Improvements to Inference Compilation for Probabilistic Programming in Large-Scale Scientific Simulators

Mario Lezcano Casado, Atılım Güneş Baydin
David Martínez Rubio, Tuan Anh Le, Frank Wood
Department of Engineering Science
University of Oxford
{lezcano,gunes,damaru,tuananh,fwood}@robots.ox.ac.uk

Lukas Heinrich, Gilles Louppe, Kyle Cranmer
Department of Physics & Center for Data Science
New York University
{cranmer,lukas.heinrich,g.louppe}@cern.ch

Karen Ng
Intel Corporation
karen.y.ng@intel.com

Wahid Bhimji, Prabhat
Lawrence Berkeley National Laboratory
{wbhimji,prabhat}@lbl.gov

Abstract

We consider the problem of Bayesian inference in the family of probabilistic models implicitly defined by stochastic generative models of data. In scientific fields ranging from population biology to cosmology, low-level mechanistic components are composed to create complex generative models. These models lead to intractable likelihoods and are typically non-differentiable, which poses challenges for traditional approaches to inference. We extend previous work in "inference compilation", which combines universal probabilistic programming and deep learning methods, to large-scale scientific simulators, and introduce a C++ based probabilistic programming library called CPPROB. We successfully use CPPROB to interface with SHERPA, a large code-base used in particle physics. Here we describe the technical innovations realized and planned for this library.

1 Introduction

Complex simulations are used for stochastic generative models for data across a wide segment of the scientific community, with applications as diverse as hazard analysis in seismology [12], supernova shock waves in astrophysics [8], market movements in economics [20], and blood flow in biology [19]. In these generative models, complex simulations are composed from low-level mechanistic components. These models lead to intractable likelihoods and are typically non-differentiable, which renders many traditional statistical inference algorithms irrelevant and motivates a new class of so-called likelihood-free inference algorithms [11].

There are two broad strategies for this type of likelihood-free inference problems. In the first, one uses a simulator indirectly to train a surrogate model endowed with a likelihood that can be used in traditional inference algorithms, for example approaches based on conditional density estimation [23, 17, 21, 13] and density ratio estimation [5, 7]. Alternatively, approximate Bayesian computation (ABC) [24, 22] refers to a large class of approaches for sampling from the posterior

Workshop on Deep Learning for Physical Sciences (DLPS 2017), NIPS 2017, Long Beach, CA, USA.

distribution of these likelihood-free models where the original simulator is used directly as part of the inference engine. While variational inference algorithms are often used when the posterior is intractable, they are not directly applicable when the likelihood of the data generating process is unknown.

The class of inference strategies that directly use a simulator avoids the necessity of approximating the generative model. Moreover, using a domain-specific simulator offers a natural pathway for inference algorithms that provide interpretable posterior samples with rich semantics. In this work, we take this approach, extend previous work in probabilistic programming [10] and inference compilation [16] to large-scale complex simulators, and demonstrate the ability to instrument and control simulators written in C++, which is the language of choice for many large-scale simulators in science and industry.

Our work is primarily motivated by applications in high-energy physics (HEP), which studies elementary particles and their interactions using energetic events created in particle accelerators such as the Large Hadron Collider (LHC) at CERN. The observed data are the result of interactions of particles generated in a collision event and observed through particle detectors. From these observations, we want to infer the properties of the particles and interactions that generated them.

Ultimately, our goal is to be able to infer the properties of the Higgs boson [1, 4] using probabilistic programming with the state-of-the-art simulators SHERPA [9], a Monte Carlo "event generator" of high-energy reactions of particles, and GEANT [2], a toolkit for the simulation of the passage of the resulting particles through the detector. Both simulators are implemented in C++ and encapsulate deep domain knowledge in significantly large code bases (> 1 M lines of code), providing a challenging setting for demonstrating our technique and its scalability.

Our key innovations in this paper include: (1) the interfacing of CPPROB with the existing SHERPA code base with very few modifications; (2) the ability to inspect execution traces leading to a novel probabilistic model debugging tool; and (3) a new code annotation scheme for dealing with traces of unbounded length in algorithms such as rejection sampling encountered in HEP settings.

2 CPProb

In order to equip large-scale simulation code bases with inference compilation, we developed CPPROB [3],[1] a probabilistic programming library written in C++14 that allows inference in probabilistic models written in C++. Using a universal probabilistic programming approach means that our library enables inference in existing simulator codes without having to rewrite the model in a specific probabilistic programming language. This is done simply by redirecting calls of the random number generation and introducing annotations to the code defined by CPPROB.

The library exports three main functions: `sample`, `observe`, and `predict`, all of which are statically typed both for efficiency and to offer compile-time type guarantees related to the addressing scheme used in CPPROB. The `sample` and `observe` statements allow CPPROB to assign addresses to and keep track of probabilistic execution traces of the simulator during training and inference. Addresses identify random choices according to their structural position in the execution, similar to a stack trace [25]. A `sample` statement generates a random value from a specified probability distribution, and appends this sample to the execution trace. An `observe` statement specifies the conditioning of a random variable upon an observed value, such as the conditioning of the simulation output on data. A `predict` statement is used to designate any latent values within the simulation that we would like to be reported as the result of inference.

The main algorithm used in CPPROB is importance sampling [6] in conjunction with inference compilation [16]. The inference compilation approach combines universal probabilistic programming and deep learning methods so that an *inference network* is trained to choose the parameters of a family of proposal distributions in a sequential importance sampling (SIS) inference engine. The inference network controls the execution of the simulator and provides substantially more efficient sampling for ABC than Markov chain Monte Carlo (MCMC) methods. Since SIS reaches deep into the control flow of the program, the posterior samples are efficiently obtained.

Interpretability: The ability to connect posterior samples to the simulator code is a key advantage of our method. This connection enables deep interpretability of inference results in the context of

[1] Code will be publicly released at a future time.

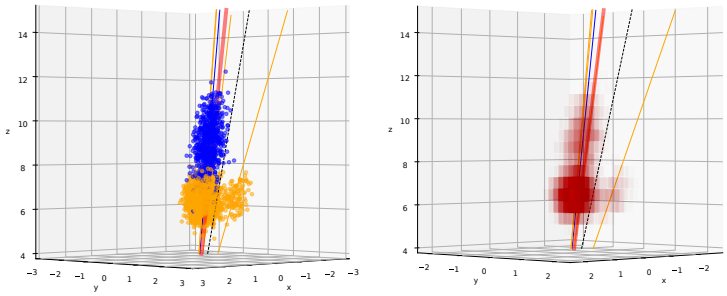


Figure 1: Simulated τ lepton decay event in the channel $\tau \to \nu_\tau \pi^- \gamma\gamma$. The red line is the τ momentum vector. *Left:* The latent momentum vectors (lines) and interactions in the calorimeter (dots) for the interacting τ decay products. *Right:* The observed energy deposits (with latent momentum vectors overlaid).

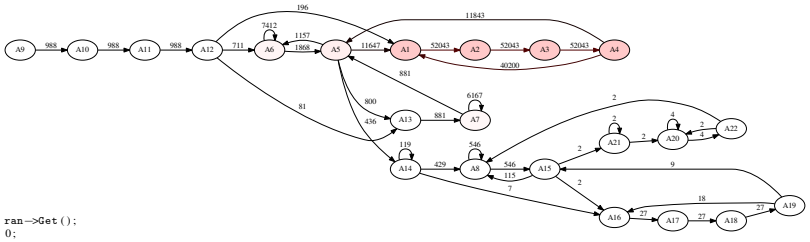
the physically-motivated latent process encoded by the simulator. Such interpretability is crucial for applications in the physical sciences. For instance, this approach gives us the ability to inspect any aspect of the latent process encoded in the simulation, such as the chain of particle decays and interactions within the detector that led to particular posterior predictions. This capability is not possible in inference techniques that do not have access to the simulator, such as those based on neural networks.

3 Results

We focus our initial studies on the decay of τ leptons, whose subsequent decay products interact in the detector. This provides a rich and realistic particle physics use-case, and directly connects to the physics of the Higgs boson. During simulation, SHERPA stochastically selects a set of particles to which the initial τ lepton will decay—a "decay channel"—and samples the momenta of these particles according to a joint density obtained from underlying physical theory. Those particles will then interact in the detector leading to observations in the raw sensor data. While GEANT is typically used to model the interactions in detector, for our initial studies we created a fast, approximate detector simulation for a calorimeter with longitudinal and transverse segmentation (20x35x35). The fast detector simulation deposits most of the energy for electrons and π^0 into the first layers and charged hadrons (e.g., $\pi^\pm$) deeper into the calorimeter with larger fluctuations. Given an observation such as the one presented in Figure 1, we would like to infer the initial momenta of the τ lepton and the decay channel that it followed. We were able to successfully interface CPPROB to SHERPA with minimal code annotations and successfully train an inference network artifact capable of performing inference on simulated test data. In order to achieve this, and to enhance the value of inference in this domain, it was necessary to develop several new features described in the rest of this section.

Trace inspection: We developed a general-purpose probabilistic model debugging/inspection tool, which allows for detailed inspection and visualization of execution traces, mapping them back to underlying code (illustrated in Figure 2). This feature also allows for debugging of problems with the whole chain of applying probabilistic programming to existing large-scale code bases, where the underlying functions and order of execution may be opaque.

In our initial experiments with inference compilation of the τ lepton decay problem in SHERPA, we were faced with probabilistic execution traces that were too long to efficiently backpropagate through in the inference network. While the average trace length was around 250, we encountered traces as long as $10,000$, which made training of the inference network in the manner described in Le et al. [16] prohibitively slow. We were able to find the cause of these very long traces using our inspection tool, which pinpointed a rejection sampling loop in the matrix element implementation present in the code base [15, 14]. In Figure 2(a) we present the address succession graph accumulated over 988 traces, describing the overall probabilistic structure of the τ decay simulation. The nodes indicate `sample` addresses in the execution that—being concatenations of stack positions at runtime—can be traced to positions in the code base. The edges indicate deterministic execution paths of the simulator labeled by the number of times a path is traced. The bottom part of Figure 2(b, c) shows the locations



(a) Address succession graph.

```
325 double rn  = ran->Get();
326 double sum = 0;
327 for (size_t i=0;;++i) {
328   if (i==channels.size()) {
329     rn  = ran->Get();
330     i   = 0;
331     sum = 0.;
332   }
333   sum += channels[i]->Alpha();
334   if (sum>rn) {
335     channels[i]->GeneratePoint(p,cuts);
336     if (nin==2) for (int i(0);i<nin+nout;++i) cms.BoostBack(p[i]);
337     m_lastdice = i;
338     break;
339   }
340 }
```

(b) `SHERPA-MC-2.2.3/PHASIC++/Channels/Multi-Channel.C`

```
72 for(i=nin;i<nin+nout;i++) {
73   C    = 2*ran->Get()-1;
74   S    = sqrt(1-C*C);
75   F    = 2*M_PI*ran->Get();
76   Q    = -log( std::min( 1.0-1.e-10, std::max(1.e-10,ran->Get()*ran->Get()) ) );
77   p[i] = Vec4D(Q, Q*S*::sin(F), Q*S*cos(F), Q*C);
78   R    += p[i];
79 }
```

(c) `SHERPA-MC-2.2.3/PHASIC++/Channels/Rambo.C`

Figure 2: The source of very long execution traces within the SHERPA code base pinpointed by our probabilistic model debugging tool. (a) The probabilistic structure of the τ decay simulator. (b) The A5 node maps to the "`ran->Get()`" call in this source file. (c) The A1 – A4 nodes map to the four "`ran->Get()`" calls in this source file.

in the SHERPA source code where `sample` statements A1 – A5 reside. Precisely identifying these `sample` statements enabled us to overcome this bottleneck as described below.

Rejection sampling scheme: Rejection sampling sections in the simulator pose a problem for our approach, as they define execution traces that are a priori unbounded; and since the inference network has to backpropagate through every sampled value, this makes the training significantly slower. Rejection sampling is key to the application of Monte Carlo methods for evaluating matrix elements and other stages of event generation in particle physics; thus an efficient treatment of this construction is primal. We solve this problem by implementing a novel scheme where only the last instances of random choices in a rejection sampling loop are considered during training. During inference we just use the first proposal distribution for every loop execution. Intuitively, this accounts for just training the inference network with the data that makes the loop end, effectively training the network to select proposal distributions such that the rejection loop is exited in as few iterations as possible.

This scheme requires annotating the simulator code to demarcate the parts where it should be applied, but this was greatly simplified as we were able to see exactly where to make the required annotations using the inspection tool. After the development of this new approach to rejection sampling, the average trace length dropped to 8.37, with the longest registered trace being of length 42. This made it possible to train the inference network and produce initial inference results for this particular physics problem.

Initial physics results: Using the setup described above, we were able to compute posterior distributions for p_x, p_y, p_z, `channel` for various simulated observations with known ground truth. The discrete variable `channel` has a known prior distribution given by the branching ratio of the τ into 38 possible decay channels [18]. While the SHERPA code has been instrumented and inference is technically possible, at the time of this writing, we have not yet prepared an efficient similarity kernel used in ABC. The choice of this kernel is critical for balancing the efficiency of the posterior sampling and the quality of the approximate inference. With our initial similarity kernel we achieve accuracies for the most frequent τ decay channels in the range of $\approx$60-90%. Therefore our further work aims to improve the similarity kernel used in ABC and improve the accuracy of the inference.

Future work: A common challenge for inference in particle physics is a large dynamic range of frequencies for various outcomes. For instance, some decay channels are much significantly more probable than the others, and the prior distribution for the momentum is often steeply falling. For this reason, the proposal parameters and the quality of the inference are better for observations with high likelihood. On the other hand, events with low likelihood (e.g., decay channels with small branching ratios) are less likely to be generated during training and the quality of the inference suffers. We

are investigating techniques for automatically adjusting the measure during training to favor these unlikely outcomes, a feature that we name as "prior inflation".

The graph describing the probabilistic structure of the simulation that is generated by our inspection tool (Figure 2, a) constitutes an interesting candidate for learning a structured proxy (or surrogate) for the generative model itself. We are considering using such learned proxies for speeding up training and inference in models where the execution of the simulator code takes a significant amount of time and resources, such as the simulation of particle detectors in GEANT.

In conclusion, this work provides an important first-step towards implementing a probabilistic programming approach in the physical sciences, which has considerable promise to leverage existing simulation software in order to provide tractable inference with a deeper level of interpretation than is possible with current analysis methods.

Acknowledgments

Lezcano Casado is supported by an Oxford-James Martin Graduate Scholarship and a La Caixa Postgraduate Fellowship. Baydin and Wood are supported under DARPA PPAML through the U.S. AFRL under Cooperative Agreement FA8750-14-2-0006, Sub Award number 61160290-111668. Martínez Rubio is supported by Intel BDC / LBNL Physics Graduate Studentship. Cranmer, Louppe, and Heinrich are supported through NSF ACI-1450310, PHY-1505463, PHY- 1205376, and the Moore-Sloan Data Science Environment at NYU. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

[1] G. Aad, T. Abajyan, B. Abbott, J. Abdallah, S. Abdel Khalek, A. A. Abdelalim, O. Abdinov, R. Aben, B. Abi, M. Abolins, and et al. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Physics Letters B*, 716: 1–29, September 2012. doi: 10.1016/j.physletb.2012.08.020.

[2] J. Allison, K. Amako, J. Apostolakis, P. Arce, M. Asai, T. Aso, E. Bagli, A. Bagulya, S. Banerjee, G. Barrand, B.R. Beck, A.G. Bogdanov, D. Brandt, J.M.C. Brown, H. Burkhardt, Ph. Canal, D. Cano-Ott, S. Chauvie, K. Cho, G.A.P. Cirrone, G. Cooperman, M.A. Cortés-Giraldo, G. Cosmo, G. Cuttone, G. Depaola, L. Desorgher, X. Dong, A. Dotti, V.D. Elvira, G. Folger, Z. Francis, A. Galoyan, L. Garnier, M. Gayer, K.L. Genser, V.M. Grichine, S. Guatelli, P. Guèye, P. Gumplinger, A.S. Howard, I. Hřivnáčová, S. Hwang, S. Incerti, A. Ivanchenko, V.N. Ivanchenko, F.W. Jones, S.Y. Jun, P. Kaitaniemi, N. Karakatsanis, M. Karamitros, M. Kelsey, A. Kimura, T. Koi, H. Kurashige, A. Lechner, S.B. Lee, F. Longo, M. Maire, D. Mancusi, A. Mantero, E. Mendoza, B. Morgan, K. Murakami, T. Nikitina, L. Pandola, P. Paprocki, J. Perl, I. Petrović, M.G. Pia, W. Pokorski, J.M. Quesada, M. Raine, M.A. Reis, A. Ribon, A. Ristić Fira, F. Romano, G. Russo, G. Santin, T. Sasaki, D. Sawkey, J.I. Shin, I.I. Strakovsky, A. Taborda, S. Tanaka, B. Tomé, T. Toshito, H.N. Tran, P.R. Truscott, L. Urban, V. Uzhinsky, J.M. Verbeke, M. Verderi, B.L. Wendt, H. Wenzel, D.H. Wright, D.M. Wright, T. Yamashita, J. Yarba, and H. Yoshida. Recent developments in GEANT4. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 835 (Supplement C):186 – 225, 2016. ISSN 0168-9002. doi: 10.1016/j.nima.2016.06.125.

[3] Mario Lezcano Casado. Compiled inference with probabilistic programming for large-scale scientific simulations. Master's thesis, University of Oxford, 2017.

[4] S. Chatrchyan, V. Khachatryan, A. M. Sirunyan, A. Tumasyan, W. Adam, E. Aguilo, T. Bergauer, M. Dragicevic, J. Erö, C. Fabjan, and et al. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Physics Letters B*, 716:30–61, September 2012. doi: 10.1016/j.physletb.2012.08.021.

[5] Kyle Cranmer, Juan Pavez, and Gilles Louppe. Approximating likelihood ratios with calibrated discriminative classifiers. *arXiv preprint arXiv:1506.02169*, 2015.

[6] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12(656-704):3, 2009.

[7] Ritabrata Dutta, Jukka Corander, Samuel Kaski, and Michael U Gutmann. Likelihood-free inference by penalised logistic regression. *arXiv preprint arXiv:1611.10242*, 2016.

[8] Eirik Endeve, Christian Y Cardall, Reuben D Budiardja, Samuel W Beck, Alborz Bejnood, Ross J Toedte, Anthony Mezzacappa, and John M Blondin. Turbulent magnetic field amplification from spiral SASI modes: implications for core-collapse supernovae and proto-neutron star magnetization. *The Astrophysical Journal*, 751(1):26, 2012.

[9] T. Gleisberg, Stefan. Hoeche, F. Krauss, M. Schonherr, S. Schumann, F. Siegert, and J. Winter. Event generation with SHERPA 1.1. *Journal of High Energy Physics*, 02:007, 2009. doi: 10.1088/1126-6708/2009/02/007.

[10] Andrew D Gordon, Thomas A Henzinger, Aditya V Nori, and Sriram K Rajamani. Probabilistic programming. In *Proceedings of the Future of Software Engineering*, pages 167–181. ACM, 2014.

[11] Florian Hartig, Justin M Calabrese, Björn Reineking, Thorsten Wiegand, and Andreas Huth. Statistical inference for stochastic simulation models–theory and application. *Ecology Letters*, 14(8):816–827, 2011.

[12] Alexander Heinecke, Alexander Breuer, Sebastian Rettenberger, Michael Bader, Alice-Agnes Gabriel, Christian Pelties, Arndt Bode, William Barth, Xiang-Ke Liao, Karthikeyan Vaidyanathan, et al. Petascale high order dynamic rupture earthquake simulations on heterogeneous supercomputers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 3–14. IEEE Press, 2014.

[13] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4743–4751. Curran Associates, Inc., 2016.

[14] R Kleiss, W James Stirling, and SD Ellis. A new Monte Carlo treatment of multiparticle phase space at high energies. *Computer Physics Communications*, 40(2-3):359–373, 1986.

[15] Ronald Kleiss and Roberto Pittau. Weight optimization in multichannel monte carlo. *Computer Physics Communications*, 83(2-3):141–146, 1994.

[16] Tuan Anh Le, Atılım Güneş Baydin, and Frank Wood. Inference compilation and universal probabilistic programming. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54 of *Proceedings of Machine Learning Research*, pages 1338–1348, Fort Lauderdale, FL, USA, 2017. PMLR.

[17] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, 2017.

[18] C Patrignani, DH Weinberg, CL Woody, RS Chivukula, O Buchmueller, Yu V Kuyanov, E Blucher, S Willocq, A Höcker, C Lippmann, et al. Review of particle physics. *Chinese Physics C*, 40:100001, 2016.

[19] Paris Perdikaris, Leopold Grinberg, and George Em Karniadakis. Multiscale modeling and simulation of brain blood flow. *Physics of Fluids*, 28(2):021304, 2016.

[20] Marco Raberto, Silvano Cincotti, Sergio M. Focardi, and Michele Marchesi. Agent-based simulation of a financial market. *Physica A: Statistical Mechanics and its Applications*, 299 (1):319 – 327, 2001. ISSN 0378-4371. doi: 10.1016/S0378-4371(01)00312-0. Application of Physics in Economic Modelling.

[21] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[22] Mikael Sunnåker, Alberto Giovanni Busetto, Elina Numminen, Jukka Corander, Matthieu Foll, and Christophe Dessimoz. Approximate Bayesian computation. *PLoS Computational Biology*, 9(1):e1002803, 2013.

[23] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *Journal of Machine Learning Research*, 17(205):1–37, 2016.

[24] Richard David Wilkinson. Approximate Bayesian computation (ABC) gives exact results under the assumption of model error. *Statistical Applications in Genetics and Molecular Biology*, 12 (2):129–141.

[25] David Wingate, Andreas Stuhlmueller, and Noah Goodman. Lightweight implementations of probabilistic programming languages via transformational compilation. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 770–778, 2011.