Bayesim: a tool for fast device characterization with Bayesian inference

Rachel Kurchin, Giuseppe Romano, and Tonio Buonassisi*
Department of Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA

Target journal: Computer Physics Communications

INTRODUCTION

There are a plethora of examples across diverse scientific and engineering fields of mathematical models used to simulate the results of experimental observations. In many cases, there are input parameters to these models which are difficult to determine via direct measurement, and it is desirable to invert the numerical model – that is, use the experimental observations to determine values of the input parameters. Bayesian inference is a fruitful framework within which to do such fitting, since the resulting posterior probability distribution over the parameters of interest can give rich insights into not just the most likely values of the parameters, but also uncertainty about these values and the potentially complicated ways in which they can covary to give equally good fits to observations. We have previously demonstrated the value of a Bayesian approach in using automated high-throughput temperature- and illumination-dependent current-voltage measurements (JVTi) to fit material/interface properties and defect recombination parameters in photovoltaic (PV) absorbers [1, 2]. In cases such as these, when the data model is not a simple analytical equation but rather a computationally intensive numerical model, efficient, nonredundant sampling of the parameter space when computing likelihoods becomes critical to making the fit feasible. In this work, we introduce extttbayesim, a Python-based code that utilizes adaptive grid sampling to perform Bayesian parameter estimation. We discuss the structure of the code, its implementation, and provide several examples of its usage. While the authors' expertise is in the realm of semiconductor physics and thus the examples herein are drawn from that space, we also discuss the general characteristics of a problem amenable to this approach so that researchers from other fields might adopt it as well.

MODEL

Should this section titel actually just be "Technical Background" as well perhaps?

Bayes' Theorem states

$$P(H|E) = \frac{P(H)P(E|H)}{P(E)} \tag{1}$$

where H is a *hypothesis* and E the observed *evidence*. $P(H)$ is termed the *prior*, $P(E|H)$ the *likelihood*, $P(H|E)$ the *posterior*, and $P(E)$ is a normalizing constant. If there are n pieces of evidence, this can generalize to an iterative process where

$$P(H|\{E_1, E_2, ...E_n\}) = \frac{P(H|\{E_1, E_2...E_{n-1}\})P(E_n|H)}{P(E_n)} \tag{2}$$

In a multidimensional parameter estimation problem, each hypothesis H is a tuple of possible values for the fitting parameters, i.e. a point in the parameter space. In `bayesim`, likelihoods are calculated for each point using a Gaussian where the argument is the difference between observed and simulated output and the standard deviation is the sum of experimental uncertainty and model uncertainty. The experimental uncertainty is a number provided by the user, while the model uncertainty is calculated by `bayesim` and reflects the sparseness of the parameter space grid, i.e. how much simulated output changes from one grid point to another. A high-level flowchart of what `bayesim` does is shown in (**??**)a.

SOFTWARE ARCHITECTURE AND INTERFACE

Structure

The structure of `bayeim` is shown in (**??**). The top-level object with which users interact is implemented in the `Model` class. The `params` module defines classes to store information about the various types of parameters (fitting parameters, experimental conditions, and measured output) while the `Pmf` class stores the probability distribution and implements the manipulations required for Bayesian updates.

Interfaces

describe ways to "talk to" `bayesim`.

Dependencies

This probably makes sense to include also, right?

APPLICATION EXAMPLES

Ideal Diode Model

- validation example - "observed" data is just generated using the model and we show we can recover the correct input parameters
- figure 3 showing PMF

Example with Real Data

- more practical example - probably fitting resistive diode to the same SnS data we used in the Joule paper
- figure 4 showing PMF (animation in ESI) and comparison of JV curves

Maybe an example with a numerical model like PC1D

Maybe a non-PV example

thermoelectrics? TIDLS?

CONCLUSIONS

talk about broader applicability of approach

ACKNOWLEDGEMENTS

APPENDIX

- include minimal code to run ideal diode example

- link to Github repo (which has installation instructions and documentation as well as list of planned future features)

---

* buonassi@mit.edu

[1] R. E. Brandt, R. C. Kurchin, V. Steinmann, D. Kitchaev, C. Roat, S. Levcenco, G. Ceder, T. Unold, T. Buonassisi, and H. Z. Berlin, Joule **1**, 843 (2017).

[2] R. C. Kurchin, J. R. Poindexter, D. Kitchaev, V. Vähänissi, C. del Cañizo, L. Zhe, H. S. Laine, C. Roat, S. Levcenco, G. Ceder, and T. Buonassisi, "Semiconductor parameter extraction via current-voltage characterization and bayesian inference methods," (2018).