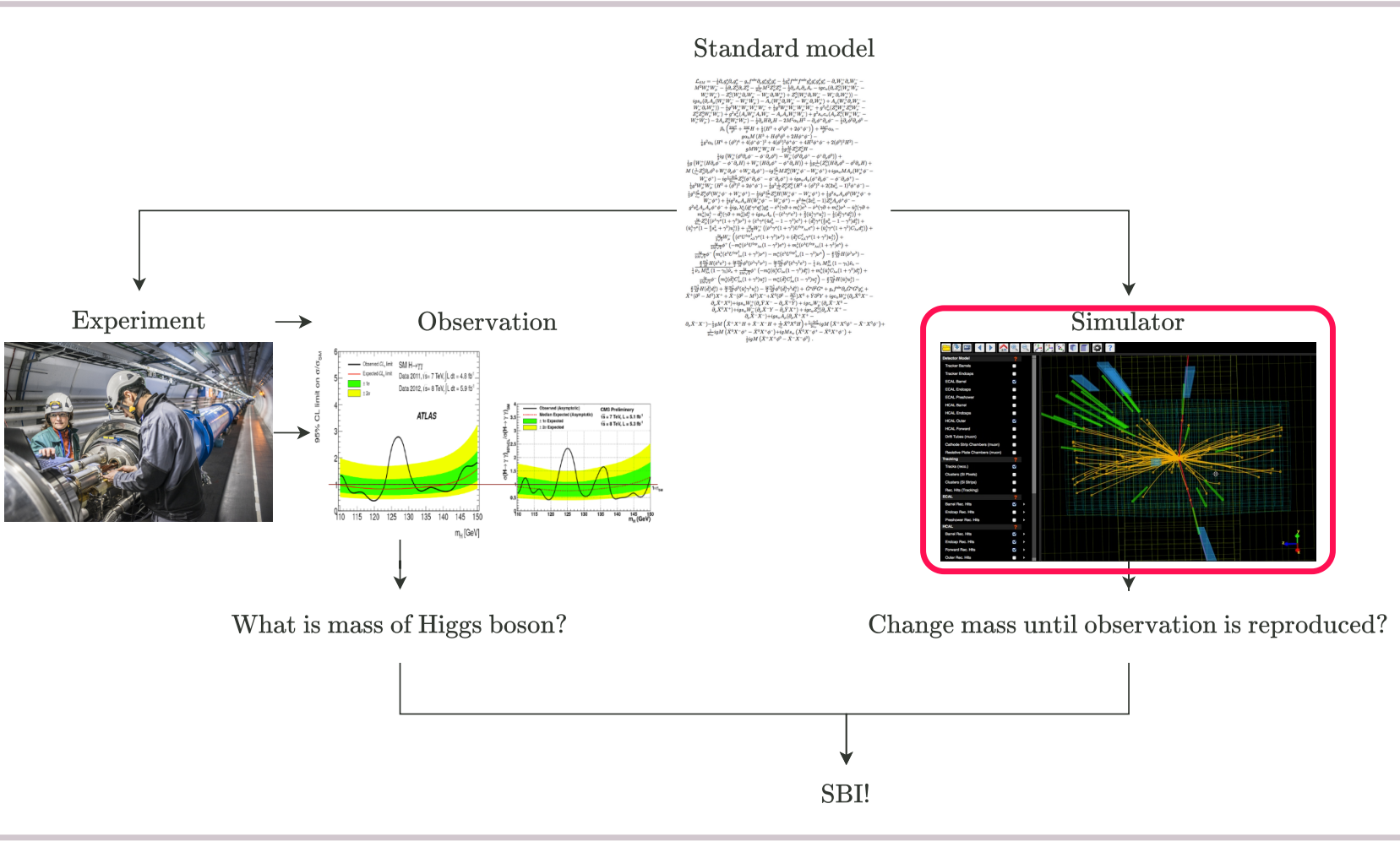


Figure derived from cern.ch, Achintya Rao and Tom McCauley, sciencealert.com

What and Why?

An Example

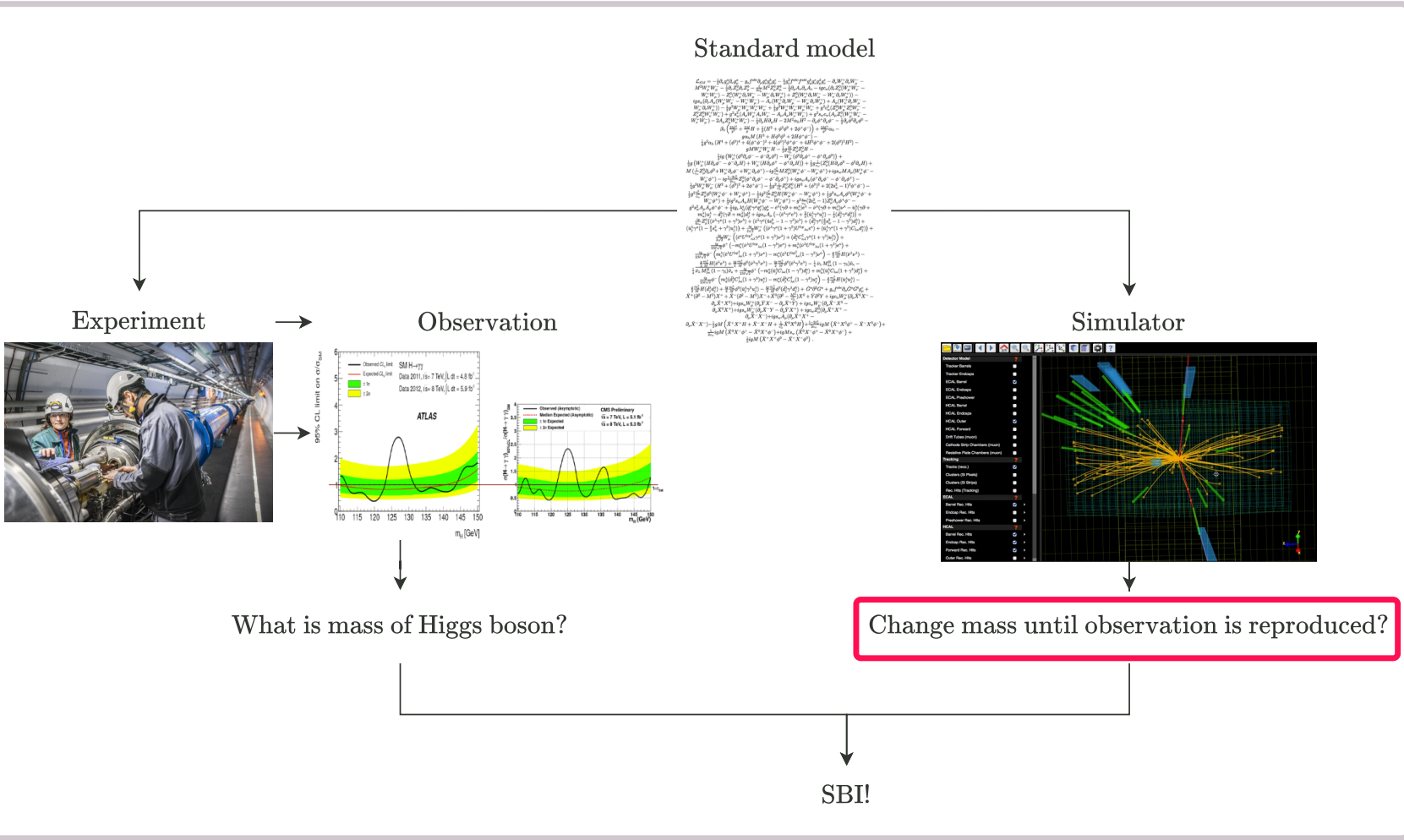


Figure derived from cern.ch, Achintya Rao and Tom McCauley, sciencealert.com

What and Why?

An Example

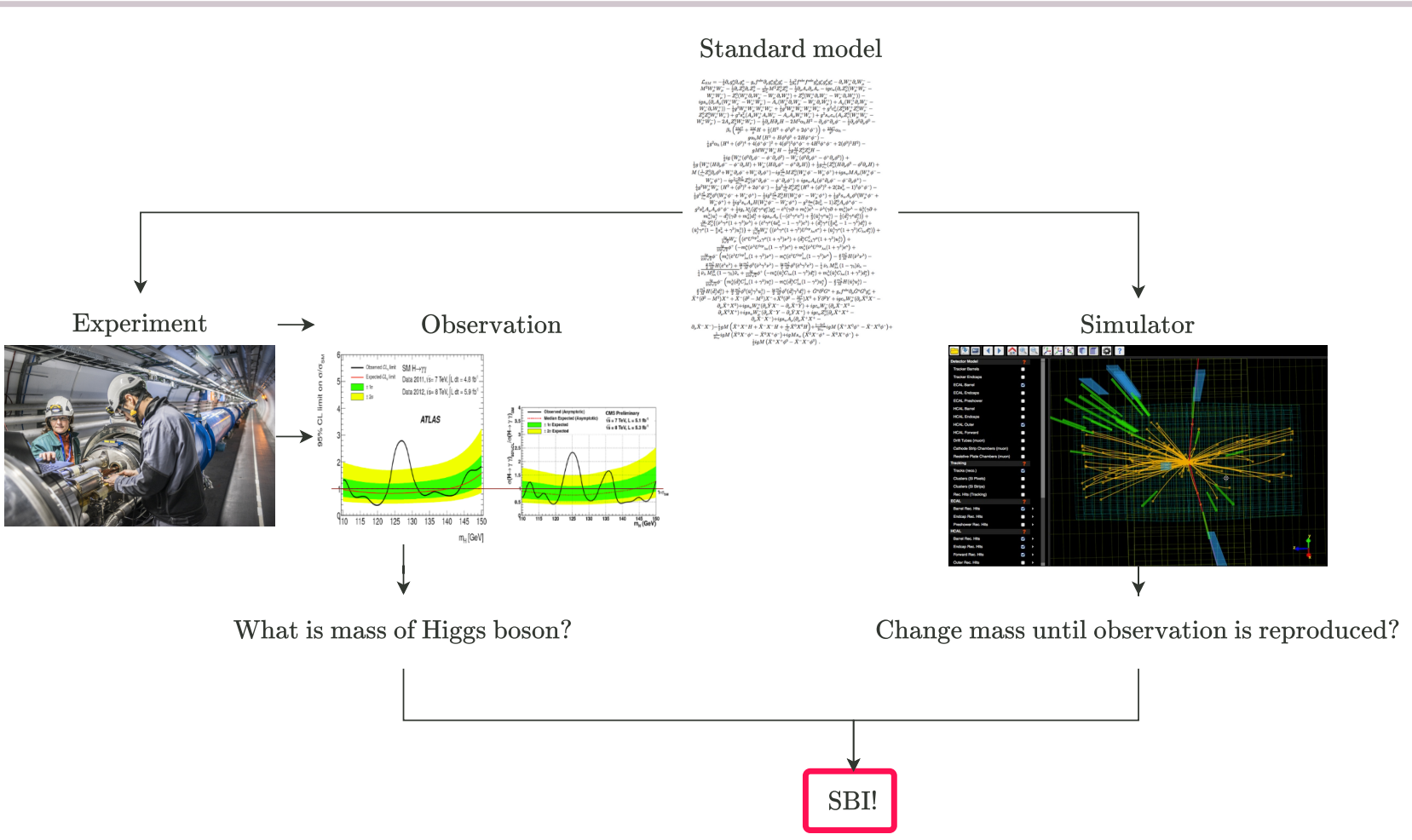


Figure derived from cern.ch, Achintya Rao and Tom McCauley, sciencealert.com

- Before we go into SBI, let's take a minute to formalize our setting
- Simulator has parameters θ, nuisance parameter z and produces data $\hat{x}$
  - It implicitly defines $p(\hat{x}, z|\theta)$ (it yields samples from this distribution)
- Given a (real) observation x_0, we would be interested in the parameters that produced it

$$p(\theta|x = x_0) = \frac{p(x|\theta)p(\theta)}{p(x)} = \frac{\overbrace{\int p(x, z|\theta)dz}^{intractable}\, p(\theta)}{\underbrace{\int p(x|\theta)p(\theta)d\theta}_{intractable}}, \tag{1}$$

What and Why?

Simulation-based Inference to the rescue

- This is exactly a problem setting that scientists often deal with
  - They have a sophistic model (the simulator) that encapsulates a lot of prior knowledge
  - But inference in this setting often boils down to: What were parameters that produced this observation
- Simulation-based Inference inference tries to solve this problem
- by inverting the simulator

Simulation-based Inference

Traditional Approach

- Broader term: Approximate Bayesian Computation (ABC)
  - Rejection ABC
  - Sampling ABC (perturb initial parameters)
  - Sequential ABC (importance sampling)
- But gives only point estimates, not full posterior
- within ϵ
- [4] propose to use neural networks to parameterize GMM

for $n = 1..N$ do
  sample $\theta_n \sim \tilde{p}(\theta)$
  sample $x_n \sim p(x|\theta_n)$
end
train $q_\phi(\theta|x)$ on $\{\theta_n, x_n\}$
$\hat{p}(\theta|x = x_0) \leftarrow \frac{p(\theta)}{\tilde{p}(c)} q_\phi(\theta|x)$

Pseudocode derived from [4]

for $n = 1..N$ do

  sample $\theta_n \sim \tilde{p}(\theta)$

  sample $x_n \sim p(x|\theta_n)$

end

train $q_\phi(\theta|x)$ on $\{\theta_n, x_n\}$ (More in a second)

$\hat{p}(\theta|x = x_0) \leftarrow \frac{p(\theta)}{\tilde{p}(c)} q_\phi(\theta|x)$

Pseudocode derived from [4]

for $n = 1..N$ do

  sample $\theta_n \sim \tilde{p}(\theta)$

  sample $x_n \sim p(x|\theta_n)$

end

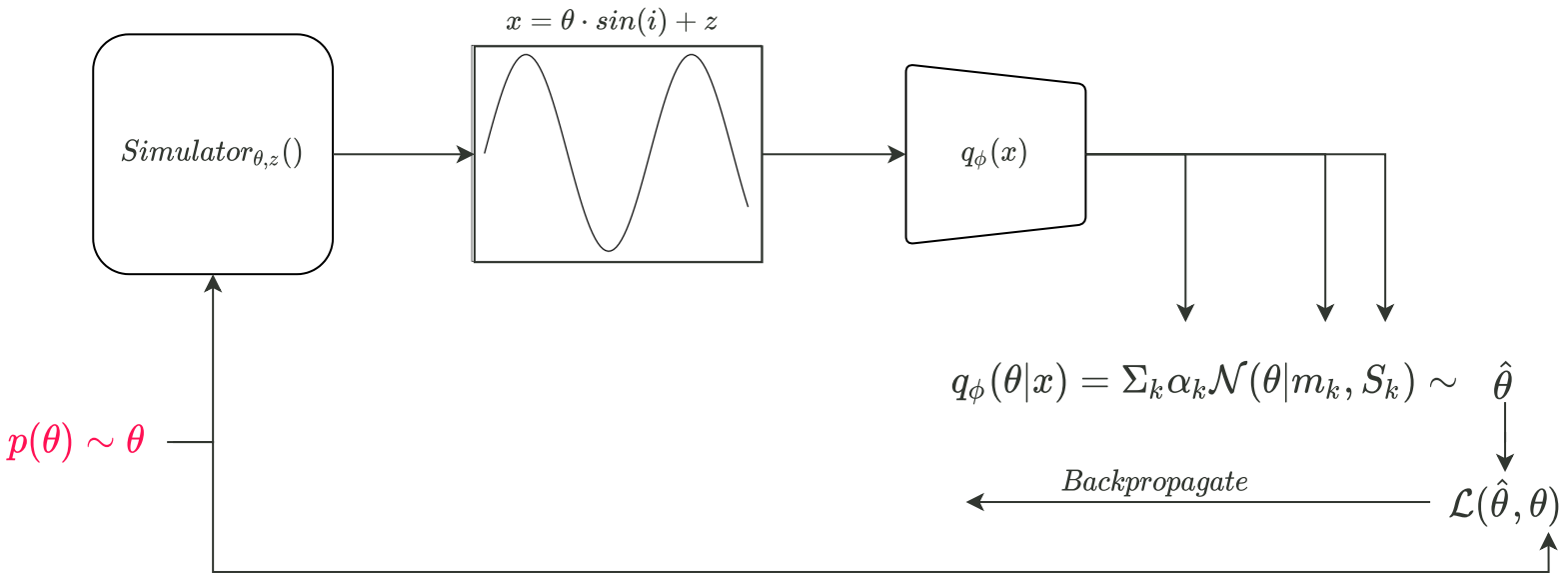
train $q_\phi(\theta|x)$ on $\{\theta_n, x_n\}$

$\hat{p}(\theta|x = x_0) \leftarrow \frac{p(\theta)}{\tilde{p}(c)} q_\phi(\theta|x)$

Pseudocode derived from [4]

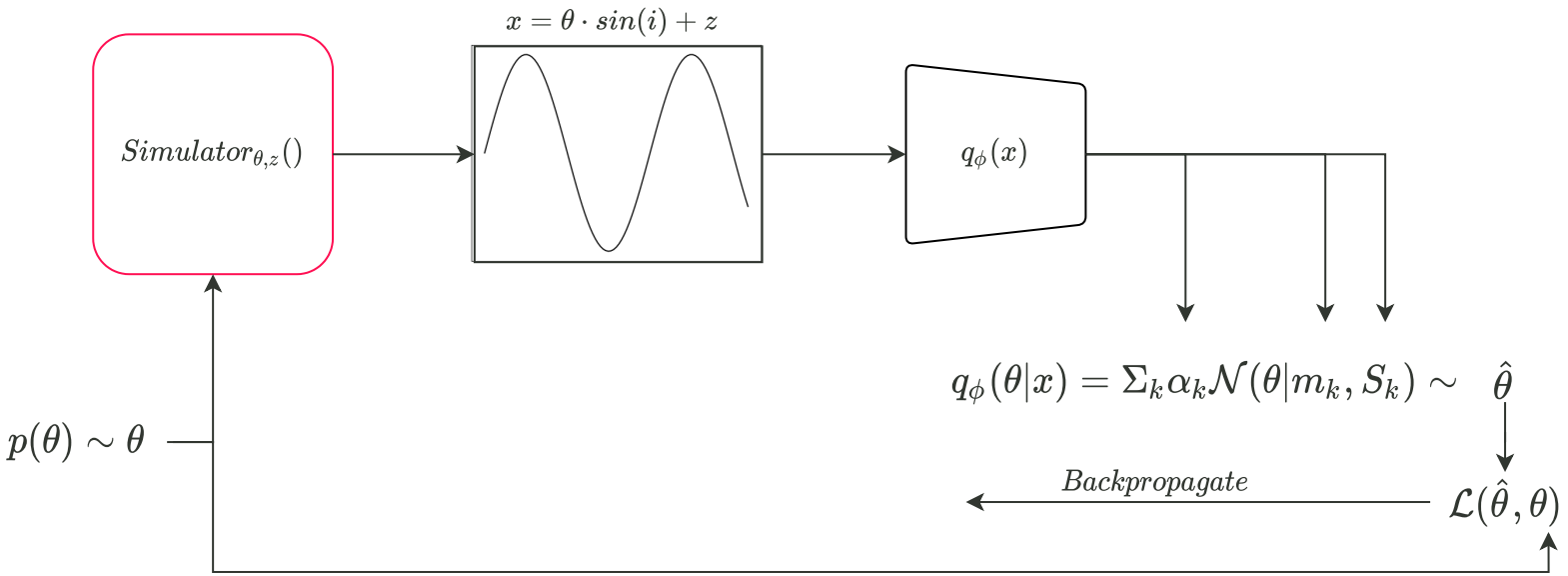
Learning a Posterior

from Simulations



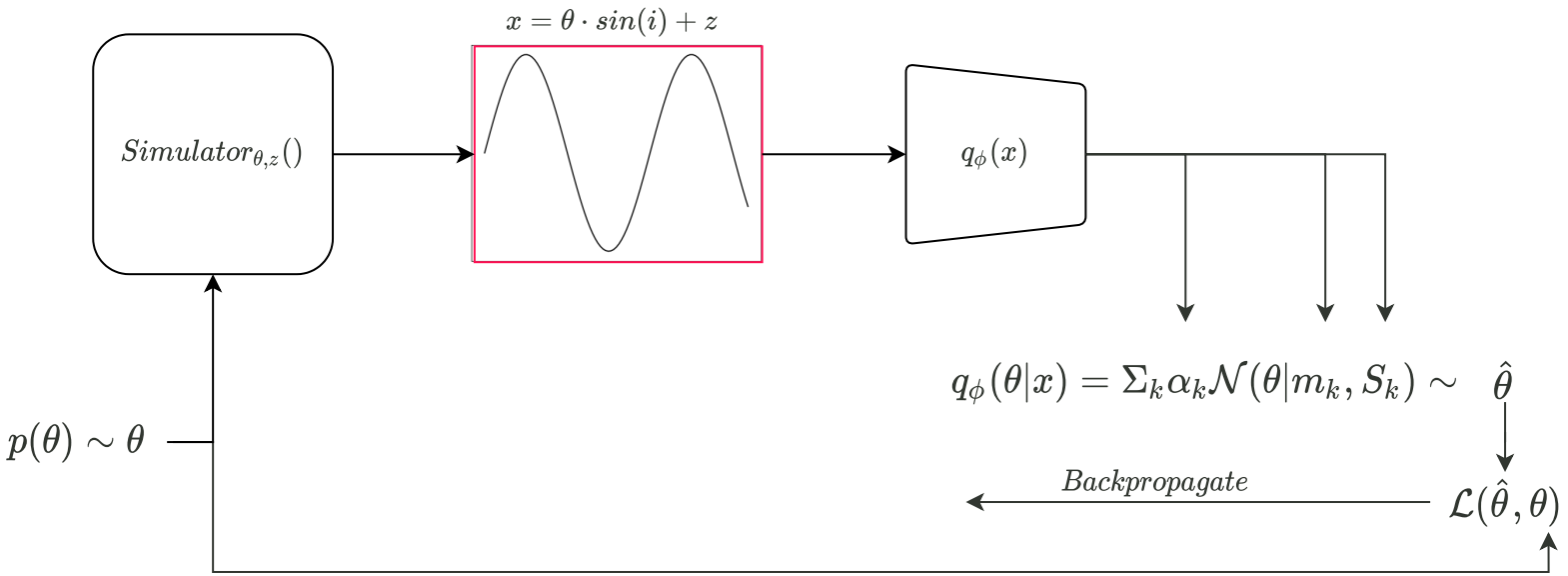
Learning a Posterior

from Simulations



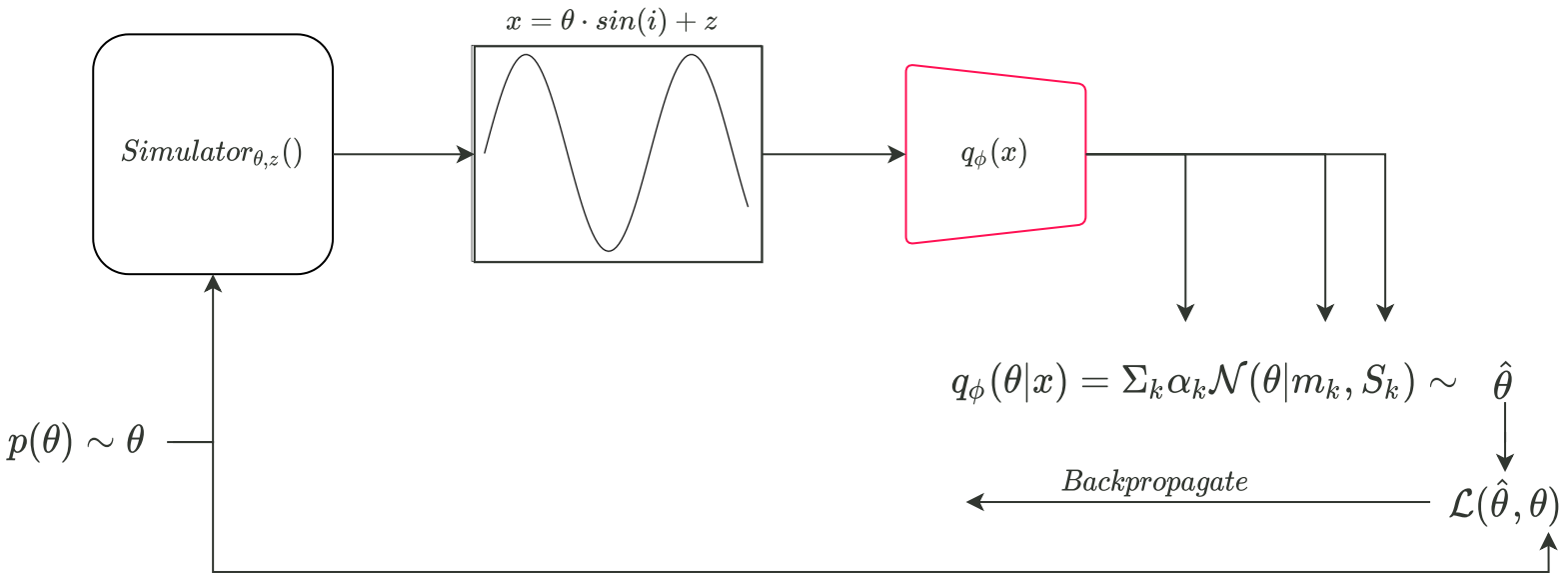
Learning a Posterior

from Simulations



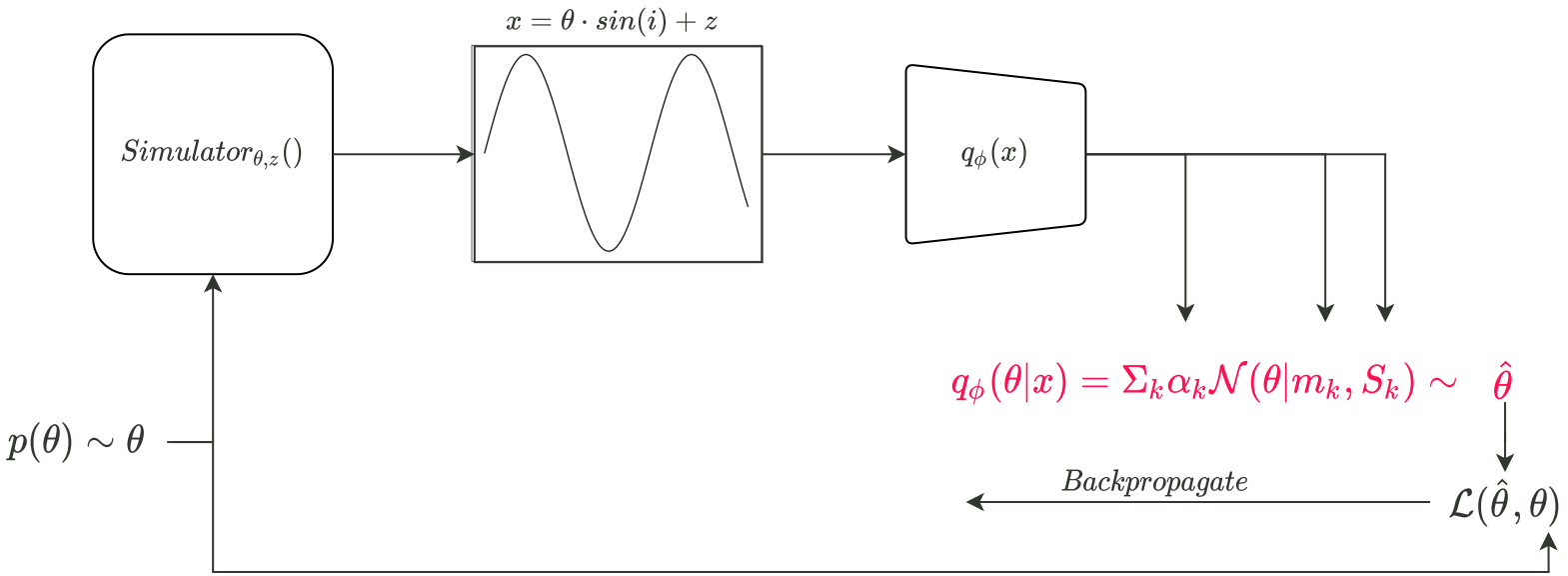
Learning a Posterior

from Simulations



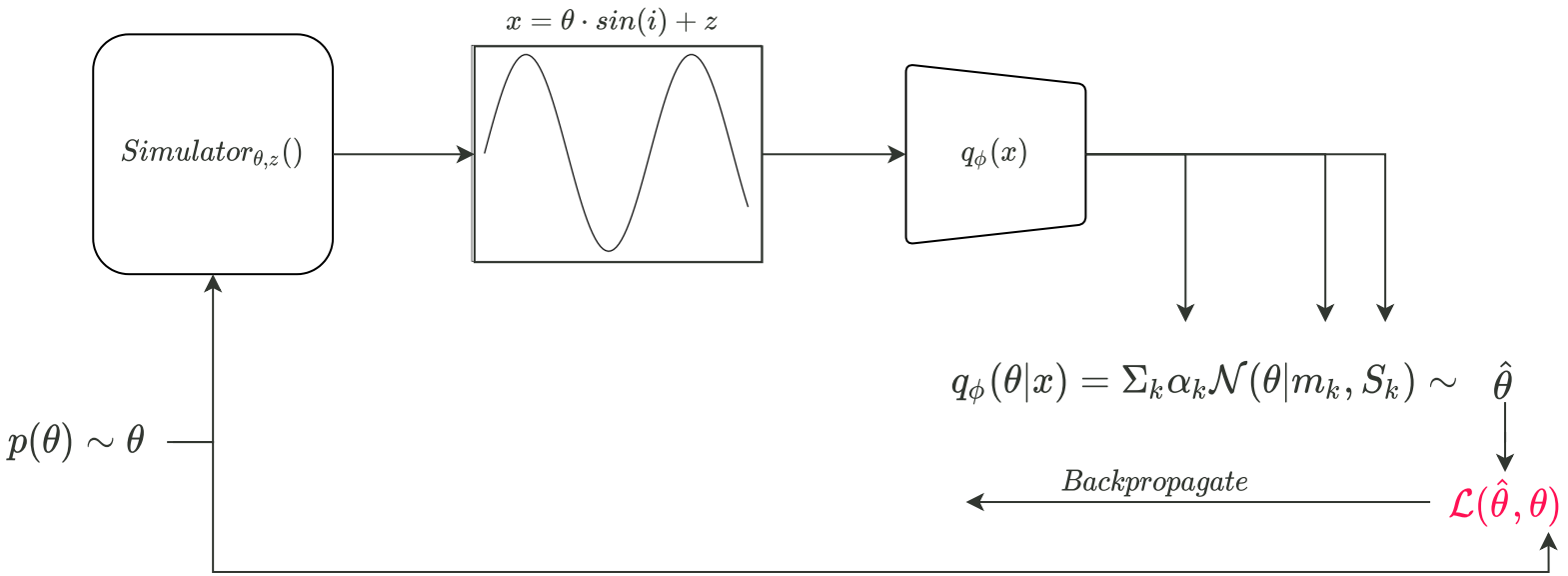
Learning a Posterior

from Simulations



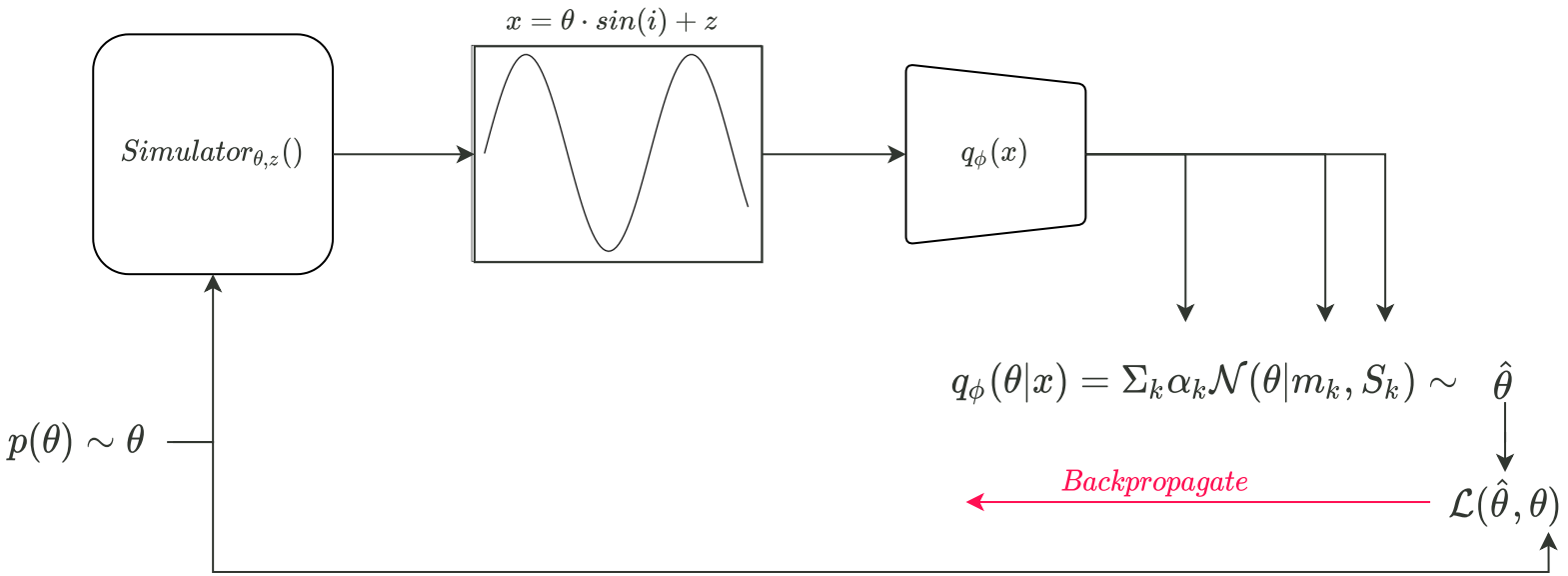
Learning a Posterior

from Simulations



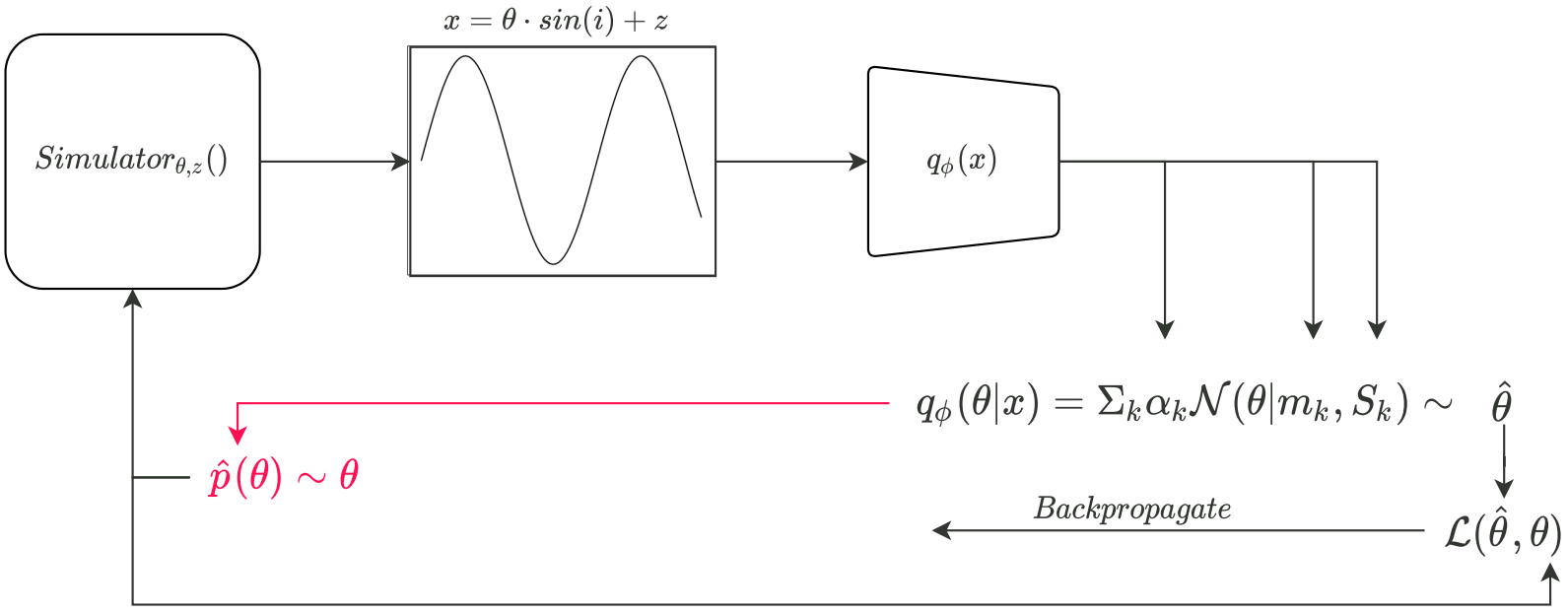
Learning a Posterior

from Simulations



Learning a Prior

Proposing a New Prior



Learning a Prior

Starting from an Initial Prior

$\tilde{p}(\theta) \leftarrow p(\theta)$

repeat

  for $n = 1..N$ **do**

    sample $\theta_n \sim \tilde{p}(\theta)$

    sample $x_n \sim p(x|\theta_n)$

  end

  train $q_\phi(\theta|x)$ on $\{x_n, \theta_n\}$

  $\hat{p}(\theta|x = x_0) \leftarrow \frac{p(\theta)}{\tilde{p}(c)} q_\phi(\theta|x)$

until $\tilde{p}(\theta)$ has converged;

Pseudocode derived from [4]

$\tilde{p}(\theta) \leftarrow p(\theta)$

repeat

  for $n = 1..N$ **do**

    sample $\theta_n \sim \tilde{p}(\theta)$

    sample $x_n \sim p(x|\theta_n)$

  end

  train $q_\phi(\theta|x)$ on $\{x_n, \theta_n\}$

  $\hat{p}(\theta|x = x_0) \leftarrow \frac{p(\theta)}{\tilde{p}(c)} q_\phi(\theta|x)$

until $\tilde{p}(\theta)$ has converged;

Pseudocode derived from [4]

$\tilde{p}(\theta) \leftarrow p(\theta)$

repeat

  for $n = 1..N$ **do**

    sample $\theta_n \sim \tilde{p}(\theta)$

    sample $x_n \sim p(x|\theta_n)$

  end

  train $q_\phi(\theta|x)$ on $\{x_n, \theta_n\}$

  $\hat{p}(\theta|x = x_0) \leftarrow \frac{p(\theta)}{\tilde{p}(c)} q_\phi(\theta|x)$

until $\tilde{p}(\theta)$ has converged;

Pseudocode derived from [4]

$\tilde{p}(\theta) \leftarrow p(\theta)$

repeat

  for $n = 1..N$ **do**

    sample $\theta_n \sim \tilde{p}(\theta)$

    sample $x_n \sim p(x|\theta_n)$

  end

  train $q_\phi(\theta|x)$ on $\{x_n, \theta_n\}$

  $\hat{p}(\theta|x = x_0) \leftarrow \frac{p(\theta)}{\tilde{p}(c)} q_\phi(\theta|x)$

until $\tilde{p}(\theta)$ has converged;

Pseudocode derived from [4]

Learning a Prior

Replacing the Initial Prior

$\tilde{p}(\theta) \leftarrow p(\theta)$

repeat

  for $n = 1..N$ **do**

    sample $\theta_n \sim \tilde{p}(\theta)$

    sample $x_n \sim p(x|\theta_n)$

  end

  train $q_\phi(\theta|x)$ on $\{x_n, \theta_n\}$

  $\hat{p}(\theta|x = x_0) \leftarrow \frac{p(\theta)}{\tilde{p}(c)} q_\phi(\theta|x)$

until $\tilde{p}(\theta)$ has converged;

Pseudocode derived from [4]

Refining SBI

Lueckman et al

- Include importance reweighting in loss
  - No analytical step necessary
  - Enables complex proposal priors
- posterior over network weights, that can be used to init network in next iteration
- Handle bad simulations to improve convergence (i.e. diverging firing rate)
  - training classifier to predict if sample fails
- (Extract Features with RNN)

Applying SBI

to Hodgkin–Huxley

- Hodgkin-Huxley model [2] describes dynamics of neuron's membrane
- Parameters (concentration of sodium, potassium, ...)
- Numerical simulators available (NEURON software [1])

- ...

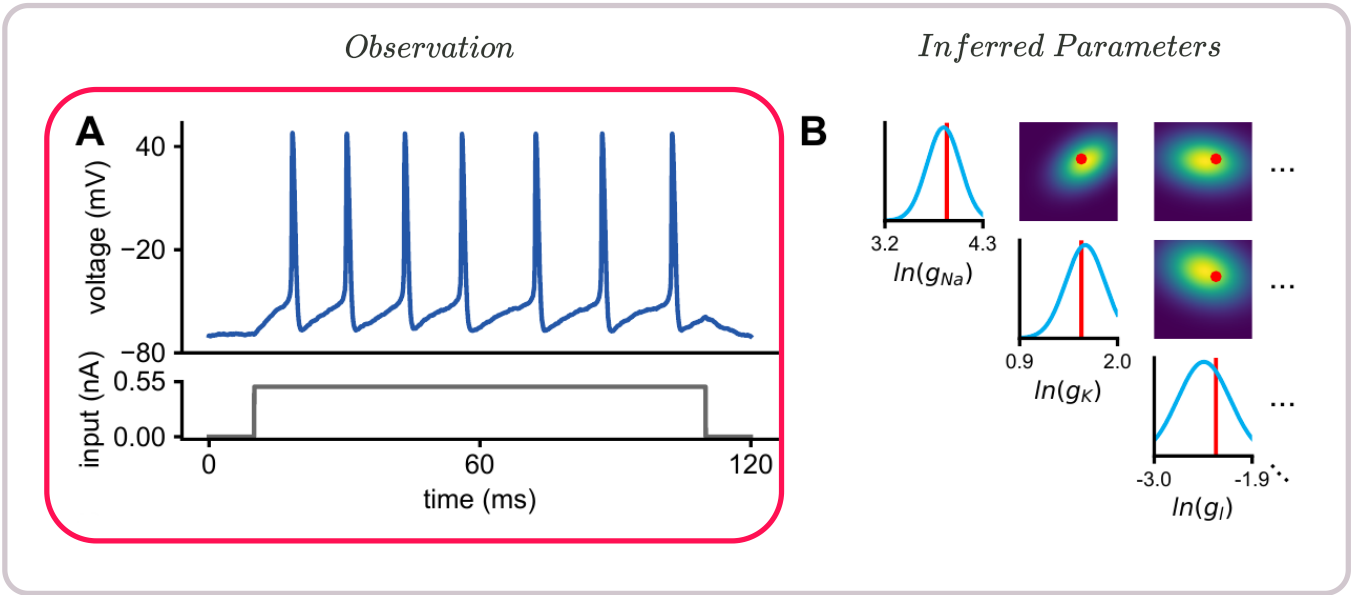


Figure derived from [3]

- ...

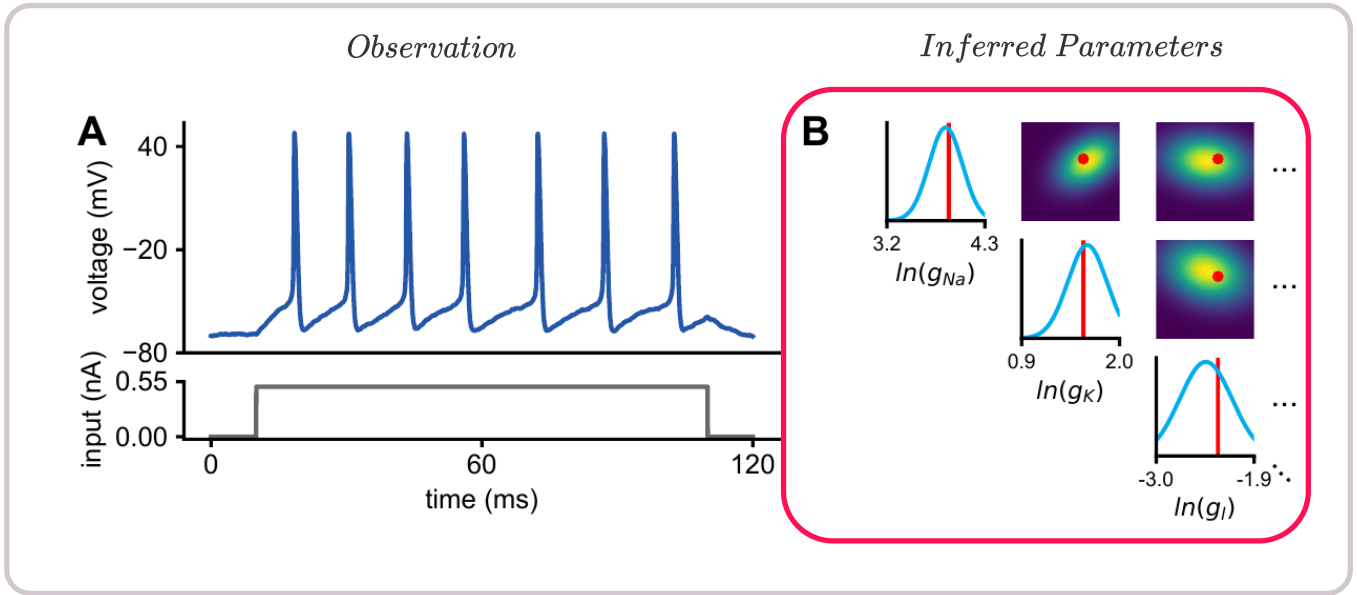


Figure derived from [3]

How to use SBI?

Some advertisement

- Now that you are pretty excited about SBI and its applications
- mlcolab and mackelab is developing a python (and eventually Julia) library (that I've used earlier)
- If you are interested: try it out and give us your feedback
- link to colab
- link to github

References

[1] N. T. Carnevale and M. L. Hines. The NEURON book. Cambridge University Press, 2006.

[2] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. The Journal of physiology, 117(4):500–544, 1952.

[3] J.-M. Lueckmann, P. J. Goncalves, G. Bassetto, K. Öcal, M. Nonnenmacher, and J. H. Macke. Flexible statistical inference for mechanistic models of neural dynamics. arXiv preprint arXiv:1711.01861, 2017.

[4] G. Papamakarios and I. Murray. Fast epsilon-free inference of simulation models with bayesian conditional density estimation. arXiv preprint arXiv:1605.06376, 2016.