Bayesian optimization of PYTHIA8 tunes

Ali Al Kadhim[*] and Harrison B. Prosper[†]
Department of Physics, Florida State University, Tallahassee, Florida, USA

Stephen Mrenna[‡]
Computational Science and AI Directorate, Fermilab, Batavia, Illinois, USA

(Received 21 May 2025; accepted 11 August 2025; published 11 September 2025)

A new tune (set of model parameters) is found for the six most important parameters of the PYTHIA8 final state parton shower and hadronization model using Bayesian optimization. The tune fits the Large Electron-Positron collider (LEPI) data from ALEPH better than the default tune in PYTHIA8. To the best of our knowledge, we present the most comprehensive application of Bayesian optimization to the tuning of a parton shower and hadronization model using the LEPI data.

DOI: 10.1103/1hn2-hz4q

I. INTRODUCTION AND MOTIVATION

An understanding of confinement (for which there is a one million dollar prize from the Clay Institute[1]) requires the identification of the mechanism by which colored partons become color singlet hadrons. This phenomenon is currently described by models that are inspired by the strong interaction, but are not derived from first principles.

Motivated by evidence from QCD factorization theorems [1], it is assumed that hadronization is a universal process that can be factorized from the perturbative calculations. If this is true, the same hadronization model can be used both in proton-proton collisions as well as electron-positron collisions: a model fit to Large Electron-Positron collider (LEPI) electron-positron collision data can be applied to simulations at the LHC. An advantage of studying hadronization using LEPI data is the absence of phenomena associated with the breakup of the colliding beams.

LEPI provided a nearly perfect environment to study hadronization effects [2], one characterized by a well-defined initial e^+e^- state and a high event rate at the Z mass [3]. Furthermore, a large improvement in the quality of the data and a reduction of systematic errors was achieved at LEPI with respect to what was achieved in the era of PETRA and PEP [4]. Finally, using one dataset from one experiment avoids the problem of having to correct for correlations that exist between different experiments and datasets.

Currently, two phenomenological models of hadronization[2] are widely studied: cluster hadronization [5] and string fragmentation [6]. The Lund string model [6] (summarized in Sec. III) is the basis for the hadronization model in PYTHIA8. This model will be the subject of the current study. The task of studying the Lund model seems daunting, since it requires expensive simulations with many possible different model parameters. The development of efficient tools to support the tuning of these parameters is a high priority for such studies.

The choice of experimental data used to tune the Lund string model parameters is an important consideration. The experimental data should cover a wide range of physics so as not to miss the full implications of the model. Surveys have been conducted that conclude that model parameters are best determined by fitting the model to inclusive distributions, jet rates, and shape distributions simultaneously [3]. For these reasons, we chose to use the LEPI data for QCD observables collected by the ALEPH detector, published in Ref. [2], for the tuning of the Lund hadronization parameters. The data offer a rich set of observables to tune these parameters and comprise a collection of $\mathcal{O}(30)$ event-shape variables such as thrust, sphericity, aplanarity, and inclusive distributions such as p_T^{in}, p_T^{out}, and the rapidity with respect to the thrust axis. See Ref. [2] for descriptions of these observables.

The investigations reported in this paper have two goals:

(1) Investigate the use of Bayesian optimization for tuning PYTHIA8 parameters by minimizing an

[*]Contact author: aalkadhim@fsu.edu
[†]Contact author: hprosper@fsu.edu
[‡]Contact author: mrenna@fnal.gov

[1]https://www.claymath.org/millennium/yang-mills-the-maths-gap/

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI. Funded by SCOAP[3].

[2]"Hadronization" and "fragmentation" can be used almost interchangeably.

objective function. In principle, Bayesian optimization is ideally suited for such a problem because of the high computational cost of evaluating the objective function, which requires simulating a large number of events at each tune parameter point.

(2) Assess whether Bayesian-optimized tuning converges to the same parameter point as obtained with methods such as PROFESSOR [7] or APPRENTICE [8].

The remainder of this paper is organized as follows. In Sec. II, we briefly review tuning methods and how our approach differs from them. In Sec. III, the six PYTHIA8 parameters, which are the focus of the tuning, are reviewed. Section IV briefly describes the LEPI event datasets employed and the objective function used in the tuning. Section V describes implementation details of the Bayesian optimization framework. The results of the Bayesian optimization of the six tune parameters are presented in Sec. VI and compared with the tune results from PROFESSOR/APPRENTICE-based fits. Finally, Sec. VII provides our conclusions. The interested reader can follow the directions in Appendix A to reproduce all results presented in this paper.

II. MONTE CARLO TUNE METHODS

A. PROFESSOR and PROFESSOR-like tunes

PROFESSOR [7] is the main event generator tuning method used by CMS, ATLAS, and other large collaborations. Recently, new tools for event generator tuning have become available, including APPRENTICE [8], AUTOTUNES [9], and machine learning methods such as MLHAD [10,11] and HAD-ML [12]. While PROFESSOR (and PROFESSOR-like tuning methods such as APPRENTICE) have produced excellent results, there is still room for improvement:

(i) the estimated uncertainties are based on the Hessian inverse; consequently, there is no guarantee that the confidence sets have valid coverage;

(ii) the polynomial fit to the Monte Carlo–(MC) based predictions is adequate for well-behaved distributions, but may fail for more singular distributions (though see [13] for a rational polynomial fitter available in APPRENTICE); and

(iii) the choice of the weights in these methods is done manually (though see [14] for algorithms to select these weights).

B. CMS CUET tunes and multiparton interaction considerations

CMS has determined tunes, dubbed CUET [15], which focus on the modeling of multiparton interactions (MPIs) in event generators. Multiparton interactions are $2 \to 2$ parton scatterings that occur within the same hadron-hadron collision, but are softer than the hard scattering ($p_T < 3$ GeV). Like the hadronization models, MPI models are also phenomenological with free parameters determined by fits to data. CMS has recently evolved beyond this and now uses the CPX family of tunes that use next-to-leading logarithmic α_s and next-to-next-to-leading-order parton distribution functions [16].

C. Tuning with Bayesian optimization

The tuning of an event generator involves generating a sufficient number of events at any given tune point, which is computationally costly. Optimization of a costly objective function is the kind of problem for which Bayesian optimization [17] is, in principle, well suited. Bayesian optimization (BayesOpt) has been attempted before [18] to tune three PYTHIA8 parameters (`TimeShower:alphaSvalue`, `TimeShower:pTmin`, and `TimeShower:pTminChgQ`) using a subset of the data used to establish Monash. However, this paper presents a more thorough investigation of BayesOpt and its application to the tuning of PYTHIA8. Since BayesOpt remains unfamiliar to many particle physicists, Sec. V presents the necessary background before the tune results obtained for PYTHIA8 are presented in Sec. VI.

III. LUND FRAGMENTATION MODEL PARAMETERS

The central assumption of the Lund model is that a quark-antiquark pair is subject to the QCD potential $V_{\mathrm{QCD}}(r) \approx -\frac{4}{3}\frac{\alpha_s}{r} + \kappa r$, where r is the distance between the quark and the antiquark and α_s is the strong coupling parameter. For an energetic quark and antiquark pair traveling in opposite directions, the linear term is most relevant. The Lund model views the confining force between colored charges as a flux tube or a string with string tension κ. The parameter $\kappa \approx 1$ GeV/fm was first inferred from Regge trajectories [19] and was later confirmed by lattice QCD calculations [20]. In the model, the string breaks through the quantum mechanical tunneling of quark-antiquark or diquark-antidiquark pairs out of the vacuum.

The Lund fragmentation function

$$f_{\mathrm{Lund}}(z) \propto \frac{(1-z)^a}{z}\exp\left(-\frac{bm_T^2}{z}\right) \tag{1}$$

is the probability density that a hadron h with transverse mass

$$m_T^2 = m^2 + p_T^2 \tag{2}$$

is produced with light cone momentum fraction z [21], with the remainder, $1-z$, taken by the string. A large value of the a parameter suppresses the hard region $z \to 1$, while a low value suppresses soft regions $z \to 0$. The a and b parameters are implemented in PYTHIA8 as `StringZ:aLund` and `StringZ:bLund`, respectively. The choice of `StringZ:aLund` and `StringZ:bLund` strongly impacts QCD observables (see, for example, Fig. 1).

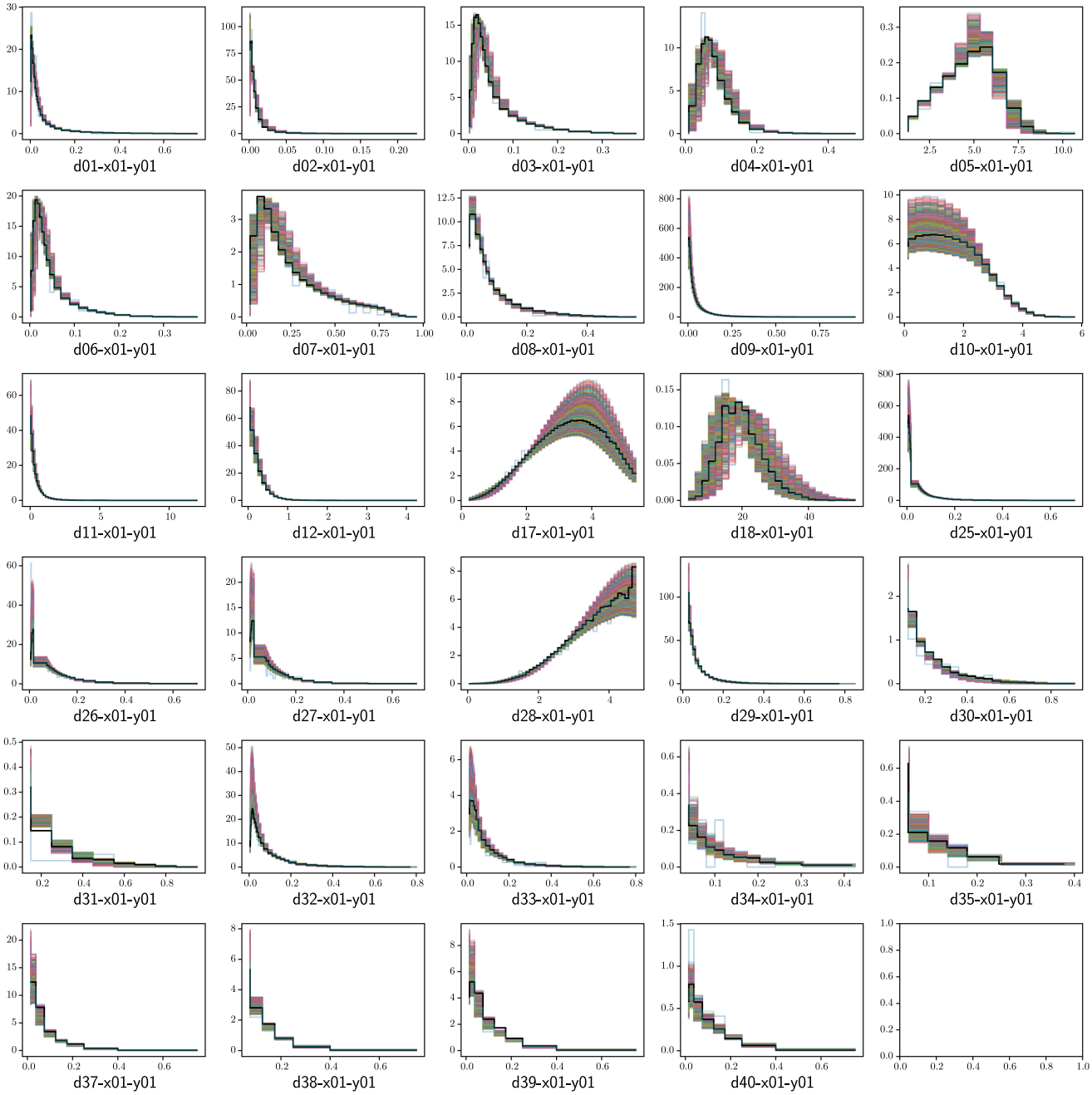


FIG. 1. Impact of randomly sampling aLund and bLund (keeping the other parameters fixed to Monash) and the associated effect it has on the ALEPH distributions. Black lines show the experimental ALEPH data; colored curves correspond to model predictions for each random parameter draw.

While the b parameter is universal, the a parameter does not need to be the same for all flavors of quarks or diquarks. The more general parametrization of fragmentation from parton i to parton j is

$$f_{\text{Lund}}(z) \propto \frac{1}{z^{1+r_Q b m_Q^2}} z^{\delta a_i - \delta a_j}(1-z)^{\delta a_j} \times \exp\left(-\frac{b m_T^2}{z}\right), \tag{3}$$

where we have included the Bowler modification for massive end point quarks with mass m_Q: $r_Q = 0$ unless $Q = c$ or $Q = b$. This introduces two new parameters `StringZ:rFactC` and `StringZ:rFactB` for the b and c quarks, respectively. The parameters δa_i are only implemented for strange quarks and diquarks and are denoted by `StringZ:aExtraDiquark` and `StringZ:aExtraSQuark`, respectively.

The tunneling of the partons from the vacuum imparts p_T to the produced hadrons: this is modeled using a Gaussian

TABLE I. Selected PYTHIA8 tune parameters. The parameters are optimized over the ranges shown in the second column. The last column lists the Monash tune in PYTHIA8, which is the default tune.

Name in PYTHIA8	Tuning range	Monash
`StringZ:aLund`	[0.0, 2.0]	0.68
`StringZ:bLund`	[0.2, 2.0]	0.98
`StringFlav:ProbStoUD`	[0.0, 1.0]	0.217
`StringFlav:probQQtoQ`	[0.0, 1.0]	0.081
`TimeShower:alphaSvalue`	[0.06, 0.25]	0.1365
`TimeShower:pTmin`	[0.1, 2.0]	0.5

distribution in p_x and p_y,

$$P(p_x, p_y, \sigma_T) = \frac{1}{2\pi\sigma_T^2} \exp\left[-(p_x^2 + p_y^2)/(2\sigma_T^2)\right], \quad (4)$$

where the standard deviation σ_T is a free parameter denoted by `StringPT:sigma` in PYTHIA8.

The fragmentation model also accounts for strangeness suppression, that is, the creation of $s\bar{s}$ relative to $u\bar{u}$ and $d\bar{d}$ and is configured with the PYTHIA8 parameter `StringFlav:probStoUD`. Also included in the tune is `StringFlav:probQQtoQ`, the parameter that controls the degree of suppression of diquark production relative to quark production.

In PYTHIA8 the amount of QCD radiation in the parton shower depends strongly on the choice of $\alpha_S(M_Z^2)$, which is denoted by `TimeShower:alphaSvalue`. The default value for this parameter was determined by a relatively crude tuning to LEPI data, which is why we choose to tune it. Finally, since the radiation rate diverges as $p_T \to 0$, a regime in which perturbation theory is invalid, a cutoff is introduced, `TimeShower:pTmin`, below which no QCD emissions are allowed. This value is usually chosen to be close to Λ_{QCD} in order to have a smooth transition between low-p_T perturbative emissions and nonperturbative string fragmentation. The parameters tuned in this study are listed in Table I. Obviously, more parameters could be included, but we have identified a few critical ones for the purpose of studying the Bayesian optimization methodology.

IV. LEPI DATA AND EVENT SIMULATION

In this study, LEPI electron-positron collision data, which are provided as histograms, are used to tune the parameters in Table I. The ALEPH histograms used and their labels are listed in Table V. The datas in the ith bin include a measured cross section σ_i and its associated uncertainty $\delta\sigma_{\mathrm{data},i}$. Ideally, the statistical uncertainty $\delta\sigma_{\mathrm{MC},i}$ in the associated cross section prediction $\sigma_{\mathrm{MC},i}$ should $\ll \delta\sigma_{\mathrm{data},i}$ so that $\delta\sigma_{\mathrm{MC},i}$ can be neglected. In the ALEPH histograms the effective count, $n \approx \sum_i (\sigma_i/\delta\sigma_i)^2$, varies from ∼330 to ∼950,000. Therefore, in principle, at any given point in the tune space we should simulate $\gg 10^6$ events. However, for our current studies, 250,000 events per tune point are simulated, which yields predictions with a statistical precision that roughly match the overall precision of the data. The statistical uncertainties in the predictions are accounted for by adding $\delta\sigma_{\mathrm{data},i}$ and $\delta\sigma_{\mathrm{MC},i}$ in quadrature to yield a cross section standard deviation $\delta\sigma_i$. To guard against the danger that a single bin dominates the objective function, bins with standard deviations less than 10^{-3} pb have their standard deviations set to 1 pb. Finally, histograms where the total cross section is < 0.1 pb are omitted.

Since the correlations between histograms are not available, the objective function $O(\mathbf{x})$ is taken to be a linear sum of χ^2-like variates,

$$O(\mathbf{x}) = \sqrt{\frac{1}{N} \sum_{\text{histograms}} \sum_{i=1}^{N_{\text{bins}}} \left(\frac{D_i - T_i(\mathbf{x})}{\delta\sigma_i}\right)^2}, \quad (5)$$

where $N = \sum_{\text{histograms}} N_{\text{bins}}$ is the total number of bins across all histograms, D_i is the measured cross section in bin i, and T_i is the theoretical cross section prediction obtained via Monte Carlo simulation with PYTHIA8 for a given tune point $\mathbf{x}$. The normalization by the total number of bins and the square root in Eq. (5) reduces the dynamic range of $O(\mathbf{x})$. However, in our studies reducing the dynamic range of the objective function does not play a significant role.

V. TUNING WITH BAYESIAN OPTIMIZATION

The Bayesian optimization frameworks and Gaussian process surrogate models described below are standard methods in the machine learning literature and are not original contributions of this work. The exposition below is provided for the benefit and convenience of readers who may be unfamiliar with this method of optimization. In particular, we follow the formulation of Bayesian optimization described in [22,23] and the Gaussian process regression formalism in [24].

Given an objective function $O: \mathcal{X} \to \mathbb{R}$ defined on a compact domain $\mathcal{X} \subseteq \mathbb{R}^d$, the goal of Bayesian optimization is to search the domain as efficiently as possible for a good approximation to the location of the global minimum

$$\mathbf{x}_0 = \arg\min_{\mathbf{x} \in \mathcal{X}} O(\mathbf{x}). \quad (6)$$

Evaluating $O(\mathbf{x})$, however, is expensive, because it requires simulating a large number of events on the fly. Moreover, one does not have a closed form for the dependence of $O(\mathbf{x})$ on $\mathbf{x}$. However, since the objective function can be computed at any given point, it is possible to obtain "observations" of the function at a set of sampled points. Bayesian optimization reduces a difficult, expensive

optimization problem into a sequence of less costly optimization problems in which a model of $O(\mathbf{x})$ is optimized. BayesOpt (also known as "surrogate-based optimization" [25]) is especially useful for optimizing objective functions $O(\mathbf{x})$ for which there is no efficient method for estimating their gradients and/or if $O(\mathbf{x})$ is nonconvex. The only assumption about $O(\mathbf{x})$ in BayesOpt is that the objective is Lipschitz continuous; that is, there exists some constant C, such that, for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$,

$$\|O(\mathbf{x}_1) - O(\mathbf{x}_2)\| \leq C\|\mathbf{x}_1 - \mathbf{x}_2\|. \tag{7}$$

In principle, any sufficiently flexible function class can be used for the surrogate model, but, in practice, one uses Gaussian processes (GPs) [24] because of their mathematical convenience. Given the data $\mathcal{D} = \{X, Y\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $y_i = O(\mathbf{x}_i)$ is the observed value of the objective function at the tune parameter point $\mathbf{x}_i$, one first computes the posterior density $p(f|\mathcal{D})$ over surrogate models $f(\mathbf{x})$ of the objective function $O(\mathbf{x})$,

$$\begin{aligned} p(f|\mathcal{D}) &= \frac{p(\mathcal{D}|f)\pi(f)}{p(\mathcal{D})}, \\ &= \frac{p(Y|\mathbf{X}, f)p(\mathbf{X}|f)\pi(f)}{p(Y|\mathbf{X})p(\mathbf{X})}, \\ &= \frac{p(Y|\mathbf{X}, f)\pi(f)}{p(Y|\mathbf{X})}, \end{aligned} \tag{8}$$

where $\pi(f)$ is a prior over functions modeled with a GP and where we have assumed that $p(\mathbf{X}|f) = p(\mathbf{X})$. The quantity $p(Y|\mathbf{X}, f)$ is the likelihood function for the data $\mathcal{D}$. Next, the posterior density together with another function, called the "acquisition function", is used to decide where the next observation of the objective function $O(\mathbf{x})$ should be made such that one progressively arrives at a good estimate of the location $\mathbf{x}_0$ of the global minimum.

In the following, we consider how each of the three terms in Eq. (8) is computed and follow with a description of the BayesOpt algorithm noting which hyperparameters are most relevant for deriving tunes.

A. Gaussian process prior $\pi(f)$

A Gaussian process in $f(\mathbf{x})$,

$$\mathcal{GP}(f; \mu, \mathbf{K}), \tag{9}$$

is a set of random variables, any countable set of which $f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots$ follow a multivariate normal distribution, where μ is the "mean function" given by

$$\mu(\mathbf{x}) = \mathbb{E}_{f\sim\mathcal{GP}}[f(\mathbf{x})|\mathbf{x}], \tag{10}$$

and $\mathbf{K}$ is the "covariance function" (or kernel or kernel function) whose matrix elements are given by

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= \mathrm{cov}[f(\mathbf{x}), f(\mathbf{x}')|\mathbf{x}, \mathbf{x}'] \\ &= \mathbb{E}_{f\sim\mathcal{GP}}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))]. \end{aligned} \tag{11}$$

The mean function for the prior is chosen to be $\mu(\mathbf{x}) = 0$, that is,

$$\pi(f) = \mathcal{GP}(f; 0, \mathbf{K}). \tag{12}$$

For the matrix elements of the covariance function, we choose the automatic relevance determination Matern kernel [26]

$$\begin{aligned} &k(\mathbf{x}_i, \mathbf{x}_j) = C\frac{1}{2^{\nu-1}\Gamma(\nu)}(2\sqrt{\nu}d)^\nu H_\nu(2\sqrt{\nu}d), \\ &\text{with} \quad d = (\mathbf{x_i} - \mathbf{x_j})^T S^{-1}(\mathbf{x_i} - \mathbf{x_j}), \end{aligned} \tag{13}$$

where S^{-1} is a diagonal matrix composed of the length scale parameter for each dimension, C is an overall scale factor, and $\Gamma(\cdot)$ and $H_\nu(\cdot)$ are the gamma function and Bessel function of order ν, respectively. The parameter ν—which determines the smoothness of the functions $f(\mathbf{x})$—is set to $\nu = 5/2$, which has been singled out as a good choice.

B. Likelihood function $p(Y|\mathbf{X}, f)$

Suppose we have observed (that is, evaluated) the objective function $y = O(\mathbf{x})$ at parameter points $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_n$. For the Bayesian inference to proceed, we need an expression for the likelihood function of the observations, $Y = y_1, \ldots, y_n$, where $y_i = O(\mathbf{x}_i)$. A typical assumption is that

$$y_i = f(\mathbf{x}_i) + \varepsilon, \tag{14}$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2_{\mathrm{noise}})$ is Gaussian noise, yielding a diagonal multivariate Gaussian likelihood

$$p(Y|\mathbf{X}, f) = \prod_{i=1}^n p(y_i|\mathbf{x}_i, f(\mathbf{x}_i)). \tag{15}$$

This choice for the likelihood is particularly convenient because it yields a posterior density that is a Gaussian process with its mean and covariance functions available in closed form, as noted in the next section.

C. Posterior density $p(f|\mathcal{D})$

Consider the set of observations Y at the points $\mathbf{X}$ and another set of points $\mathbf{X}^* = \mathbf{x}_1^*, \ldots, \mathbf{x}_m^*$ at which we wish to estimate the objective function. A Gaussian process prior and a Gaussian likelihood yield a posterior density that is a Gaussian process with mean and covariance functions given by

$$\mu(\mathbf{x}^*|\mathcal{D}) = \mu(\mathbf{x}^*) + \mathbf{K}_{\mathbf{X}\mathbf{x}^*}^T \mathbf{K}_{\mathbf{X}\mathbf{X}}^{-1}[Y - \mu(\mathbf{X})] \quad (16)$$

and

$$k(\mathbf{x}^*, \mathbf{x}^{*\prime}|\mathcal{D}) = k(\mathbf{x}^*, \mathbf{x}^{*\prime}) - \mathbf{K}_{\mathbf{X}\mathbf{x}^*}^T \mathbf{K}_{\mathbf{X}\mathbf{X}}^{-1} \mathbf{K}_{\mathbf{X}\mathbf{x}^*}, \quad (17)$$

respectively, where the prior mean $\mu(\mathbf{x}) = 0$ [see Eq. (12)]. The elements of the matrices $\mathbf{K}_{\mathbf{X}\mathbf{x}^*}$ and $\mathbf{K}_{\mathbf{X}\mathbf{X}}$ are given by $k(\mathbf{x}_i, \mathbf{x}^*)$ and $k(\mathbf{x}_i, \mathbf{x}_j') + \sigma_{\text{noise}}^2 \delta_{ij}$, $i, j = 1, \ldots, n$, respectively, where the kernel is given by Eq. (13) and the likelihood by Eq. (15). As more observations are added to the existing set, the mean and covariance functions are updated recursively according to Eqs. (16) and (17).

As in all Bayesian calculations, the posterior density completes the inference. However, one typically computes a few useful functionals of the posterior density including the predictive distribution

$$p(y|\mathbf{x}^*, \mathcal{D}) = \int p(y|\mathbf{x}^*, f) p(f|\mathcal{D}) df, \quad (18)$$

which in this context is the probability density that an observation of the objective function at the new point $\mathbf{x}^*$ yields the value y. The mean of the predictive distribution is often taken to be an estimate of the objective function at $\mathbf{x}^*$, while the width of the distribution is a measure of the uncertainty associated with that estimate.

D. Acquisition functions

Another important quantity that can be computed from the posterior density is the expected utility $\mathbb{E}[u]$ of moving to the point $\mathbf{x}^*$ with the goal of moving along an efficiently selected sequence of points to the minimum of the objective function. The expected utility in this context is called an acquisition function of which several were considered in this study. The "expected improvement" acquisition function (α_{EI}) is defined by the utility function

$$u(f^*, y^+) = \max(y^+ - \xi - f^*, 0), \quad (19)$$

where f^* is the value of the surrogate function at point $\mathbf{x}^*$ and $y^+ = O(\mathbf{x}^+)$ is the lowest value of the objective function value found so far. The hyperparameter ξ controls the degree to which the expected improvement favors exploration versus exploitation. Large values of ξ penalize exploitation in favor of exploration. For a GP, the expected improvement can also be obtained in closed form [27] and is given by

$$\alpha_{\text{EI}}(\mathbf{x}) = \begin{cases} (y^+ - \mu(\mathbf{x}) - \xi)\Phi\left(\frac{y^+ - \mu(\mathbf{x}) - \xi}{\sigma(\mathbf{x})}\right) + \sigma(\mathbf{x})\phi\left(\frac{y^+ - \mu(\mathbf{x}) - \xi}{\sigma(\mathbf{x})}\right), & \sigma(\mathbf{x}) > 0, \\ 0, & \sigma(\mathbf{x}) = 0, \end{cases} \quad (20)$$

where Φ is the cumulative distribution function of the standard normal whose density is denoted by $\phi(\cdot)$, and $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are, respectively, the posterior mean and standard deviation of the GP at $\mathbf{x}$. In Ref. [23], it is found that $\xi = 0.01$ gives the best results in all settings, while a schedule for ξ that encourages exploration early and exploitation later does *not* work in practice, contrary to intuition. We therefore keep $\xi = 0.01$ constant throughout our studies.

The next point at which to make an observation of the objective function is found by maximizing the acquisition function

$$\mathbf{x}^* = \arg\max_{\mathbf{x}^* \in \mathcal{X}} \alpha(\mathbf{x}^*). \quad (21)$$

E. Optimizing the kernel and likelihood hyperparameters

The likelihood (15) and the prior (12) both contain hyperparameters, denoted collectively by $\phi = C, S^{-1}, \sigma_{\text{noise}}$, which need to be optimized. In a purely Bayesian approach, the hyperparameters would be eliminated from the problem through marginalization with respect to a hyperprior. However, it is simpler to use an empirical Bayes [28] approach and optimize the hyperparameters by maximizing

$$p(Y|\mathbf{X}, \phi) = \int p(Y|\mathbf{X}, f)\pi(f) df, \quad (22)$$

that is, the prior predictive distribution with respect to ϕ,

$$\hat{\phi} = \arg\max_{\phi} p(Y|\mathbf{X}, \phi). \quad (23)$$

The data, $\mathcal{D}_{\text{Sobol}} = \{Y, \mathbf{X}\}$, for the likelihood are observations Y of the objective function at a Sobol [29] sequence of points $\mathbf{X}$ that provide a quasirandom sampling of the tune space. Equation (23) is implemented using the Adam [30] optimizer.

F. BayesOpt algorithm

Algorithm 1 shows the key steps of the BayesOpt algorithm and has its own set of hyperparameters or choices that need to be optimized to obtain satisfactory algorithmic behavior.

These hyperparameters are in addition to those of the kernel and likelihood function. The following were found to be the key BayesOpt hyperparameters to be optimized and the key choices to be made:

(i) number of BayesOpt iterations, N_{BayesOpt};
(ii) number of Sobol points N_{Sobol};

Algorithm 1. Bayesian optimization with expected improvement.

Require:
(a) Number of BayesOpt iterations: N_{BayesOpt}.
(b) Number of Sobol points: N_{Sobol}.
(c) Number of restarts: N_{restarts}.
Ensure:
Estimate $\hat{\mathbf{x}}$ of global minimum $\mathbf{x}_0$
Initialize: $\mathcal{D} \leftarrow \emptyset$
Generate Sobol points $\{\mathbf{x}_i\}_{i=1}^{N_{\text{Sobol}}} \subset \mathcal{X}$
Observe $y_i = O(\mathbf{x}_i)$
$\mathcal{D} \leftarrow \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$
Optimize hyperparameters ϕ:
$\phi^* = \arg\max_\phi p(Y|\mathbf{X}, \phi)$
for $t = 1, \ldots, N_{\text{BayesOpt}}$ **do**
  Compute posterior functions: $\mu(\mathbf{x}^*), k(x^*, x^{*'})$
  Compute expected improvement $\alpha_{\mathbf{EI}}(x^*)$
  Select next point:
  $\mathbf{x}^* = \arg\max_{\mathbf{x}^* \in \mathcal{X}} \alpha_{\text{EI}}(\mathbf{x}^*)$
  Observe: $y = O(\mathbf{x}^*)$
  Augment data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}, y)\}$
end for
return $\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \mathcal{D}$

(iii) optimization algorithm for the acquisition function; and
(iv) the number of restarts in the optimization of the acquisition function, N_{restarts}.

In order to cross-check our results, two different toolkits—GPYTORCH [31] and BOTORCH [32]—are employed to carry out the Bayesian optimization studies, in three stages. In stage 1, an *ab initio* implementation of BayesOpt was developed using GPYTORCH with its hyperparameters optimized on toy objective functions defined on the PYTHIA8 tune parameter space. In stage 2, the BayesOpt was applied to PYTHIA8 using the choices and hyperparameter settings established in stage 1 to obtain a PYTHIA8 tune. In stage 3, a PYTHIA8 tune was obtained using the built-in BayesOpt routines of BOTORCH. The BOTORCH implementation used a different acquisition function from that used in our GPYTORCH implementation, namely, qNoisyExpectedImprovement [32].

VI. RESULTS

The selected tune parameters, motivated in Sec. III, are optimized over the ranges shown in Table I, which are the allowable ranges in PYTHIA8. The last column in the table gives the default PYTHIA8 tune.

Before comparing our objective function values to those of Monash [33], it is important to note that the Monash tune is not the result of a formal χ^2 minimization, but rather the result of a manual procedure based on physically motivated parameter choices and constraints, notably for the infrared shower cutoff $p_{T,\text{min}}$ (`TimeShower:pTmin`) and the inclusion of theoretical uncertainties to help guard against overfitting.

Table II shows the results of five BayesOpt runs using our GPYTORCH implementation, where each run uses different choices of the BayesOpt hyperparameters N_{BayesOpt}, N_{Sobol}, and N_{restarts}. The result of one these fits, BO-EI4, is shown in Fig. 2. By definition, the optimal tune is the one with the lowest value of the objective function $O(\mathbf{x})$, which is seen to arise from run BO-EI5. However, Table II shows that the Monash point gives a better fit, as indicated by its lower objective function value. We also note that the BayesOpt reaches a tune that differs from the Monash tune.

Tables III and IV show details of two different BOTORCH runs, where each row apart from the one labeled Monash corresponds to a different step in the Bayesian optimization. We find that the two runs arrive at essentially the same tune. Furthermore, this point is close to the tune obtained using GPYTORCH, suggesting that the latter would converge to the same point had the algorithm been allowed to run for longer. Interestingly, the value of the objective function of the tune obtained with BOTORCH is *lower* than that of the Monash point; that is, we obtain a tune that fits the ALEPH data better than the Monash tune. The result of the best BOTORCH fit (BO-EI29 in Table III) is shown in Fig. 3.

However, there are some caveats on the physics implications of the results. The properties of the parton shower are set mainly by the value $\alpha_s(M_Z)$ and the shower cutoff $p_{T,\text{min}}$. A large high-scale value for α_s leads also to large values at lower scales, suppressing the no radiation probability. On the other hand, a large value of $p_{T,\text{min}}$ means that small values of hadron p_T must come from some other mechanism. The Lund model provides both longitudinal and transverse momentum to hadrons and can serve as this mechanism, but only for a certain range of parameters. For the large values of a_{Lund}, b_{Lund} appearing in these tunes, the average z values of the string splittings are higher than those for Monash. On the other hand, the average z values are constrained by the average particle multiplicity, leading to a tightly constrained system. In other words, the properties of hadrons depend upon several models, but there are no clear boundaries between those models. If the parton shower fails to produce enough transverse momentum, the hadronization model will be forced to compensate. This could lead to a choice of parameters for the parton shower and the hadronization model that appear to be extreme, but lead to a reasonable fit to the data.

Another caveat is that the predictions are computed with a statistical precision comparable to that of the ALEPH data. One might be tempted to interpret the likelihood hyperparameter σ_{noise} as a measure of the noisiness of the objective function. For our GPYTORCH results, we find $\sigma_{\text{noise}} = 1.6$. However, this value should be interpreted with caution. A direct way to estimate the noise in the objective function is to repeatedly simulate many sets of 250,000 events at a given tune point $\mathbf{x}$ and compute the standard deviation of $O(\mathbf{x})$. For the Monash point, the mean of $O(\mathbf{x})$ is 3.19 and its standard deviation is 0.014. This suggests

TABLE II. GPYTORCH results using all of the ALEPH histograms and Eq. (5). Each row shows the results for a BayesOpt run with the indicated hyperparameter values for N_{BayesOpt}, N_{Sobol}, and N_{restart}.

Label	N_{BayesOpt}	N_{Sobol}	N_{restarts}	$O(\hat{\mathbf{x}})$	aLund	bLund	ProbStoUD	probQQtoQ	alphaSvalue	pTmin
Monash				3.2083	0.68	0.98	0.217	0.081	0.1365	0.500
BO-EI1	80	25	25	7.763	1.388	1.458	0.019	0.048	0.133	0.260
BO-EI2	90	225	200	4.082	0.741	0.646	0.232	0.077	0.133	0.344
BO-EI3	100	75	25	6.446	1.050	0.592	0.401	0.069	0.128	1.887
BO-EI4	120	300	200	4.174	1.260	1.858	0.325	0.060	0.137	0.138
BO-EI5	180	300	5000	3.789	1.991	1.873	0.252	0.084	0.135	1.207

TABLE III. Results of a single BOTORCH run using all of the ALEPH histograms with $N_{\text{BayesOpt}} = 100$, $N_{\text{Sobol}} = 25$. The first row shows the Monash point and the optimal tune found with BOTORCH is in bold.

Label	N_{BayesOpt}	N_{Sobol}	$O(\hat{\mathbf{x}})$	aLund	bLund	ProbStoUD	probQQtoQ	alphaSvalue	pTmin
Monash			3.2083	0.68	0.98	0.217	0.081	0.1365	0.500
BO-qEI1	100	25	3.185	1.807	1.885	0.228	0.087	0.141	1.170
BO-qEI2	100	25	3.179	1.777	1.834	0.198	0.094	0.141	1.001
BO-qEI3	100	25	3.156	1.793	1.837	0.196	0.085	0.139	1.138
BO-qEI4	100	25	3.153	1.904	1.867	0.209	0.094	0.138	0.999
BO-qEI5	100	25	3.152	2.000	1.841	0.204	0.085	0.143	1.326
BO-qEI6	100	25	3.145	2.000	1.813	0.200	0.072	0.136	1.118
BO-qEI7	100	25	3.139	1.825	1.935	0.203	0.087	0.139	0.987
BO-qEI8	100	25	3.131	2.000	1.938	0.209	0.083	0.135	0.748
BO-qEI9	100	25	3.130	1.842	1.853	0.221	0.090	0.138	0.845
BO-qEI10	100	25	3.115	1.815	1.840	0.198	0.081	0.136	0.916
BO-qEI11	100	25	3.102	2.000	1.870	0.220	0.079	0.136	0.933
BO-qEI12	100	25	3.095	1.893	1.867	0.221	0.078	0.138	0.917
BO-qEI13	100	25	3.073	1.984	1.935	0.217	0.087	0.138	0.938
BO-qEI14	100	25	3.054	1.937	1.875	0.196	0.080	0.138	1.060
BO-qEI15	100	25	3.050	2.000	1.640	0.207	0.082	0.142	1.649
BO-qEI16	100	25	3.049	2.000	1.594	0.215	0.078	0.137	1.398
BO-qEI17	100	25	3.042	1.945	1.888	0.211	0.076	0.139	1.183
BO-qEI18	100	25	3.034	1.875	1.588	0.176	0.085	0.139	1.465
BO-qEI19	100	25	3.033	2.000	1.837	0.232	0.075	0.139	1.235
BO-qEI20	100	25	3.029	1.862	1.895	0.210	0.077	0.137	0.986
BO-qEI21	100	25	3.023	1.939	1.881	0.225	0.085	0.140	1.139
BO-qEI22	100	25	3.007	1.818	1.807	0.204	0.076	0.138	1.075
BO-qEI23	100	25	3.006	1.901	1.768	0.197	0.078	0.139	1.333
BO-qEI24	100	25	3.006	1.886	1.842	0.223	0.080	0.138	1.128
BO-qEI25	100	25	3.000	1.874	1.776	0.214	0.074	0.138	1.166
BO-qEI26	100	25	2.986	1.918	1.697	0.214	0.076	0.137	1.292
BO-qEI27	100	25	2.946	1.987	1.650	0.204	0.079	0.140	1.508
BO-qEI28	100	25	2.939	1.907	1.643	0.223	0.078	0.139	1.444
BO-qEI29	**100**	**25**	**2.875**	**1.839**	**1.587**	**0.205**	**0.077**	**0.139**	**1.533**

TABLE IV. Results of a single BOTORCH run using all of the ALEPH histograms with $N_{\text{BayesOpt}} = 200$, $N_{\text{Sobol}} = 30$. The first row shows the Monash point and the optimal tune found with BOTORCH is in bold.

Label	N_{BaysOpt}	N_{Sobol}	$O(\hat{\mathbf{x}})$	aLund	bLund	ProbStoUD	probQQtoQ	alphaSvalue	pTmin
Monash			3.2083	0.68	0.98	0.217	0.081	0.1365	0.500
BO-qEI-0	200	30	2.944	1.639	1.573	0.210	0.083	0.139	1.184
BO-qEI-1	200	30	2.941	1.757	1.557	0.204	0.086	0.139	1.437
BO-qEI-2	200	30	2.931	1.929	1.559	0.215	0.077	0.140	1.771
BO-qEI-3	200	30	2.929	1.820	1.619	0.207	0.081	0.138	1.208

(Table continued)

TABLE IV. *(Continued)*

Label	N_{BaysOpt}	N_{Sobol}	$O(\hat{\mathbf{x}})$	aLund	bLund	ProbStoUD	probQQtoQ	alphaSvalue	pTmin
BO-qEI-4	200	30	2.928	1.885	1.556	0.198	0.080	0.139	1.637
BO-qEI-5	200	30	2.925	1.807	1.584	0.204	0.082	0.139	1.418
BO-qEI-6	200	30	2.922	1.796	1.573	0.207	0.077	0.139	1.489
BO-qEI-7	200	30	2.920	1.842	1.524	0.216	0.081	0.139	1.655
BO-qEI-8	200	30	2.919	1.802	1.549	0.218	0.080	0.139	1.512
BO-qEI-9	200	30	2.915	1.783	1.543	0.214	0.082	0.139	1.568
BO-qEI-10	200	30	2.914	1.809	1.570	0.213	0.085	0.140	1.446
BO-qEI-11	200	30	2.913	1.807	1.567	0.198	0.085	0.139	1.385
BO-qEI-12	200	30	2.913	1.743	1.553	0.210	0.085	0.138	1.283
BO-qEI-13	200	30	2.912	1.842	1.555	0.202	0.083	0.140	1.549
BO-qEI-14	200	30	2.911	1.779	1.568	0.213	0.082	0.138	1.380
BO-qEI-15	200	30	2.911	1.855	1.559	0.201	0.085	0.139	1.540
BO-qEI-16	200	30	2.910	1.841	1.582	0.212	0.087	0.139	1.479
BO-qEI-17	200	30	2.909	1.838	1.584	0.217	0.083	0.138	1.408
BO-qEI-18	200	30	2.902	1.814	1.582	0.201	0.085	0.139	1.424
BO-qEI-19	200	30	2.901	1.788	1.571	0.204	0.080	0.139	1.446
BO-qEI-20	200	30	2.898	1.887	1.550	0.211	0.083	0.140	1.636
BO-qEI-21	200	30	2.896	1.756	1.564	0.216	0.080	0.140	1.415
BO-qEI-22	200	30	2.895	1.846	1.594	0.206	0.080	0.139	1.444
BO-qEI-23	200	30	2.893	1.840	1.570	0.206	0.087	0.139	1.433
BO-qEI-24	200	30	2.892	1.870	1.546	0.212	0.077	0.139	1.613
BO-qEI-25	200	30	2.892	1.824	1.555	0.219	0.081	0.139	1.538
BO-qEI-26	200	30	2.891	1.787	1.580	0.208	0.081	0.138	1.318
BO-qEI-27	200	30	2.888	1.793	1.570	0.214	0.081	0.139	1.321
BO-qEI-28	200	30	2.886	1.830	1.612	0.215	0.082	0.139	1.455
BO-qEI-29	**200**	**30**	**2.845**	**1.836**	**1.564**	**0.208**	**0.079**	**0.139**	**1.549**

that $O(\mathbf{x})$ is a relatively smooth function of $\mathbf{x}$ despite the fact that the statistical precision of the predictions is of the same order as that of the ALEPH data. Presumably, this is because these data are of high precision.

VII. CONCLUSIONS

Bayesian optimization has been used to tune six parameters that are most relevant for modeling hadronization in PYTHIA8. Two different implementations of BayesOpt (GPYTORCH and BOTORCH) yield essentially the same tune point, which differs from the Monash tune. Moreover, we find that the tune obtained with BOTORCH fits the ALEPH data better than does the Monash tune. However, in drawing conclusions about the new tune, it is important to take note of the caveats described above. Through extensive investigation, we find that the most important factor in producing a good universal PYTHIA8 tune is to use all the histograms in the ALEPH data.

All results are reproducible using the resources at GitHub [34].

ACKNOWLEDGMENTS

We are thankful to Pushpalatha Bhat and Tony Menzo for their insightful feedback on this project. This document was prepared using the resources of the Fermi National Accelerator Laboratory (Fermilab), a U.S. Department of Energy, Office of Science, Office of High Energy Physics HEP User Facility. Fermilab is managed by FermiForward Discovery Group, LLC, acting under Contract No. 89243024CSC000002. This work is supported in part by the U.S. Department of Energy under Award No. DE-SC0010102 and in part by the University Research Association (URA) Visiting Scholars Program Award No. 23-S-01. Special thanks to the Fermilab LHC Physics Center (LPC) for providing access to resources and facilities that were instrumental to the completion of this work.

DATA AVAILABILITY

The data that support the findings of this article are openly available [35].

APPENDIX A: REPRODUCIBILITY

All of the code, data, and execution environment needed to reproduce our toy problem studies and the PYTHIA8 tune are publicly available.

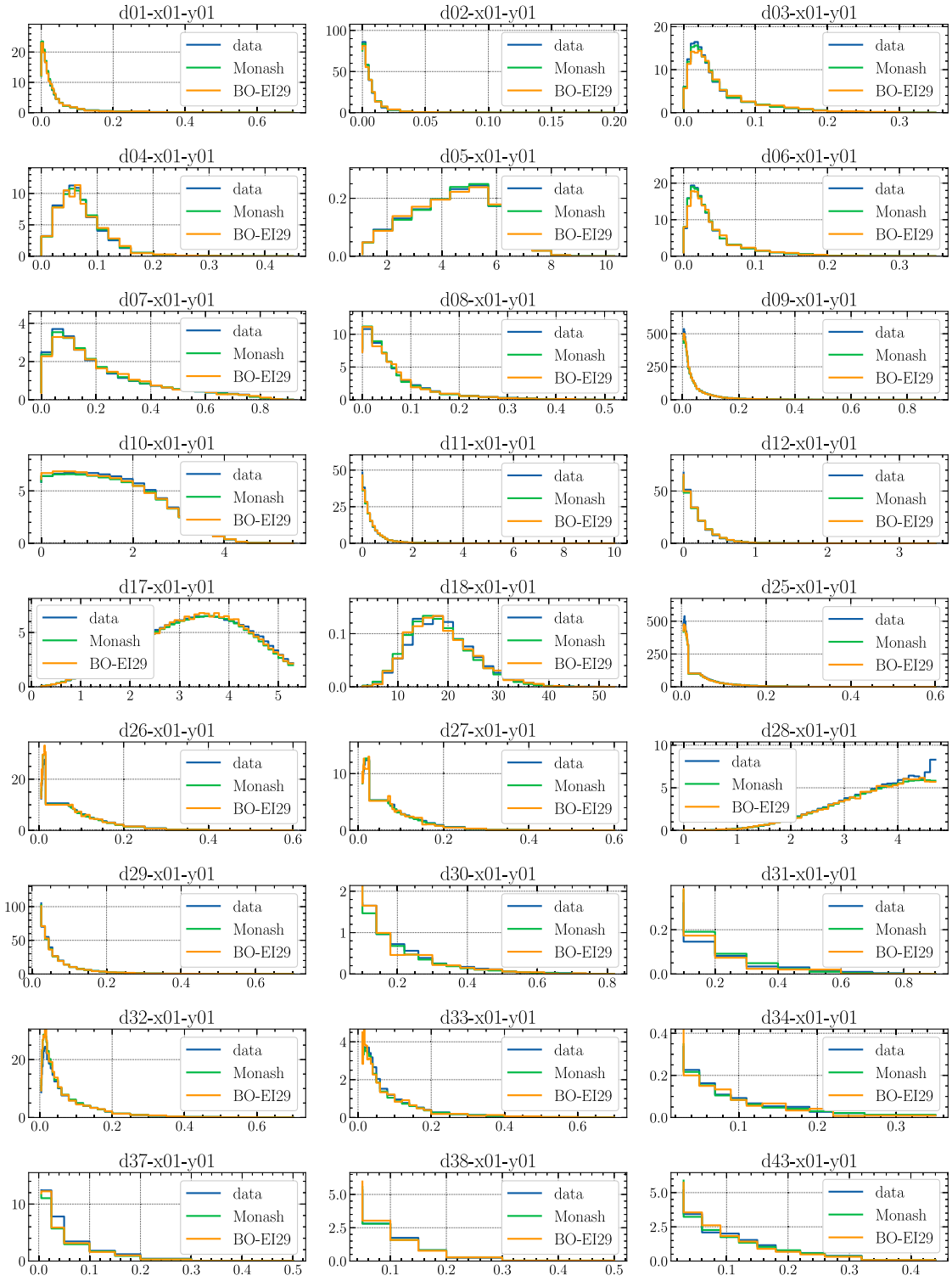


FIG. 2. The BO EI tune results using GPYTORCH for label "BO-EI4" in Table II.

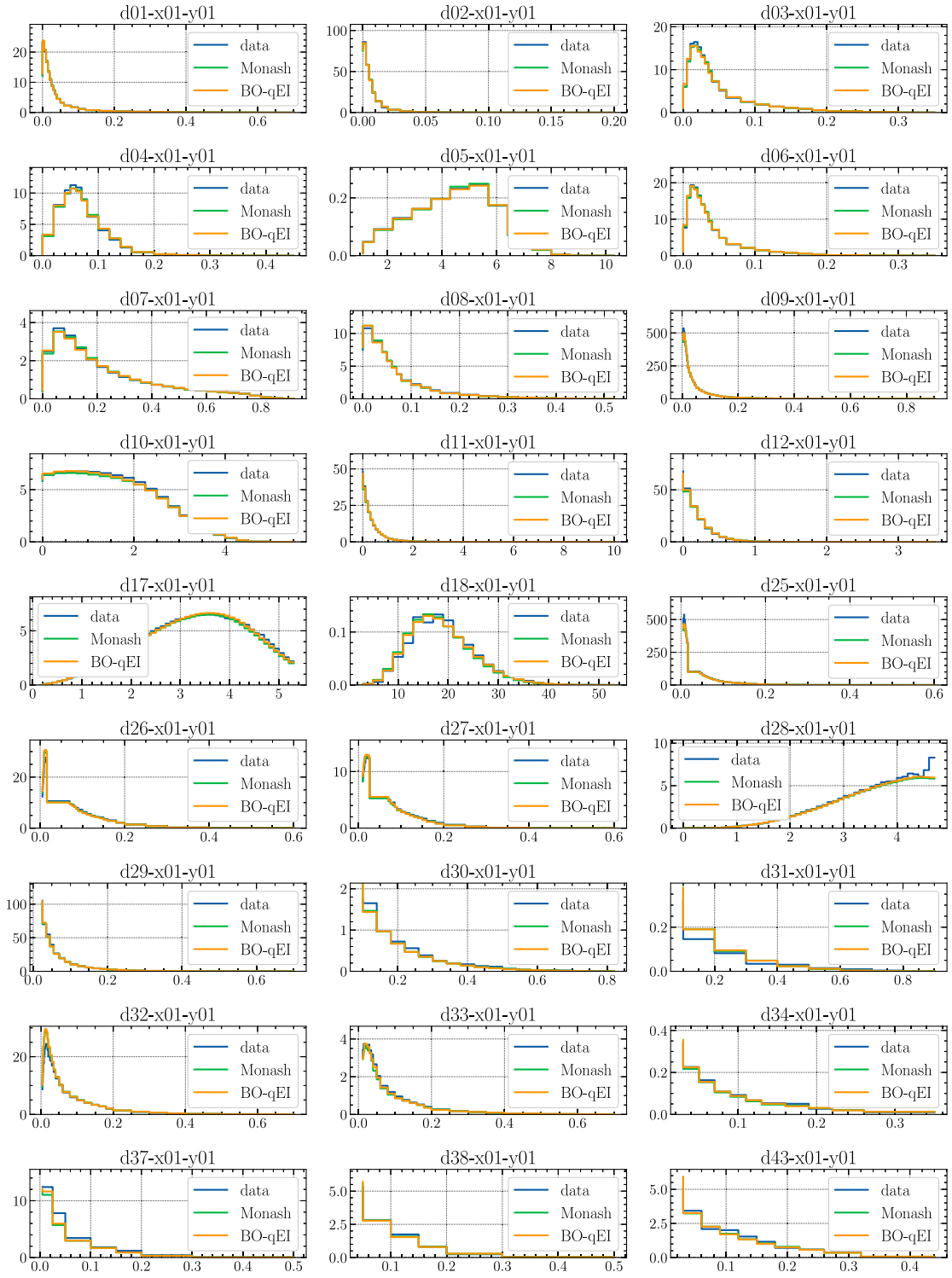


FIG. 3. BOTORCH results on all of the histograms and Eq. (5) with $N_{BO} = 100, N_{\text{Sobol}} = 25$, corresponding to the bolded row in Table III.

1. Repository and code

The complete analysis is hosted on GitHub [34].

Key scripts and notebooks live under `BayesOpt/src/`:

(i) `main_toy.py`: single toy problem Bayesian optimization run;
(ii) `main_toy_all_hyperparams.py`: full 108-experiment hyperparameter scan;
(iii) `validate_pythia_EI.py`: PYTHIA8 Bayesian optimization tune;
(iv) `post_processing_toy.ipynb`, `post_processing_pythia.ipynb`: Jupyter notebooks to regenerate all figures and tables.

2. Docker environment

To avoid dependency issues, we provide a Docker image with all required software such as PYTHIA8 version 8.309, RIVET version 3.1.9, YODA version 1.9.9, HEPMC version 3.02.06, LHAPDF version 6.5.4, GPYTORCH, BOTORCH, etc. After installing Docker, pull and run:

```
docker pull alialkadhim/pythia_sbi_tune:latest
docker run -v $PWD:$PWD \
-w $PWD \
-p 8889:8889 \
-it alialkadhim/pythia_sbi_tune:latest
```

Inside the container,

```
source setup.sh
jupyter-lab - -ip 0.0.0.0 - -port 8889 - -allow-root &
```

Then open the printed URL in your browser.

3. Running the experiments

a. Toy problems

(1) Edit hyperparameters in `BayesOpt/src/configs.py`.
(2) Run a single toy run:
`python BayesOpt/src/main_toy.py.`
(3) To reproduce the full hyperparameter scan,
`python BayesOpt/src/main_toy_all_hyperparams.py.`
(4) Generate diagnostic plots by executing `post_processing_toy.ipynb`.

b. PYTHIA8 tune

(1) Configure the tune parameters in `configs.py`.
(2) Launch the tune
`python BayesOpt/src/validate_pythia_EI.py.`
(3) Produce comparison plots and uncertainty envelopes via `post_processing_pythia.ipynb`.

4. Computational resources

A full PYTHIA8 tune of six parameters $\times 250{,}000$ events per point takes around 374 s on a machine with the following specifications: 8th Generation Intel Core i7-855OU Processor, 16 GB DDR4-2400 SDRAM (2×8 GB). For a reasonable run of $N_{\mathrm{BaysOpt}} = 70$ optimization, this amounts to 7.3 h. The majority of this time is taken up by RIVET [35]. Although we consider a very large number of restarts in our EI GPYTORCH experiments, the time taken by the optimization is small in comparison to the time required for PYTHIA8 and RIVET.

APPENDIX B: DESCRIPTION OF LEPI DATA

TABLE V. Histogram labels and their corresponding titles for the ALEPH data. These data can be accessed with RIVET [35] using the RIVET label ALEPH_1996_S3486095.

Histogram label	Title
d01-x01-y01	Sphericity, S (charged)
d02-x01-y01	Aplanarity, A (charged)
d03-x01-y01	1-thrust, $1 - T$ (charged)
d04-x01-y01	Thrust minor, m (charged)
d05-x01-y01	Two-jet resolution variable, Y_3 (charged)
d06-x01-y01	Heavy jet mass (charged)
d07-x01-y01	C parameter (charged)
d08-x01-y01	Oblateness, $M - m$ (charged)
d09-x01-y01	Scaled momentum, $x_p = \lvert p\rvert/\lvert p_{\mathrm{beam}}\rvert$ (charged)
d10-x01-y01	Rapidity with respect to thrust axes, y_T (charged)
d11-x01-y01	In-plane p_T in GeV with respect to sphericity axes (charged)
d12-x01-y01	Out-of-plane p_T in GeV with respect to sphericity axes (charged)
d17-x01-y01	Log of scaled momentum, $\log(1/x_p)$ (charged)
d18-x01-y01	Charged multiplicity distribution

(Table continued)

TABLE V. *(Continued)*

Histogram label	Title
d19-x01-y01	Mean charged multiplicity
d20-x01-y01	Mean charged multiplicity for rapidity $\|Y\| < 0.5$
d21-x01-y01	Mean charged multiplicity for rapidity $\|Y\| < 1.0$
d22-x01-y01	Mean charged multiplicity for rapidity $\|Y\| < 1.5$
d23-x01-y01	Mean charged multiplicity for rapidity $\|Y\| < 2.0$
d25-x01-y01	$\pi^{\pm}$ spectrum
d26-x01-y01	$K^{\pm}$ spectrum
d27-x01-y01	p spectrum
d28-x01-y01	γ spectrum
d29-x01-y01	π^0 spectrum
d30-x01-y01	η spectrum
d31-x01-y01	η' spectrum
d32-x01-y01	K^0 spectrum
d33-x01-y01	Λ^0 spectrum
d34-x01-y01	Ξ^- spectrum
d35-x01-y01	$\Sigma^{\pm}(1385)$ spectrum
d36-x01-y01	$\Xi^0(1530)$ spectrum
d37-x01-y01	ρ spectrum
d38-x01-y01	$\omega(782)$ spectrum
d39-x01-y01	$K^{*0}(892)$ spectrum
d40-x01-y01	ϕ spectrum
d43-x01-y01	$K^{*\pm}(892)$ spectrum
d44-x01-y02	Mean π^0 multiplicity
d44-x01-y03	Mean η multiplicity
d44-x01-y04	Mean η' multiplicity
d44-x01-y05	Mean $K_S + K_L$ multiplicity
d44-x01-y06	Mean ρ^0 multiplicity
d44-x01-y07	Mean $\omega(782)$ multiplicity
d44-x01-y08	Mean ϕ multiplicity
d44-x01-y09	Mean $K^{*\pm}$ multiplicity
d44-x01-y10	Mean K^{*0} multiplicity
d44-x01-y11	Mean Λ multiplicity
d44-x01-y12	Mean Σ multiplicity
d44-x01-y13	Mean Ξ multiplicity
d44-x01-y14	Mean $\Sigma(1385)$ multiplicity
d44-x01-y15	Mean $\Xi(1530)$ multiplicity
d44-x01-y16	Mean $\Omega^{\mp}$ multiplicity

[1] John C. Collins, Davison E. Soper, and George F. Sterman, Factorization of hard processes in QCD, Adv. Ser. Dir. High Energy Phys. **5**, 1 (1989).

[2] R. Barate *et al.*, Studies of quantum chromodynamics with the ALEPH detector, Phys. Rep. **294**, 1 (1998).

[3] P. Abreu *et al.*, Tuning and test of fragmentation models based on identified particles and precision event shape data, Z. Phys. **73**, 11 (1996).

[4] I. G. Knowles *et al.*, QCD event generators, in *CERN Workshop on LEP2 Physics (Followed by 2nd Meeting, 15-16 Jun 1995 and 3rd Meeting 2-3 Nov 1995)* (1995).

[5] Richard D. Field and Stephen Wolfram, A QCD model for e^+e^- annihilation, Nucl. Phys. B **213**, 65 (1983).

[6] B. Andersson, G. Gustafson, G. Ingelman, and T. Sjöstrand, Parton fragmentation and string dynamics, Phys. Rep. **97**, 31 (1983).

[7] Andy Buckley, Hendrik Hoeth, Heiko Lacker, Holger Schulz, and Jan Eike von Seggern, Systematic event generator tuning for the LHC, Eur. Phys. J. C **65**, 331 (2010).

[8] Mohan Krishnamoorthy, Holger Schulz, Xiangyang Ju, Wenjing Wang, Sven Leyffer, Zachary Marshall, Stephen Mrenna, Juliane Muller, and James B. Kowalkowski,

APPRENTICE for event generator tuning, EPJ Web Conf. **251**, 03060 (2021).
[9] Johannes Bellm and Leif Gellersen, High dimensional parameter tuning for event generators, Eur. Phys. J. C **80**, 54 (2020).
[10] Christan Bierlich, Philip Ilten, Tony Menzo, Stephen Mrenna, Manuel Szewc, Michael K. Wilkinson, Ahmed Youssef, and Jure Zupan, Reweighting Monte Carlo predictions and automated fragmentation variations in PYTHIA8, SciPost Phys. **16**, 134 (2024).
[11] Phil Ilten, Tony Menzo, Ahmed Youssef, and Jure Zupan, Modeling hadronization using machine learning, SciPost Phys. **14**, 027 (2023).
[12] Anders Andreassen and Benjamin Nachman, Neural networks for full phase-space reweighting and parameter tuning, Phys. Rev. D **101**, 091901 (2020).
[13] Anthony P. Austin, Mohan Krishnamoorthy, Sven Leyffer, Stephen Mrenna, Juliane Müller, and Holger Schulz, Practical algorithms for multivariate rational approximation, Comput. Phys. Commun. **261**, 107663 (2021).
[14] Wenjing Wang, Mohan Krishnamoorthy, Juliane Muller, Stephen Mrenna, Holger Schulz, Xiangyang Ju, Sven Leyffer, and Zachary Marshall, BROOD: Bilevel and Robust Optimization and Outlier Detection for efficient tuning of high-energy physics event generators, SciPost Phys. Core **5**, 001 (2022).
[15] Vardan Khachatryan *et al.*, Event generator tunes obtained from underlying event and multiparton scattering measurements, Eur. Phys. J. C **76**, 155 (2016).
[16] Albert M Sirunyan *et al.*, Extraction and validation of a new set of CMS PYTHIA8 tunes from underlying-event measurements, Eur. Phys. J. C **80**, 4 (2020).
[17] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas, Taking the human out of the loop: A review of Bayesian optimization, Proc. IEEE **104**, 148 (2016).
[18] Philip Ilten, Mike Williams, and Yunjie Yang, Event generator tuning using Bayesian optimization, J. Instrum. **12**, P04028 (2017).
[19] Geoffrey F. Chew and Steven C. Frautschi, Regge trajectories and the principle of maximum strength for strong interactions, Phys. Rev. Lett. **8**, 41 (1962).
[20] Edward Shuryak and Ismail Zahed, Hadronic structure on the light front. II. QCD strings, Wilson lines, and potentials, Phys. Rev. D **107**, 034024 (2023).
[21] Christian Bierlich, Smita Chakraborty, Nishita Desai, Leif Gellersen, Ilkka Helenius, Philip Ilten, Leif Lönnblad, Stephen Mrenna, Stefan Prestel, Christian T. Preuss, Torbjörn Sjöstrand, Peter Skands, Marius Utheim, and Rob Verheyen, A comprehensive guide to the physics and usage of PYTHIA8.3, SciPost Phys. Codebases (SciPost, 2022), p. 8.
[22] Eric Brochu, Vlad M. Cora, and Nando de Freitas, A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, arXiv:1012.2599.
[23] Daniel James Lizotte, Practical Bayesian optimization, Ph.D. thesis, University of Alberta, 2008.
[24] Carl Edward Rasmussen and Christopher K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)* (The MIT Press, Cambridge, MA, 2005).
[25] A. J. Booker, J. E. Dennis, P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset, A rigorous framework for optimization of expensive functions by surrogates, Struct. Optim. **17**, 1 (1999).
[26] Marc G. Genton, Classes of kernels for machine learning: A statistics perspective, J. Mach. Learn. Res. **2**, 299 (2001).
[27] Donald R. Jones, Matthias Schonlau, and William J. Welch, Efficient global optimization of expensive black-box functions, J. Global Optim. **13**, 455 (1998).
[28] Thomas A Carlin and Bradley P. Louis, Empirical Bayes: Past, Present, and Future (2002).
[29] I.M Sobol', On the distribution of points in a cube and the approximate evaluation of integrals, USSR Computational Mathematics and Mathematical Physics **7**, 86 (1967).
[30] Diederik P Kingma and Jimmy Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.
[31] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson, GPYTORCH: Blackbox matrix-matrix Gaussian process inference with GPU acceleration, in *Advances in Neural Information Processing Systems* (2018).
[32] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy, BoTorch: A framework for efficient Monte-Carlo Bayesian optimization, arXiv:1910.06403.
[33] Peter Skands, Stefano Carrazza, and Juan Rojo, Tuning PYTHIA8.1: The Monash 2013 tune, Eur. Phys. J. C **74**, 3024 (2014).
[34] https://github.com/AliAlkadhim/BayesOptPythia
[35] Christian Bierlich *et al.*, Robust independent validation of experiment and theory: RIVET version 3, SciPost Phys. **8**, 026 (2020).