Quantifying properties of hot and dense QCD matter through systematic model-to-data comparison

Jonah E. Bernhard, Peter W. Marcy, Christopher E. Coleman-Smith, Snehalata Huzurbazar,

Robert L. Wolpert, and Steffen A. Bass

Quantifying properties of hot and dense QCD matter
through systematic model-to-data comparison

Jonah E. Bernhard,[1] Peter W. Marcy,[2] Christopher E. Coleman-Smith,[1]
Snehalata Huzurbazar,[3] Robert L. Wolpert,[4] and Steffen A. Bass[1]

[1]*Department of Physics, Duke University, Durham, NC 27708*
[2]*Statistical Sciences Group, Los Alamos National Laboratory, Los Alamos, NM 87545*
[3]*Department of Statistics, University of Wyoming, Laramie, WY 82071*
[4]*Department of Statistical Science, Duke University, Durham, NC 27708*
(Dated: April 10, 2015)

We systematically compare an event-by-event heavy-ion collision model to data from the Large Hadron Collider. Using a general Bayesian method, we probe multiple model parameters including fundamental quark-gluon plasma properties such as the specific shear viscosity η/s, calibrate the model to optimally reproduce experimental data, and extract quantitative constraints for all parameters simultaneously. The method is universal and easily extensible to other data and collision models.

I. INTRODUCTION

Relativistic heavy-ion collisions produce a hot, dense phase of strongly-interacting matter commonly known as the quark-gluon plasma (QGP), which rapidly expands and freezes into hadrons [1–7]. Since the QGP is not directly observable—only final-state hadrons are detected—present research seeks to quantify the fundamental properties of the QGP, such as its transport coefficients and the nature of the initial state, through comparisons of experimental measurements to computational model calculations.

Computational models must take a set of input parameters including the physical properties of interest, simulate the full time-evolution of heavy-ion collisions, and produce outputs analogous to experimental measurements. The true values of the physical properties are extracted by calibrating the input parameters so that the model output optimally reproduces experimental data. This generic recipe is called "model-to-data comparison".

Notably, the QGP shear viscosity to entropy density ratio η/s has been constrained by comparing anisotropic flow coefficients v_n between model and experiment. Explicit calculation of η/s directly from QCD is not yet feasible, and while there is a conjectured lower bound $\eta/s \geq 1/4\pi \simeq 0.08$ from AdS/CFT holography [8], model-to-data comparison is the most attractive option for determining the optimal input parameter value and corresponding uncertainty. To this end, previous studies used viscous relativistic fluid dynamics and hybrid transport models to compute v_n at several values of η/s, then chose the value which most closely matched experimental v_n. A variety of complementary calculations have constrained η/s to an approximate range of 0.08–0.20 [9–12].

However, η/s is not the only model input parameter: many other parameters remain poorly determined, e.g. the hydrodynamic thermalization time τ_0 and initial conditions; and models often have non-physical nuisance parameters that nonetheless should be tuned to optimal values. The flow coefficients v_n are only a small subset of all QGP observables: models must also describe basic quantities such as the charged-particle multiplicity and transverse-momentum distributions.

Recent work [13] moved toward a more global analysis of multiple model parameters and observables, but encountered practical limitations attempting to simultaneously tune these free parameters. In general, input parameters correlate among each other and contribute to multiple observables, so they cannot be constrained independently.

Algorithms such as Markov chain Monte Carlo (MCMC) can rigorously explore this type of complex high-dimensional parameter space, but require a very large number of model evaluations—often thousands or millions, depending on the problem at hand. Heavy-ion collision models may run for several hours, so a direct MCMC approach is intractable. The situation is exacerbated when studying event-by-event fluctuations as opposed to average quantities: while event-averaged models save computation time by using a smooth initial condition and single hydrodynamic calculation, event-by-event models have realistic, fluctuated initial conditions, each of which requires its own hydrodynamic treatment. Many thousands of complete events are necessary *at each point in parameter space* to capture event-by-event fluctuations.

These limitations may be overcome through a modern Bayesian method for analyzing computationally expensive models [14–16]. A set of salient model parameters is chosen for calibration—the set should include any fundamental physical properties of interest—and the model is evaluated at a relatively small $\mathcal{O}(10^2)$ number of points. Those points are then interpolated with a Gaussian process emulator [17] to provide a continuous picture of the parameter space. The emulator acts as a fast surrogate to the full model: it predicts model output at arbitrary points in parameter space with negligible computational cost. This effectively removes most practical barriers and enables parameter calibration through standard techniques such as MCMC.

Emulators have been successfully used to study a wide range of physical systems, including galaxy formation [18] and heavy-ion collisions [19–21]. Reference [19] calibrated a hydrodynamic model to identified particle spectra from the Relativistic Heavy Ion Collider (RHIC) and extracted constraints on η/s and several initial state parameters. However, this study used an event-averaged initial condition model, limiting its ability to investigate event-by-event fluctuations.

In this work, we apply Bayesian methodology to a full event-by-event heavy-ion collision model. We calibrate to multiplicity and flow data from the Large Hadron Collider (LHC) and constrain the shear viscosity η/s along with other hydrodynamic and initial condition parameters. The analysis framework handles arbitrary numbers of inputs and outputs, systematically calculates quantitative constraints on all inputs simultaneously, and quickly evaluates the efficacy of physical models.

II. MODEL

State-of-the-art heavy-ion collision models simulate QGP spacetime evolution in several stages [10, 11, 22–27]:

1. an initial condition model describes the initial state and non-equilibrium dynamics until QGP formation,

2. viscous relativistic hydrodynamics calculates the dynamical expansion of the hot and dense QGP medium including the phase transition to a hadron gas,

3. then a particlization model converts the system into a microscopic ensemble of hadrons,

4. and finally a Boltzmann transport model calculates hadronic rescattering and decays.

In this work, we opt for a mature, well-tested set of event-by-event models [27] with an established track record of describing diverse RHIC and LHC data [10, 28, 29]. This choice will permit direct comparison between existing results and the outcome of the following systematic model-to-data comparison. We emphasize, however, that the methodology in this paper can easily be applied to any set of models and corresponding data.

A. Initial conditions

Initial condition models provide the outcome of the collision's pre-equilibrium evolution at the hydrodynamic thermalization time, approximately 0.5 fm/c. Some models explicitly calculate pre-equilibrium dynamics [30] starting from the initial state of the collision; others skip this time frame and generate initial conditions directly at the thermalization time [31–33].

We select two of the most widely used models in the latter category: the Monte Carlo Glauber [32] and Monte Carlo KLN [31] models. Although more sophisticated models were recently introduced [30, 33], both Glauber and KLN provide reasonable event-by-event initial conditions with well-understood behavior and a broad basis of published results.

B. Hydrodynamics

The initial condition furnishes the hydrodynamic stress-energy tensor $T^{\mu\nu}$ at the thermalization time τ_0. Viscous hydrodynamics then solves the conservation equations

$$\partial_\mu T^{\mu\nu} = 0 \qquad (1)$$

where

$$T^{\mu\nu} = (\epsilon + P)u^\mu u^\nu - Pg^{\mu\nu} + \pi^{\mu\nu}; \qquad (2)$$

ϵ, P, and u^μ are the energy density, pressure, and flow velocity of the fluid; $g^{\mu\nu}$ is the metric tensor; and $\pi^{\mu\nu}$ is the shear stress tensor. An equation of state

$$P = P(\epsilon) \qquad (3)$$

closes the system of hydrodynamic equations and is usually provided by a parameterization of lattice QCD calculations.

We employ an improved version of VISH2+1 [34], a stable, extensively tested implementation of boost-invariant viscous hydrodynamics that was recently updated to handle fluctuating event-by-event initial conditions [27]. VISH2+1 uses the prevalent s95 partial chemical equilibrium equation of state [35].

C. Particlization

As the hydrodynamic medium expands and cools, it undergoes a transition from a deconfined QGP to a hot and dense hadronic system. At this point it's advantageous to switch to a microscopic transport model, for such models naturally account for the system's increasing viscosity, non-equilibrium break-up, and eventual freeze-out. Thus, a particlization model converts the fluid into a microscopic ensemble of hadrons once the fluid cools to a pre-specified switching temperature, typically just below the QCD transition temperature $T_c \sim 165$ MeV. The model generates particles by sampling the Cooper-Frye formula [36]

$$E\frac{dN_i}{d^3p} = \int_\sigma f_i(x,p)\, p^\mu\, d^3\sigma_\mu, \qquad (4)$$

where f_i is the distribution function for particle species i, p^μ is the four-momentum, and the integral is taken over

the isothermal spacetime hypersurface σ defined by the switching temperature.

We use a recent hypersurface sampler designed to couple with VISH2+1 [27, 37].

D. Hadronic phase

After particlization, the medium continues to expand as an interacting hadron gas (e.g. scatterings and decays). A hadronic "afterburner" calculates these interactions through the Boltzmann equation

$$\frac{df_i(x,p)}{dt} = \mathcal{C}_i(x,p), \tag{5}$$

where f_i is the distribution function and $\mathcal{C}_i$ is the collision kernel which contains all possible hadronic interactions for particle species i. Particles emerging from the afterburner are analogous to particles streaming into an experimental detector.

We adopt Ultra-relativistic Quantum Molecular Dynamics (UrQMD) [38, 39] as an afterburner.

E. Postprocessing

The full event-by-event model is executed $\mathcal{O}(10^4)$ times for each set of input parameters, yielding $\mathcal{O}(10^7)$ events in total. Events are binned into centrality intervals and the raw event data are postprocessed into physical observables for direct comparison with experiment. In this analysis we calculate the centrality dependence of several standard observables: the average charged-particle multiplicity $\langle N_{\text{ch}} \rangle$ and multi-particle flow cumulants $v_n\{2k\}$. Note that the method trivially extends to arbitrary numbers and types of observables—all that's required is a model calculation and a corresponding experimental measurement.

Flow cumulants $v_n\{2k\}$ are defined as the $2k$-particle correlation function of the nth-order azimuthal anisotropy. For example, the two-particle cumulant is

$$v_n\{2\}^2 \equiv \left\langle e^{in(\phi_i - \phi_j)} \right\rangle, \tag{6}$$

where ϕ_i is the azimuthal angle of the transverse momentum of particle i and the average is over all distinct pairs of particles i, j. The two-particle cumulant is also approximately equal to the root-mean-square of the full v_n distribution [40]:

$$v_n\{2\} \simeq \sqrt{\langle v_n^2 \rangle}. \tag{7}$$

We compute two-particle cumulants for elliptic and triangular flow $v_2\{2\}$, $v_3\{2\}$ using the direct Q-cumulant method [41]. Higher-order cumulants are currently out of reach due to insufficient quantities of events.

Postprocessed observables are compared to corresponding experimental results recently measured by the ALICE experiment at the LHC for Pb-Pb collisions at $\sqrt{s_{\text{NN}}} = 2.76$ TeV [42]. All observables are subjected to the same kinematic cuts as the ALICE detector, namely charged particles with $|\eta| < 1$ and $0.2 < p_T < 3.0$ GeV.

III. EMULATOR

This section constructs a Gaussian process (GP) emulator to act as a surrogate for the full event-by-event model. The strategy is to evaluate the model on a carefully chosen set of input parameter points, then use a GP to interpolate the parameter space. Unlike alternative interpolation schemes such as splines or polynomial interpolation, a GP emulator provides a *probability distribution* at each point in parameter space, hence, it not only predicts the output of the model at arbitrary points in parameter space, but also quantifies the uncertainty of its predictions. Further, GPs are non-parametric interpolators, i.e. they do not require an assumed functional form for the underlying model. These features are essential for emulation of computer codes.

A. Gaussian processes

This subsection summarizes the theory of Gaussian process emulators as detailed in chapter 2 of [17].

A Gaussian process (GP) is defined as a collection of random variables, any finite number of which have a joint Gaussian distribution. A GP may be thought of as a stochastic function $f(\mathbf{x})$ which maps n-dimensional input vectors $\mathbf{x}$ to normally distributed outputs y. It is fully specified by a mean function $\mu(\mathbf{x})$ which gives the mean of f at input point $\mathbf{x}$ and a covariance function $\sigma(\mathbf{x}, \mathbf{x}')$ which provides the covariance of f between a pair of points $\mathbf{x}, \mathbf{x}'$.

As a concrete example, let $\mathbf{x}_1$ be an input point and $y_1 = f(\mathbf{x}_1)$ be the output of the GP at $\mathbf{x}_1$; then y_1 has a normal distribution with mean $\mu(\mathbf{x}_1)$ and variance $\sigma(\mathbf{x}_1, \mathbf{x}_1)$:

$$y_1 \sim \mathcal{N}(\mu(\mathbf{x}_1), \sigma(\mathbf{x}_1, \mathbf{x}_1)). \tag{8}$$

Now if $\mathbf{x}_2$ is another input and $y_2 = f(\mathbf{x}_2)$ is the corresponding output, y_1 and y_2 have a bivariate normal distribution

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} \mu(\mathbf{x}_1) \\ \mu(\mathbf{x}_2) \end{pmatrix}, \begin{pmatrix} \sigma(\mathbf{x}_1, \mathbf{x}_1) & \sigma(\mathbf{x}_1, \mathbf{x}_2) \\ \sigma(\mathbf{x}_2, \mathbf{x}_1) & \sigma(\mathbf{x}_2, \mathbf{x}_2) \end{pmatrix} \right]. \tag{9}$$

In general, the set of m random output variables $\mathbf{y} = \{y_1, \ldots, y_m\} = f(X)$ corresponding to input points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ have a multivariate normal distribution

$$\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma) \tag{10}$$

where

$$\boldsymbol{\mu} = \mu(X) = \{\mu(\mathbf{x}_1), \mu(\mathbf{x}_2), \ldots, \mu(\mathbf{x}_m)\} \tag{11}$$
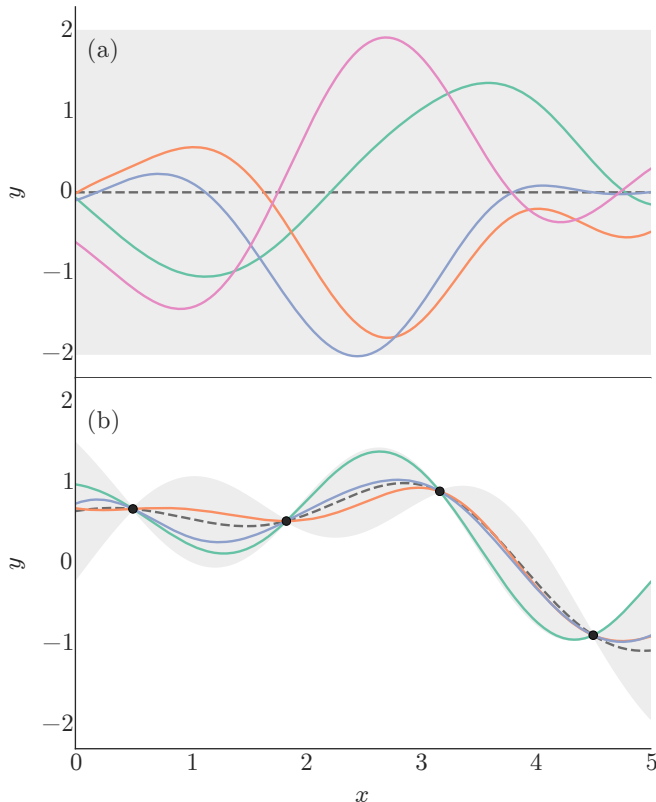
FIG. 1. (Color online) Top: random functions drawn from a Gaussian process using a squared-exponential covariance function with length scale $\ell = 1$. Bottom: functions drawn from a GP conditioned on the training points indicated by dots. In both plots, the dashed line represents the GP mean and the grey band is twice the GP standard deviation (roughly 95% confidence interval).

is the m-dimensional mean vector from applying the mean function to each input, and

$$\Sigma = \sigma(X, X) = \begin{pmatrix} \sigma(\mathbf{x}_1, \mathbf{x}_1) & \cdots & \sigma(\mathbf{x}_1, \mathbf{x}_m) \\ \vdots & \ddots & \vdots \\ \sigma(\mathbf{x}_m, \mathbf{x}_1) & \cdots & \sigma(\mathbf{x}_m, \mathbf{x}_m) \end{pmatrix} \quad (12)$$

is the $m \times m$ covariance matrix from applying the covariance function to each pair of inputs.

In practice, the mean function is often set to zero, since the mean of a distribution can always be subtracted off. The covariance function must be carefully chosen, for it controls the degree of similarity between pairs of points. A standard choice is the squared-exponential function

$$\sigma(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{|\mathbf{x} - \mathbf{x}'|^2}{2\ell^2}\right), \quad (13)$$

where ℓ is a characteristic length scale. Notice that nearby points are strongly correlated ($\sigma \approx 1$) while distant points approach independence ($\sigma \to 0$). This implies that the GP is smooth, i.e. nearby input points produce similar outputs.

Just as we can sample random numbers from a distribution, we can draw random functions from a GP. We choose a set of test points X_* (the reason for the subscript $*$ will become clear in a moment), calculate the covariance matrix $\Sigma = \sigma(X_*, X_*)$, and generate multivariate normal samples from $\mathcal{N}(\mathbf{0}, \Sigma)$. We can then plot the input-output points as smooth curves, as in the top panel of Fig. 1.

Of course, simply generating random functions is not particularly useful—we want to use a GP to interpolate a computer model. Suppose we have a model which takes a vector of input parameters $\mathbf{x}$ and produces an output y according to some unknown GP $f(\mathbf{x})$; for example, f could be a hydrodynamic model with input parameters $\mathbf{x} = (\tau_0, \eta/s)$ and the output could be elliptic flow v_2. We choose a set of training points X, run the model at each point, and observe a set of outputs $\mathbf{y}$. Now, instead of completely random functions, we desire functions which pass through (interpolate) all the training points $(X, \mathbf{y})$. This is achieved by *conditioning* the GP on the training data to yield a predictive distribution for y at any input point $\mathbf{x}$. Recalling the test points X_*, the predictive distribution for the corresponding outputs $\mathbf{y}_*$ is the multivariate normal distribution

$$\begin{aligned} \mathbf{y}_* &\sim \mathcal{N}(\boldsymbol{\mu}, \Sigma), \\ \boldsymbol{\mu} &= \sigma(X_*, X)\sigma(X, X)^{-1}\mathbf{y}, \quad (14) \\ \Sigma &= \sigma(X_*, X_*) - \sigma(X_*, X)\sigma(X, X)^{-1}\sigma(X, X_*). \end{aligned}$$

See the bottom panel of Fig. 1 for an example of conditioning a GP on one-dimensional training data.

We emphasize that the prediction $\mathbf{y}_*$ is not constant, but a *probability distribution* for the model outputs at X_*. As demonstrated in Fig. 1, the predictive distribution is narrow when near the training points and wide when far away, hence, it reflects the true state of knowledge of the interpolation. This is accomplished without assuming a parametric form for the model—we must only assume that the model is a GP with a specified covariance function.

B. Computer experiment design

The full event-by-event model is to be evaluated on a set of m training points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, where each $\mathbf{x}_i$ is an n-dimensional vector of input parameters, so X may be viewed as an $m \times n$ design matrix. This subsection details the choice of input parameters and their distribution in parameter space.

For the present study, we choose a set of $n = 5$ input parameters

$$\mathbf{x} = (\text{Norm}, \text{I.C. param}, \tau_0, \eta/s, \tau_\pi) \quad (15)$$

where

- Norm is the overall normalization factor, a multiplicative constant that determines how much entropy is deposited in the initial condition.

- I.C. param is a parameter specific to each initial condition model. For the Glauber model the parameter is α, which controls how entropy is distributed to wounded nucleons and binary collisions; for the KLN model it is λ, a dimensionless exponent in the saturation scale parameterization. Both are related to the centrality dependence of multiplicity.

- τ_0 is the QGP thermalization time and the starting time for hydrodynamic evolution.

- η/s is the shear viscosity to entropy density ratio of the QGP, assumed to be fixed throughout the hydrodynamic evolution stage.

- τ_π is the shear stress relaxation time, which dictates how quickly the hydro medium relaxes to the Navier-Stokes limit. Since the relaxation time is a function of the shear viscosity and temperature and thus cannot be tuned explicitly, we use the coefficient k_π in the relation $\tau_\pi = 6k_\pi\eta/(sT)$ as a tunable parameter.

We set intentionally large ranges for each parameter, summarized in table I. In this work, we fix several auxiliary parameters to reasonable defaults: nucleons are assumed to be disks with size determined by the inelastic nucleon-nucleon cross section σ_{NN}, and the hydro to micro switching temperature is set to 165 MeV, just below the equation of state transition temperature. However, the method can handle arbitrary numbers of parameters provided the sample size is sufficiently large.

The training points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ must be chosen to simultaneously optimize emulator accuracy and computation time. Perhaps the most obvious design strategy is a uniform grid (factorial design), e.g. k evenly-spaced points in each dimension. Unfortunately, this leads to a total sample size $m = k^n$ which even for a modest $k = 10$ and $n = 5$ is intractably large.

A popular algorithm for generating efficient design points is maximin Latin hypercube sampling [43]. This method produces space-filling randomized designs with several desirable properties:

- The *minimum* distance between points is *maximized*, thus avoiding large gaps and tight clusters.

TABLE I. Input parameter ranges for the Glauber (Glb) and KLN initial condition models and for the hydrodynamic model.

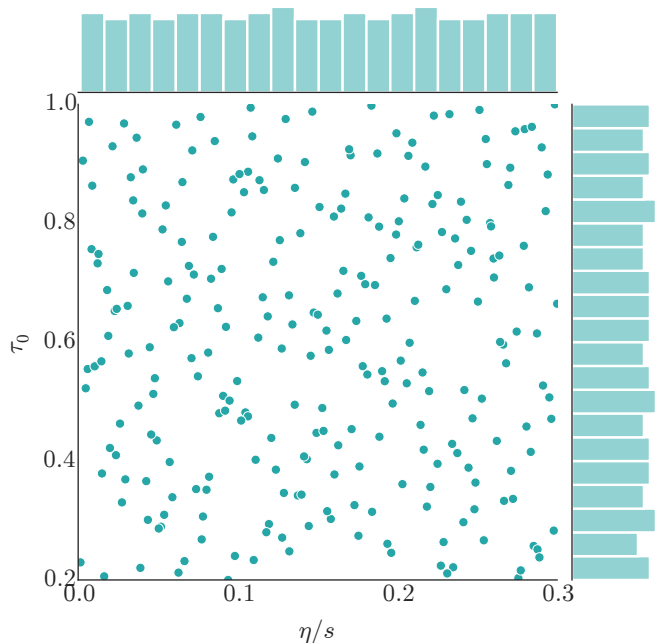| Parameter	Description	Range
Glb Norm	Overall normalization	20–60
Glb α	Wounded nucleon / binary coll.	0.05–0.30
KLN Norm	Overall normalization	5–15
KLN λ	Saturation scale exponent	0.1–0.3
τ_0	Thermalization time	0.2–1.0 fm
η/s	Specific shear viscosity	0–0.3
k_π	Shear relaxation time coeff.	0.2–1.1

FIG. 2. (Color online) The Latin hypercube experiment design projected into the $(\eta/s, \tau_0)$ dimensions. All other parameters also vary across the design, so the points that appear very close in the projection are not necessarily close in the full-dimensional space. The edge histograms show the distributions flattened into one dimension; note that they are space-filling and approximately uniform.

- Projections of the design into lower dimensions are uniformly distributed.

Figure 2 illustrates these traits. A Latin hypercube design with a relatively small sample size provides an efficient scaffolding of parameter space for interpolation by a GP emulator. As a general rule of thumb, a sample size $m \sim 10n$ yields acceptable interpolation accuracy [44] and is a common choice for an initial experiment with limited computation time, however there is no harm in a larger sample.

We use a 256 point Latin hypercube design across the $n = 5$ input parameters; Fig. 2 shows a two-dimensional projection. At each design point, we have executed the event-by-event model $\mathcal{O}(10^3)$ times in each of six centrality bins 0–5%, 10–15%, ..., 50-55%, for both the Glauber and KLN models, yielding $\mathcal{O}(10^7)$ events in total. Two design points that were very near to the edge of the design space gave non-physical results and have been discarded, so the operational design has $m = 254$ points.

Figure 3 shows the postprocessed observables $\langle N_{ch}\rangle$, $v_2\{2\}$, $v_3\{2\}$ as a function of centrality for each point in the design. The results have a broad distribution which is a direct result of the wide ranges of input parameters. There is some statistical error present in $v_3\{2\}$ due to insufficient quantities of events.

Note that these results constitute the training data for the GP emulator, not any kind of best-fit.
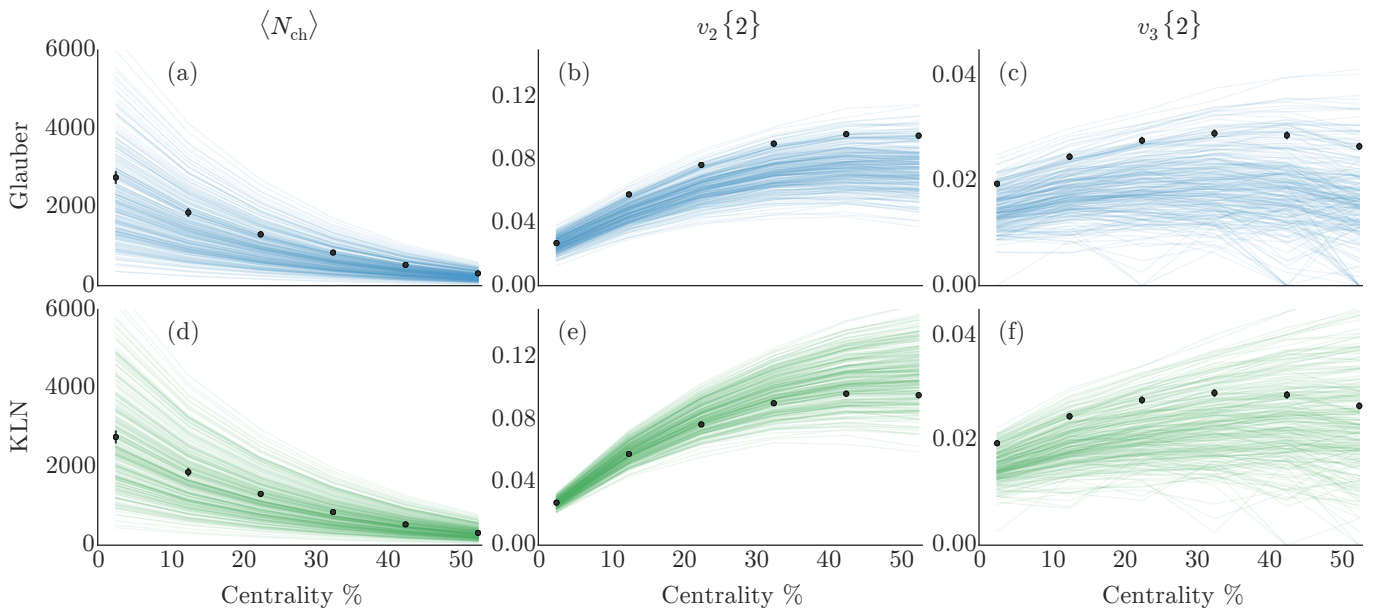
FIG. 3. (Color online) Model calculations from Glauber (top, blue) and KLN (bottom, green) initial conditions. Each plot has 254 lines corresponding to the 254 Latin hypercube design points. From left to right: average charged-particle multiplicity $\langle N_{\mathrm{ch}} \rangle$, elliptic flow two-particle cumulant $v_2\{2\}$, and triangular flow two-particle cumulant $v_3\{2\}$. Data points are experimental measurements from ALICE [42].

C. Multivariate output

Gaussian processes are fundamentally scalar functions, but computer models often produce multivariate output. In general, the model takes the $m \times n$ design matrix X and computes an $m \times p$ output matrix Y. The present event-by-event model has $p = 18$ outputs (three observables each in six centrality bins).

An obvious workaround is to use independent GP emulators for each of the p outputs, however, this would neglect correlations and quickly become unwieldy for higher-dimensional output. Instead, we decompose the outputs into orthogonal linear combinations called principal components (PCs) and emulate each transformed PC. The PCs are uncorrelated by construction and can also be used to reduce the dimensionality of the output space. Figure 4 shows an example PC decomposition.

To calculate the PCs, we first subtract the mean of the output data Y so that each column has mean zero, then compute the eigendecomposition of the sample covariance matrix $Y^{\mathsf{T}}Y$:

$$Y^{\mathsf{T}}Y = U\Lambda U^{\mathsf{T}}, \tag{16}$$

where U is an orthogonal $p \times p$ matrix containing the eigenvectors of $Y^{\mathsf{T}}Y$ and Λ is diagonal containing the eigenvalues $\lambda_1, \ldots, \lambda_p$ in non-increasing order. U now defines a linear transformation which "rotates" the output data Y into PC space:

$$Z = \sqrt{m}\,YU, \tag{17}$$

where Z is an $m \times p$ matrix (same shape as Y) of the transformed PCs. The eigenvalues λ_i represent the variance explained by principal component i; since they are sorted in non-increasing order, the *fraction* of the variance explained by the first $q \leq p$ PCs is

$$V(q) = \frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{p} \lambda_i}. \tag{18}$$

Often, the first few PCs describe most of the variance, as demonstrated for the present data in Fig. 5. Hence we can construct a reduced-dimension transformation with minimal loss of precision by choosing $q < p$ so that $V(q)$ satisfies some threshold (e.g. $V(q) \geq 0.99$) and taking only the first q columns of U:

$$Z_q = \sqrt{m}\,YU_q, \tag{19}$$

where Z_q is now an $m \times q$ matrix.

We may now use q independent GP emulators for each of the columns of Z_q. GPs are conditioned on the design X according to Eq. (14) and predict the PCs Z_* at arbitrary test points X_* which are then transformed back to physical space as

$$Y_* = \frac{1}{\sqrt{m}} Z_* U^{\mathsf{T}}. \tag{20}$$

There is an important caveat for principal components: the original data Y must have a multivariate normal distribution for the transformed PCs Z to be uncorrelated. There is no guarantee that a particular model will produce normally-distributed outputs so this must be verified on a case-by-case basis. For the present event-by-event model we perform the following steps:

1. Assess the normality of each observable $\langle N_{\mathrm{ch}} \rangle$, $v_2\{2\}$, $v_3\{2\}$. While the flow cumulants are approximately normal without modification, we take the square root of multiplicity $\sqrt{\langle N_{\mathrm{ch}} \rangle}$ to obtain a normal distribution, as shown in Fig. 4.

2. Divide each observable by its corresponding experimental value from ALICE [42]. This converts everything to unitless quantities of order one.

3. Multiply each observable by a manually specified weight factor, ratios $1.2 : 1.0 : 0.6$ for observables $\langle N_{\mathrm{ch}} \rangle : v_2\{2\} : v_3\{2\}$. These subjective weights encourage the model to fit more strongly to more fundamental observables, e.g. we prefer a model that describes N_{ch} and v_2 at the expense of v_3 rather than fitting v_3 with incorrect N_{ch}. The weights will be discussed further in the results section IV B.

4. Concatenate the unitless, weighted data into a 254×18 matrix Y, where each row corresponds to a design point and each column to an observable and centrality bin.

5. Subtract the mean of each column and transform Y into principal components Z_q with $q = 5$ PCs, retaining over 99% of the variance as shown in Fig. 5. The PC transformation matrices U_q, shown in Fig. 6, reflect the natural correlations among
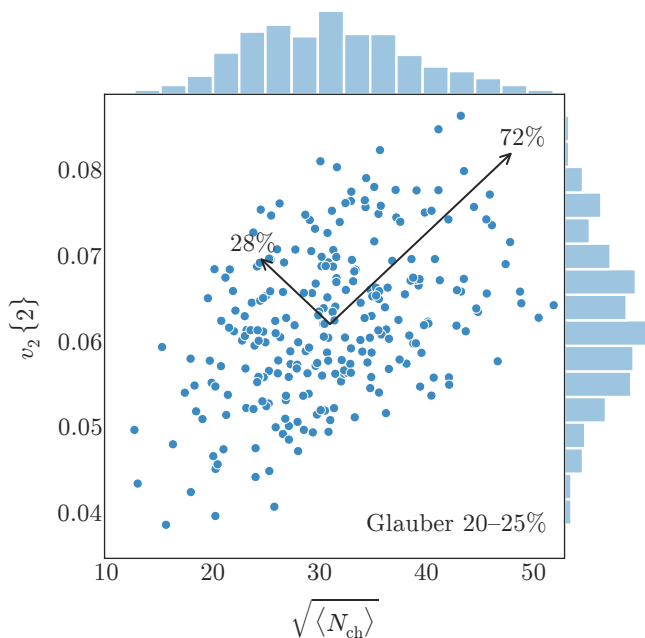

FIG. 5. (Color online) Fraction of the variance $V(q)$ explained by the first q principal components for Glauber (blue circles) and KLN (green triangles). $q = 5$ explains approximately 99% of the total variance, a significant reduction from the original 18 dimensions.

observables, for example all observables are correlated in the first PC ($\sim$75% of total variance), while N_{ch} is anti-correlated with the v_n in the second PC ($\sim$20% of total variance).

We invert these steps to transform PCs back to physical space.

In practice, the covariance method for computing principal components is prone to numerical error, so a more robust algorithm using the singular value decomposition (SVD) is preferred. The SVD of the data Y is

$$Y = VDW^{\mathsf{T}} \tag{21}$$

where V, W are orthogonal matrices containing the so-called left and right-singular vectors of Y and D is diagonal containing the singular values. Inserting (21) into (16) yields

$$Y^{\mathsf{T}}Y = WD^2W^{\mathsf{T}} = U\Lambda U^{\mathsf{T}}, \tag{22}$$

hence the singular values D are the square root of the eigenvalues Λ and the right singular vectors W are the eigenvectors U.

FIG. 4. (Color online) Principal component decomposition of the observables $\sqrt{\langle N_{\mathrm{ch}} \rangle}$, $v_2\{2\}$ for the Glauber model in 20–25% centrality. Each data point represents a model calculation and the edge histograms show the approximate normal distribution of each observable. Arrows represent the PC vectors with lengths proportional to the explained variance.
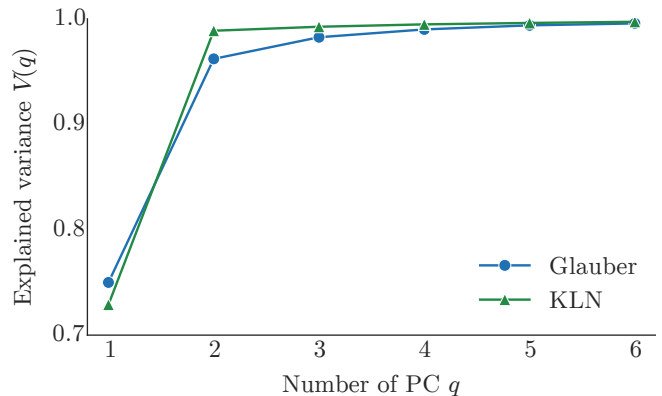
D. Constructing and validating the emulator

We emulate the model by conditioning independent Gaussian processes on each of the principal components Z_q and the input design X according to Eq. (14). Model outputs inevitably include statistical noise, i.e. we cannot compute $y = f(\mathbf{x})$ exactly, only $y = f(\mathbf{x}) + \epsilon$ where ϵ is Gaussian noise. This is accounted for by adding a noise term to the diagonal of the covariance matrix:

$$\sigma(\mathbf{x}, \mathbf{x}') \to \sigma(\mathbf{x}, \mathbf{x}') + \sigma_n^2 \delta_{\mathbf{x}\mathbf{x}'},$$

where σ_n^2 is the variance of the noise and $\delta_{\mathbf{xx'}}$ is a Kronecker delta. Effectively, the noise term relaxes the requirement that the GP must pass exactly through each training point.

We use a squared-exponential covariance function with a noise term:

$$\sigma(\mathbf{x}, \mathbf{x}') = \sigma_{\mathrm{GP}}^2 \exp\left[-\sum_{k=1}^{n} \frac{(x_k - x_k')^2}{2\ell_k^2} \right] + \sigma_n^2 \delta_{\mathbf{xx'}}, \quad (23)$$

where σ_{GP}^2 is the overall variance of the GP and ℓ_k is the characteristic length scale for dimension k. These *hyperparameters* (σ_{GP}, σ_n, ℓ_k) are not known a priori and must be estimated from the training data, however, in the present case predictions appear to be relatively insensitive to the precise choice of hyperparameters, as will be demonstrated promptly. For details about the selection of hyperparameters see appendix A.

As with any interpolation scheme, the GP emulator must be validated to ensure it faithfully predicts model output. In other words, given an arbitrary test point $\mathbf{x}_*$, the GP prediction at $\mathbf{x}_*$ should agree (within its uncertainty) with an explicit computation at $\mathbf{x}_*$. To this end, we have generated a separate 64-point Latin hypercube validation design X_*, evaluated the full event-by-event

	(a) Glauber					(b) KLN				
N_{ch} 00–05	0.38	-0.18	0.14	-0.16	-0.05	0.37	-0.18	-0.01	0.02	-0.04
N_{ch} 10–15	0.37	-0.19	0.09	-0.09	-0.03	0.37	-0.17	0.00	0.02	-0.03
N_{ch} 20–25	0.36	-0.19	0.03	-0.03	-0.01	0.36	-0.17	0.01	0.00	-0.02
N_{ch} 30–35	0.35	-0.20	-0.03	0.04	0.01	0.37	-0.17	0.02	-0.01	0.01
N_{ch} 40–45	0.34	-0.20	-0.09	0.12	0.03	0.37	-0.17	0.03	-0.03	0.03
N_{ch} 50–55	0.34	-0.21	-0.17	0.20	0.06	0.38	-0.17	0.05	-0.06	0.07
v_2 00–05	0.19	0.27	0.53	-0.41	-0.13	0.09	0.16	0.02	0.11	-0.22
v_2 10–15	0.11	0.22	0.25	-0.16	0.01	0.08	0.20	0.12	-0.03	-0.03
v_2 20–25	0.11	0.23	0.17	-0.03	0.07	0.09	0.24	0.17	-0.06	0.03
v_2 30–35	0.12	0.25	0.11	0.07	0.16	0.10	0.28	0.21	-0.16	0.07
v_2 40–45	0.13	0.26	0.02	0.15	0.22	0.13	0.32	0.26	-0.17	0.11
v_2 50–55	0.16	0.28	-0.10	0.21	0.24	0.16	0.37	0.34	-0.30	0.27
v_3 00–05	0.11	0.21	0.12	-0.04	-0.03	0.09	0.16	-0.11	0.36	-0.28
v_3 10–15	0.11	0.22	0.02	0.12	0.03	0.09	0.18	-0.01	0.14	-0.43
v_3 20–25	0.12	0.23	0.02	0.31	0.26	0.10	0.21	0.08	0.11	-0.32
v_3 30–35	0.13	0.25	-0.09	0.29	0.13	0.12	0.25	0.04	0.01	-0.46
v_3 40–45	0.16	0.30	-0.18	0.28	-0.86	0.15	0.29	-0.07	0.75	0.50
v_3 50–55	0.18	0.30	-0.70	-0.61	0.09	0.20	0.37	-0.84	-0.33	0.11
	1	2	3	4	5	1	2	3	4	5
	Principal component					Principal component				

FIG. 6. (Color online) Visualization of the principal component transformation matrices U_q for Glauber (left) and KLN (right). The numerical values of each matrix element are annotated and color-coded, where darker red indicates more positive values, darker blue indicates more negative, and grey indicates zero.

model at each validation point just as for the training design X, and predicted the model outputs at X_* using the GP emulator.

Figure 7 validates that the emulator does indeed faithfully predict the model. Recall that the emulator provides probability distributions of finite width, so it need not predict every validation point exactly—in fact, in the ideal case the residuals would have a normal distribution with mean zero. Most of the uncertainty visible in Fig. 7 is actually due to the statistical noise in the flow cumulants, especially $v_3\{2\}$. The emulator accurately accounts for the noise present in the underlying data.

IV. CALIBRATION

With the validated Gaussian process emulator in hand, it may be used as a fast surrogate to the full event-by-event model for calibration. *Calibration* means tuning the model input parameters so that the output optimally agrees with experimental data and in the process extracting probability distributions for each parameter. Recall the input parameters are

$$\mathbf{x} = (\mathrm{Norm}, \mathrm{I.C.\ param}, \tau_0, \eta/s, k_\pi).$$

Presumably there exists a *true* set of parameters $\mathbf{x}_\star$; the task now is to find the probability distribution of $\mathbf{x}_\star$ given the training data (X, Y) and experimental measurements $\mathbf{y}_{\mathrm{exp}}$. This distribution may be framed in terms of Bayes' theorem as

$$P(\mathbf{x}_\star | X, Y, \mathbf{y}_{\mathrm{exp}}) \propto P(X, Y, \mathbf{y}_{\mathrm{exp}} | \mathbf{x}_\star) P(\mathbf{x}_\star) \quad (24)$$

where

- $P(\mathbf{x}_\star)$ is the *prior* probability which embodies initial knowledge of $\mathbf{x}_\star$;

- $P(X, Y, \mathbf{y}_{\mathrm{exp}} | \mathbf{x}_\star)$ is the likelihood: the probability of observing $(X, Y, \mathbf{y}_{\mathrm{exp}})$ given a proposed value of $\mathbf{x}_\star$; and

- $P(\mathbf{x}_\star | X, Y, \mathbf{y}_{\mathrm{exp}})$ is the *posterior* probability for $\mathbf{x}_\star$ given the observations $(X, Y, \mathbf{y}_{\mathrm{exp}})$. This is the probability distribution we wish to construct.

In general Bayes' theorem has a normalization constant which has been omitted since we are only concerned with relative probabilities.

The remainder of this section applies the methodology from [14–16] to calibrate the model and determine the posterior probability for $\mathbf{x}_\star$.

A. MCMC

The workhorse of any Bayesian calibration is Markov chain Monte Carlo (MCMC), a powerful and flexible method for directly sampling the posterior probability.
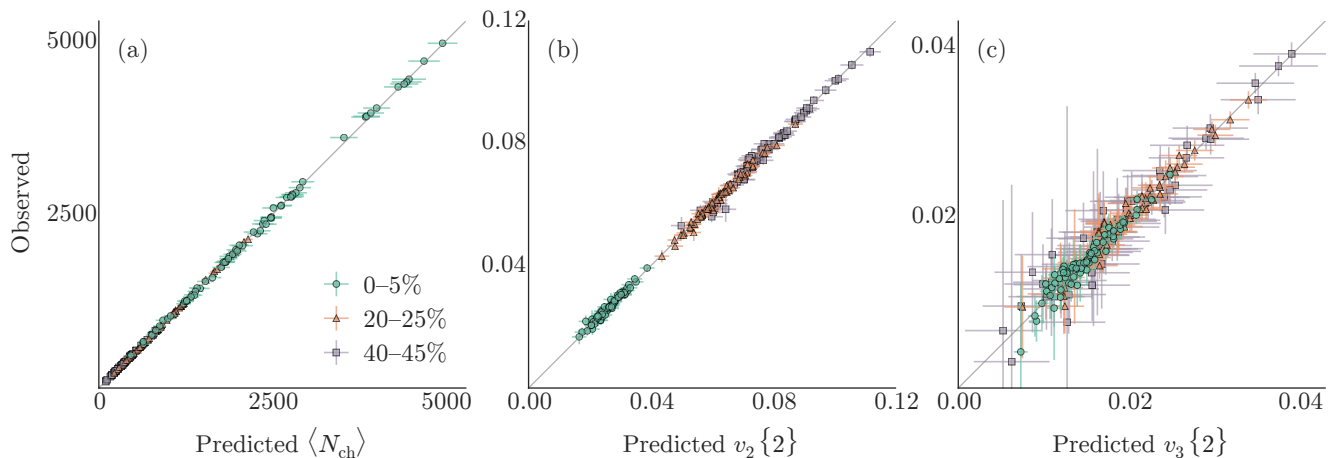
FIG. 7. (Color online) Validation of the Gaussian process emulator for the Glauber model. Each plot shows emulator predictions against explicit calculations for the 64 validation design points in centrality bins 0–5% (green circles), 20–25% (orange triangles), and 40-45% (purple squares). The x-value of each data point is the emulator prediction with 2σ (95%) horizontal error bars, the y-value is the explicit calculation with 2σ (95%) vertical error bars, and the diagonal grey line represents $y = x$.

Perhaps the most common version is the Metropolis-Hastings algorithm, which generates a random walk through parameter space by accepting or rejecting steps based on the posterior probability. For a large number of steps the samples of the random walk equilibrate to the posterior distribution. We use the affine-invariant ensemble sampler for MCMC [45, 46], an alternative algorithm that uses a large ensemble of interdependent walkers. Ensemble sampling notably has a much shorter autocorrelation time than Metropolis-Hastings sampling and hence converges more quickly to the equilibrium distribution.

The MCMC algorithm samples proposal points $\mathbf{x}_\star$ and calculates the posterior probability at each point via Bayes' theorem. We place a non-informative flat prior on $\mathbf{x}_\star$, that is, the prior probability is constant within the design range (table I) and zero outside. We evaluate the likelihood in principal component space:

$$P(\mathbf{z}_{\mathrm{exp}}|\mathbf{x}_\star) \propto \exp\left\{ -\frac{1}{2}(\mathbf{z}_\star - \mathbf{z}_{\mathrm{exp}})^\intercal \Sigma_z^{-1}(\mathbf{z}_\star - \mathbf{z}_{\mathrm{exp}}) \right\}, \quad (25)$$

where $\mathbf{z}_{\mathrm{exp}}$ is the PC transform of the experimental data, $\mathbf{z}_\star$ is the emulator prediction of the PCs at $\mathbf{x}_\star$, and Σ_z is the covariance (uncertainty) matrix on the PCs assuming normally-distributed errors. Given the flat prior, the posterior $P(\mathbf{x}_\star|\mathbf{z}_{\mathrm{exp}})$ is equal to the likelihood within the design range and zero outside.

There are a number of sources of uncertainty including experimental statistical and systematic error, model statistical and systematic error, and emulator uncertainty. In the present study, we do not attempt to precisely account for each contribution, for this would inevitably require dubious assumptions about systematic error correlations and the unknown error of the model. We assume a simple fractional error on the principal components, i.e. the covariance matrix is

$$\Sigma_z = \mathrm{diag}(\sigma_z^2 \mathbf{z}_{\mathrm{exp}}), \quad (26)$$

where σ_z^2 is a manually set constant, $\sigma_z = 0.06$ for the present study, to account for the typical experimental error of 3–5% [42] plus some additional uncertainty. While this is itself a rough assumption, it is perhaps no worse than the alternative, since experimental systematic errors are typically estimated percentages themselves and the principal component transformation automatically includes natural correlations among observables. The primary goal of this study is to develop and test a model-to-data comparison framework; details such as the precise treatment of uncertainties can be improved later.

We run $\mathcal{O}(10^6)$ MCMC steps to allow the chain to equilibrate, discard these "burn-in" samples, then run $\mathcal{O}(10^7)$ steps to generate the posterior distribution.

B. Results

The primary MCMC calibration results are presented in Figs. 8 and 9 for the Glauber and KLN models, respectively. These are visualizations of the posterior probability distributions of the true parameters $\mathbf{x}_\star$, including the distribution of each individual parameter and all correlations. The diagonal histograms show the marginal distributions for each parameter (all other parameters integrated out); the lower-triangle plots are two-dimensional scatter histograms of joint distributions between pairs of parameters, where darker color denotes higher probability density; and the upper triangle has contour plots of the same joint distributions, where the contour lines enclose the 68%, 95%, and 99% confidence regions.

A wealth of information may be gained from these posterior visualizations; the following highlights some important features.

Focusing on the Glauber results in Fig. 8, we see the shear viscosity η/s (fourth diagonal plot) has a narrow
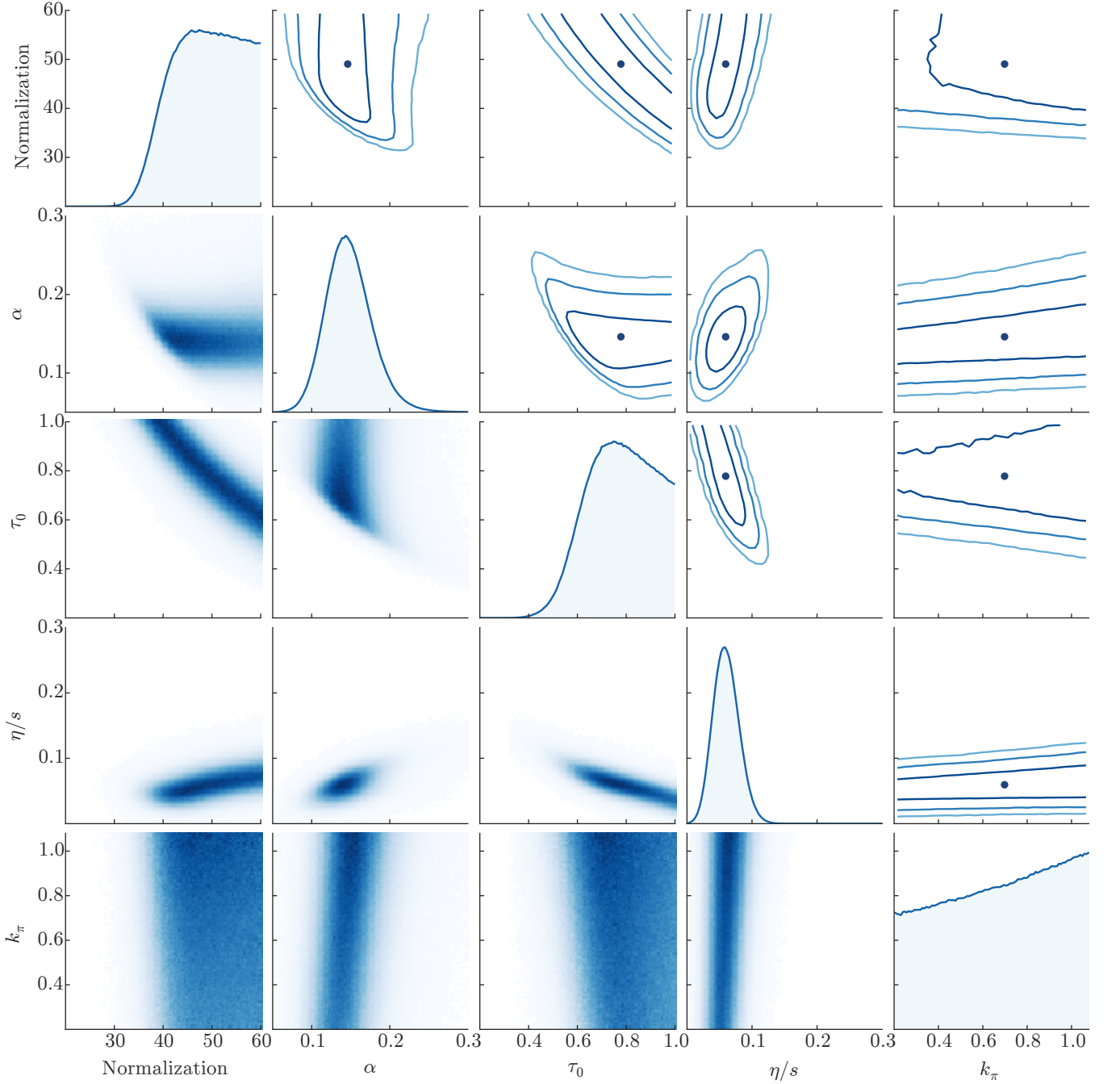
FIG. 8. (Color online) Posterior marginal and joint distributions of the calibration parameters for the Glauber model. On the diagonal are histograms of MCMC samples for the respective parameters, on the lower triangle are two-dimensional scatter histograms of MCMC samples showing the correlation between pairs of parameters, and on the upper triangle are approximate contours for 68%, 95%, and 99% confidence regions along with a dot indicating the median.

approximately-normal distribution located near the commonly quoted value 0.08. As expected, η/s is tightly constrained by experimental flow data. Going across the fourth row, we observe nontrivial correlations among η/s and other parameters, for example, η/s and the hydrodynamic thermalization time τ_0 are negatively correlated (fourth row, third column). As τ_0 increases, the medium

expands as a fluid for less time, so less flow develops, and viscosity must decrease to compensate.

Both τ_0 and normalization (third and first diagonals) have broad distributions without strong peaks, and they are strongly-correlated (third row, first column). This is because the hydrodynamic model is boost-invariant and lacks any pre-equilibrium dynamics, so τ_0 is effectively
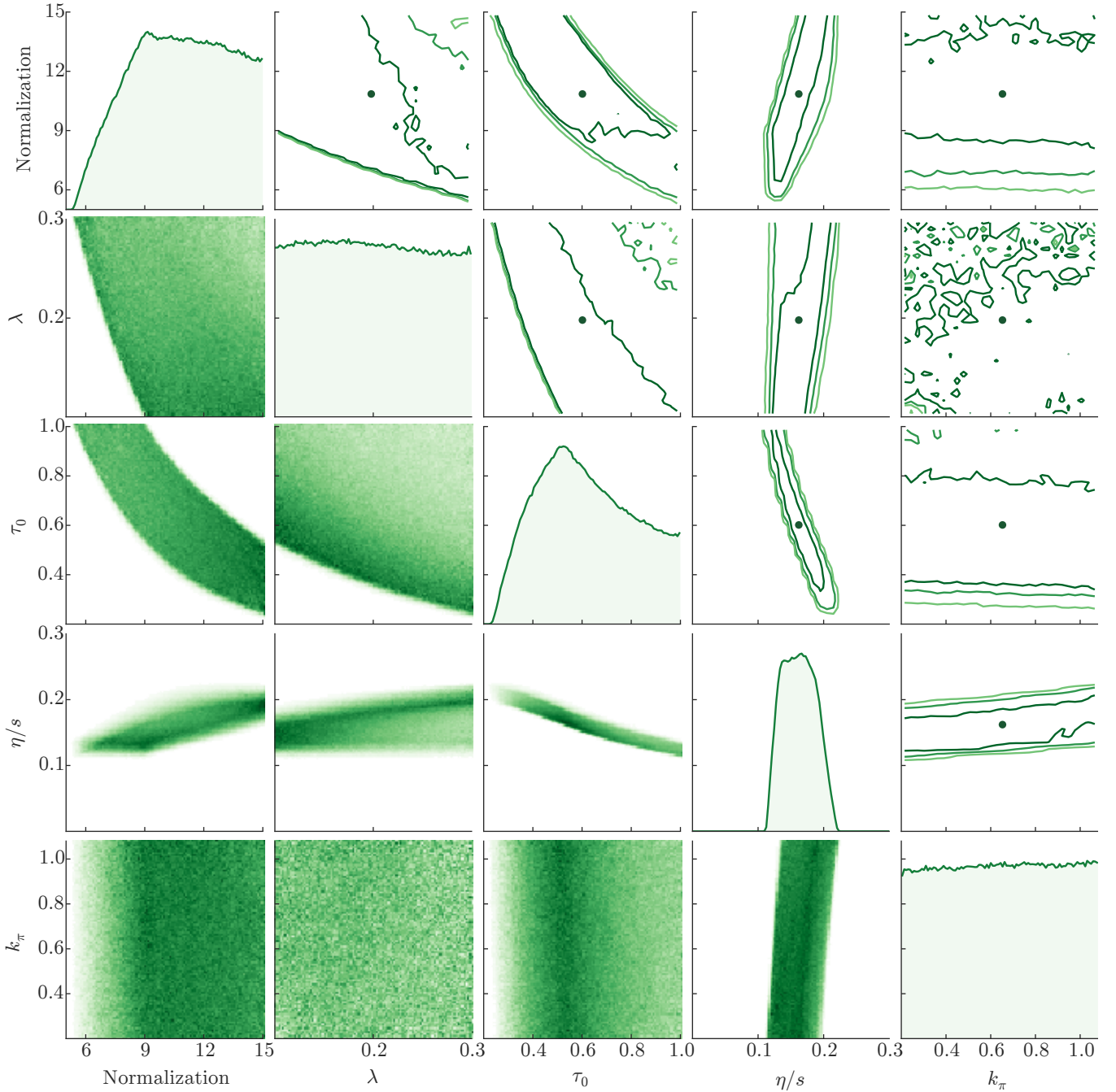
FIG. 9. (Color online) Same as Fig. 8 for the KLN model.

an inverse normalization factor. The joint distribution shows a narrow acceptable band whose shape is governed by the inverse relationship.

The wounded nucleon / binary collision parameter α (second diagonal) has a roughly-normal distribution located near the typical value 0.12. It is mainly related to the slope of multiplicity vs. centrality and hence has a nontrivial correlation with normalization and τ_0, e.g. we can decrease the normalization to the lower end of its distribution provided we also increase α to compensate.

Meanwhile, the shear stress relaxation time coefficient k_π (fifth diagonal) has an almost flat distribution and its joint distributions show no correlations. Evidently, this parameter does not influence flow coefficients or multiplicity.

The KLN results in Fig. 9 generally exhibit wider, less normal distributions than Glauber. This suggests that KLN is somewhat less flexible than Glauber, so its overall behavior is relatively insensitive to the specific values of input parameters.
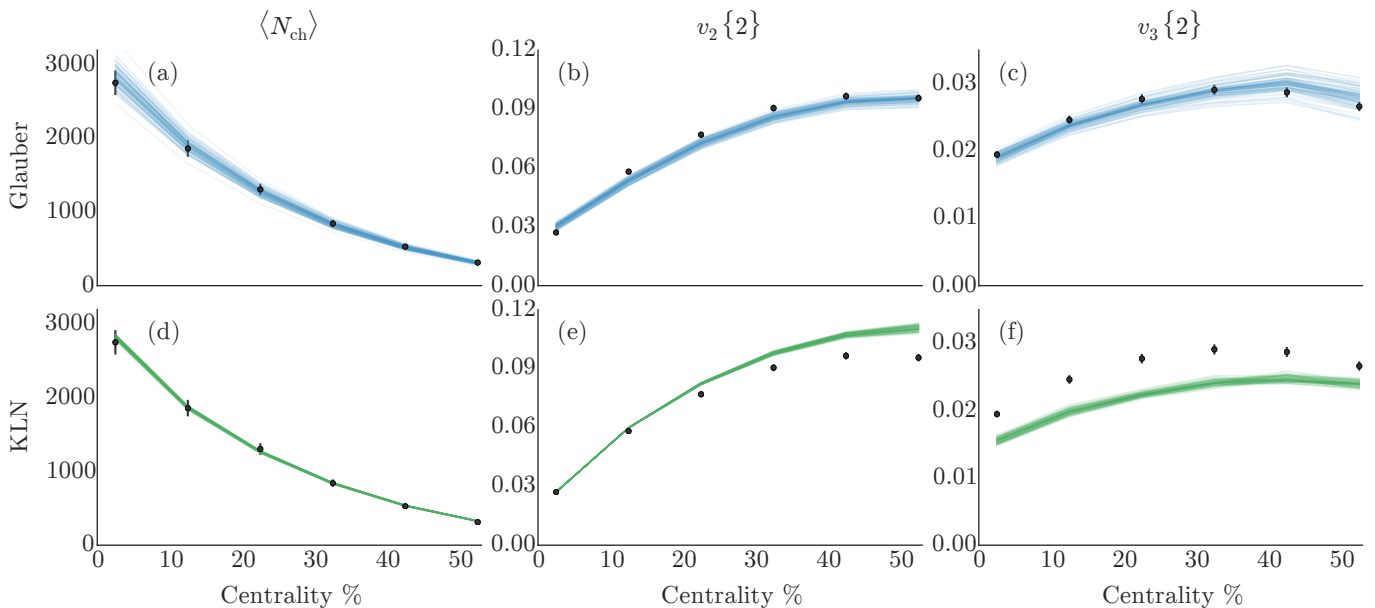
FIG. 10. (Color online) Random realizations of the calibrated posterior for Glauber (top, blue) and KLN (bottom, green) initial conditions. Similar to Fig. 3 except the lines are posterior emulator predictions instead of explicit prior calculations.

The shear viscosity η/s has a narrow, irregular distribution covering the common value 0.20. As with Glauber, η/s has a negative correlation with τ_0, there is a strong inverse relationship between normalization and τ_0, and k_Π has no effect. The KLN parameter λ has a flat marginal distribution, but there are strongly excluded regions in the joint distributions with normalization and τ_0. This appears to be the same effect as observed with Glauber α, except the dependence on λ is significantly weaker.

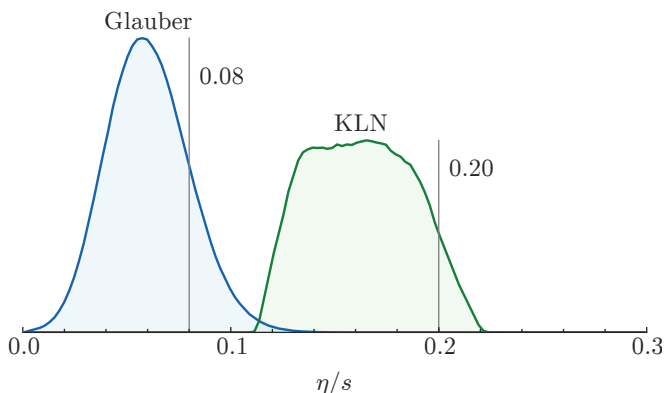The posteriors may be validated by drawing samples from the calibrated distributions and visualizing the corresponding emulator predictions: if the model is correct and properly calibrated, the posterior samples will be close to experimental measurements. Figure 10 confirms—for the most part—that the posteriors are indeed tightly clustered around the data points. Visualizations such as this will always have some uncertainty since samples are drawn from the full posterior, however, Fig. 10 has markedly narrower clusters than Fig. 3, in which the input parameters varied across their full ranges and were not tuned to match experiment.

As shown in the top row of Fig. 10, the Glauber model nearly fits the centrality dependence of all the present observables ($\langle N_{\rm ch}\rangle$, $v_2\{2\}$, $v_3\{2\}$). The v_3 samples have a somewhat larger variance than the others, in part due to the underlying noise in the model calculations and also because v_3 is explicitly given a lower weight (recall that $\langle N_{\rm ch}\rangle : v_2\{2\} : v_2\{3\}$ are weighted $1.2 : 1.0 : 0.6$).

The KLN results in the bottom row tell a somewhat different story, as they cannot fit all observables simultaneously. While the fit to $\langle N_{\rm ch}\rangle$ is excellent, the ratio of v_2 to v_3 is simply too large and the model has no choice but to compromise between the two, similar to previous KLN results [49]. The posterior biases more towards v_2 than v_3 due to the explicit higher weight on v_2.

Figure 11 shows an expanded view of the η/s marginal distributions for Glauber and KLN. The Glauber distribution is approximately normal with mean ~0.06 and 95% confidence interval ~0.02–0.10, consistent with but mostly below 0.08. This is unsurprising and easily within the uncertainty of existing results. KLN has a wider plateau-like distribution with mean ~0.16 and 95% confidence interval ~0.12–0.21. While the common estimate 0.20 was derived primarily from comparisons to v_2, the additional constraint from v_3 shifts the distribution to

FIG. 11. (Color online) Comparison of posterior distributions of η/s for Glauber (blue) and KLN (green). These are the same histograms as in Figs. 8 and 9, expanded and placed on the same axis. The vertical grey lines indicate the common values 0.08 for Glauber and 0.20 for KLN [28, 47].

TABLE II. Quantitative summary of posterior distributions. For each parameter, the previous estimate [28, 47, 48], mean, median, and confidence intervals are given.

	Parameter	Prev. est.	Mean	Median	68% C.I.	95% C.I.	99% C.I.
Glauber	Norm.	57	48.9	49.0	41.6–56.4	36.5–59.4	33.9–59.9
	α	0.12	0.148	0.146	0.119–0.176	0.0954–0.212	0.0808–0.242
	τ_0	0.6	0.776	0.778	0.638–0.922	0.527–0.987	0.461–0.997
	η/s	0.08	0.0604	0.0595	0.0407–0.0801	0.0244–0.101	0.0149–0.116
	k_π	0.5	0.682	0.698	0.373–0.978	0.228–1.08	0.206–1.09
KLN	Norm.	9.9	10.8	10.9	8.15–13.6	6.40–14.8	5.82–15.0
	λ	0.14	0.199	0.198	0.132–0.267	0.105–0.295	0.101–0.299
	τ_0	0.6	0.620	0.602	0.415–0.846	0.302–0.975	0.265–0.995
	η/s	0.20	0.163	0.162	0.135–0.190	0.121–0.208	0.116–0.215
	k_π	0.5	0.651	0.653	0.347–0.955	0.223–1.07	0.205–1.09

somewhat smaller values and causes the plateau shape: rather than a strong peak, there is a range of values which all fit the data roughly equally.

Table II quantitatively summarizes the posterior distributions for each parameter including basic statistics, confidence intervals, and comparisons to previous estimates from earlier work with the same models [28, 47, 48]. All previous estimates fall within 95% confidence intervals, and most within 68%.

V. CONCLUSION

We have applied modern Bayesian methodology to systematically compare an event-by-event heavy-ion collision model to experimental data. We chose a set of salient model parameters including the shear viscosity η/s, evaluated the model over wide ranges of each parameter, and interpolated the results with a Gaussian process emulator. Then, we used the emulator to calibrate the model to optimally reproduce experimental data and thereby extracted probability distributions for the true values of all model parameters simultaneously, including all correlations.

When properly calibrated, the Monte Carlo Glauber model provides a good simultaneous fit to experimental multiplicity and flow data, while the Monte Carlo KLN model fails to simultaneously fit elliptic and triangular flow. The η/s distributions for the Glauber and KLN models are consistent with the commonly quoted values 0.08 and 0.20, respectively, and in general the calibrated distributions reinforce and expand upon existing knowledge of these models.

This study represents a significant step forward in state-of-the-art model-to-data comparison and establishes a framework for future analysis. Since the method does not reduce each parameter to a "best-fit" value but instead furnishes full probability distributions, it may be used to rigorously quantify uncertainties, examine correlations among parameters, and evaluate the efficacy of

physical models, among other possibilities. It is easily extensible to arbitrary numbers of parameters and physical observables and to different models.

Indeed, we plan to apply the methodology to a variety of other models, including the new initial condition model T$_R$ENTo—a flexible effective model which is ideal for this type of analysis [33]—and a 3+1D viscous hydrodynamics model with finite baryon chemical potential combined with recent data from the RHIC beam energy scan. By considering data from multiple beam energies, we can probe the temperature dependence of η/s.

We will include additional physical properties such as the size and shape of nucleons in the initial state, the hydrodynamic equation of state, and the switching temperature from hydrodynamics to microscopic transport; and compare to more observables, e.g. identified particle spectra and differential flow.

Finally, we anticipate upgrades to the methodology itself, notably more rigorous treatment of uncertainties and quantification of input-output correlations (analysis of variance).

the U.S. Department of Energy's Office of Science. All code used in this project is publicly available [52]. Several third-party software packages were invaluable: the Python MCMC toolkit `emcee` [46] and Gaussian process library `george` [53], and the general-purpose tool `GNU Parallel` [54].

Appendix A: Training the emulator

Gaussian process emulators are non-parametric models (they do not assume a functional form) but they do require an assumed covariance function. One typically chooses a parameterized functional form based on physical considerations, for example the squared-exponential function

$$\sigma(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{|\mathbf{x} - \mathbf{x}'|^2}{2\ell^2}\right)$$

generates smoothly-varying processes and is therefore compatible with many models. If the model is known to oscillate, one would choose a periodic covariance function. Most models have statistical noise which one accounts for by adding a diagonal noise term to the covariance function:

$$\sigma(\mathbf{x}, \mathbf{x}') \to \sigma(\mathbf{x}, \mathbf{x}') + \sigma_n^2 \delta_{\mathbf{x}\mathbf{x}'},$$

where σ_n^2 is the variance of the noise and $\delta_{\mathbf{x}\mathbf{x}'}$ is a Kronecker delta.

Covariance functions often have free parameters known as *hyperparameters*, e.g. the squared-exponential correlation length ℓ, which are not known a priori and must be estimated from model data. The selection of hyperparameters is known as *training* and may be accomplished by maximizing the likelihood function [17]

$$\log P(\mathbf{y}|X, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^\mathsf{T}\Sigma_y^{-1}\mathbf{y} - \frac{1}{2}\log|\Sigma_y| - \frac{m}{2}\log 2\pi,$$
$$(A1)$$

where $\mathbf{y}$ is the vector of training outputs, X is the matrix of training input points, $\boldsymbol{\theta}$ is the vector of hyperparameters, and Σ_y is the covariance matrix from applying the covariance function to the training data. Hence, the likelihood is the probability of observing the data given the model. The first term in the likelihood is the fit to data, the second term is a complexity penalty, and the third term is a normalization constant.

This is best demonstrated by an example. The one-dimensional training data in Fig. 12 were generated from a Gaussian process with covariance function

$$\sigma(x, x') = \exp\left(-\frac{|x - x'|^2}{2\ell^2}\right) + \sigma_n^2 \delta_{xx'} \qquad (A2)$$

and hyperparameters $\boldsymbol{\theta} = (\ell, \sigma_n) = (0.5, 0.2)$; now let us pretend we don't know the true values of $\boldsymbol{\theta}$ and attempt to train a Gaussian process on the data. In the top panel of the figure, we use a short length scale ℓ with small noise
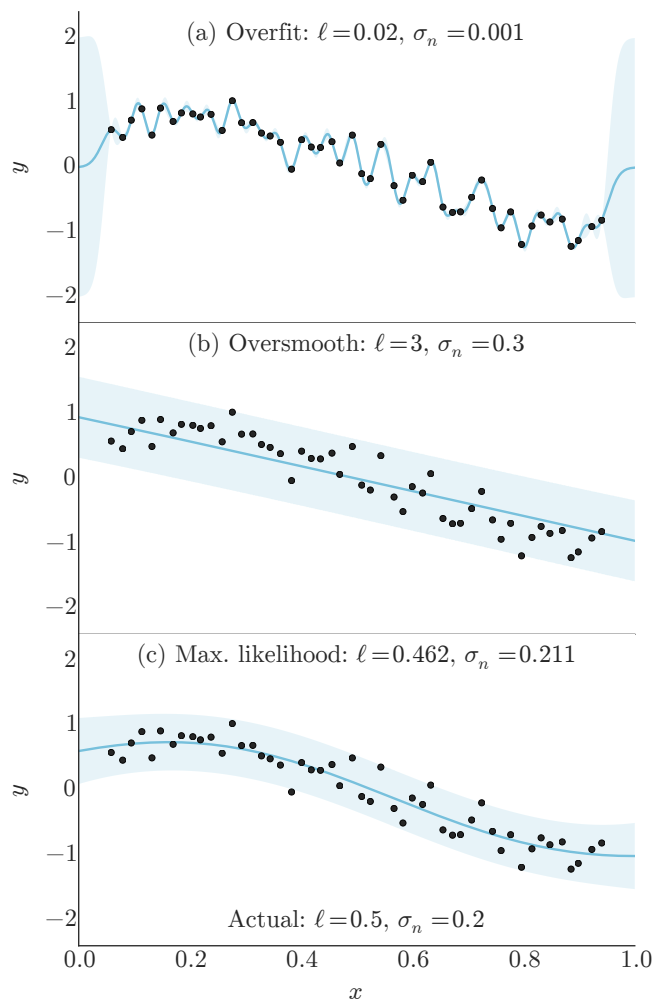

FIG. 12. (Color online) Effect of varying the hyperparameters. Each panel shows a Gaussian process conditioned on fabricated training data using the covariance function Eq. (A2), where the line is the mean and the band is a 2σ confidence interval. The covariance function hyperparameters are different in each plot as indicated by the annotations. The data points are identical in each plot and were generated from a Gaussian process with hyperparameters annotated at the bottom.

σ_n, so the GP interpolates each point exactly; however it rapidly wiggles and is almost certainly "overfit". This choice of hyperparameters has a large complexity penalty and therefore a low likelihood. In the other extreme, we use a long length scale with large noise (middle panel), leading to a nearly linear GP that attributes most of the variance to noise. Here the likelihood is also low due to the poor fit to data. The most likely scenario is the compromise in the bottom panel, in which we estimate the hyperparameters by numerically maximizing the likelihood. Now, the trained GP smoothly interpolates the curvature of the training data while leaving some of the variance as noise, true to the actual GP.

For the present study we use the covariance function

TABLE III. Maximum-likelihood estimates of the covariance function hyperparameters.

		Principal component				
		1	2	3	4	5
Glauber	σ_{GP}	4.29	2.20	2.67	0.923	0.558
	ℓ Norm	1.89	1.62	1.95	0.622	0.300
	$\ell\,\alpha$	3.26	1.61	1.49	0.579	10.0
	$\ell\,\tau_0$	1.20	1.05	1.65	1.34	0.300
	$\ell\,\eta/s$	1.69	1.01	1.17	2.09	10.0
	$\ell\,k_\pi$	10.0	4.77	4.46	10.0	1.48
	σ_n	0.0349	0.106	0.558	0.800	0.933
KLN	σ_{GP}	5.11	3.36	1.48	1.28	0.996
	ℓ Norm	1.82	1.39	1.16	0.907	0.713
	$\ell\,\lambda$	8.47	4.54	0.985	1.10	0.300
	$\ell\,\tau_0$	0.927	0.678	0.808	0.534	0.359
	$\ell\,\eta/s$	1.63	0.851	0.500	0.434	0.369
	$\ell\,k_\pi$	10.0	8.33	2.04	1.43	0.389
	σ_n	0.0192	0.0568	0.807	0.803	0.606

given in Eq. (23) and restated here:

$$\sigma(\mathbf{x}, \mathbf{x}') = \sigma_{\mathrm{GP}}^2 \exp\left[-\sum_{k=1}^{n} \frac{(x_k - x_k')^2}{2\ell_k^2}\right] + \sigma_n^2 \delta_{\mathbf{x}\mathbf{x}'}.$$

The hyperparameter σ_{GP}^2 is the overall variance of the Gaussian process and the ℓ_k are the independent length scales for each design parameter. We estimate the hyperparameters by numerically maximizing the likelihood (A1) using a nonlinear conjugate gradient algorithm. Since the likelihood may have non-optimal local extrema, we repeat the optimization algorithm many times (minimum 100) from different random starting points.

Table III lists the maximum-likelihood estimates of the hyperparameters for each principal component in standardized units—input parameters scaled to $[0, 1]$ and principal components scaled to unit variance. We constrain the length scales to $[0.3, 10]$ for numerical robustness.

In this work we fix the hyperparameters to the maximum-likelihood estimates during calibration. This neglects uncertainty in the hyperparameters themselves, although the present event-by-event model is well-behaved and the sample size is large, so varying the hyperparameters weakly affects the actual emulator predictions. But ideally, one would consider *all* predictions consistent with the data—not only the most likely—by MCMC-sampling the hyperparameter posteriors. This significantly increases computational cost, since the GPs must be re-conditioned for every set of hyperparameters, and conditioning requires computation of the inverse covariance matrix, an $\mathcal{O}(n^3)$ operation. We forgo this refinement until a future study.

[1] I. Arsene *et al.* (BRAHMS Collaboration), Nucl.Phys. **A757**, 1 (2005).

[2] K. Adcox *et al.* (PHENIX Collaboration), Nucl.Phys. **A757**, 184 (2005).

[3] B. Back, M. Baker, M. Ballintijn, D. Barton, B. Becker, *et al.*, Nucl.Phys. **A757**, 28 (2005).

[4] J. Adams *et al.* (STAR Collaboration), Nucl.Phys. **A757**, 102 (2005).

[5] M. Gyulassy and L. McLerran, Nucl.Phys. **A750**, 30 (2005).

[6] B. Muller and J. L. Nagle, Ann.Rev.Nucl.Part.Sci. **56**, 93 (2006).

[7] B. Muller, J. Schukraft, and B. Wyslouch, Ann.Rev.Nucl.Part.Sci. **62**, 361 (2012).

[8] P. Kovtun, D. T. Son, and A. O. Starinets, Phys.Rev.Lett. **94**, 111601 (2005).

[9] M. Luzum and P. Romatschke, Phys.Rev. **C78**, 034915 (2008).

[10] H. Song, S. A. Bass, U. Heinz, T. Hirano, and C. Shen, Phys.Rev.Lett. **106**, 192301 (2011).

[11] B. Schenke, S. Jeon, and C. Gale, Phys.Rev.Lett. **106**, 042301 (2011).

[12] M. Luzum and J.-Y. Ollitrault, Nucl.Phys. **A904-905**, 377c (2013).

[13] R. Soltz, I. Garishvili, M. Cheng, B. Abelev, A. Glenn, *et al.*, Phys.Rev. **C87**, 044901 (2013).

[14] A. OHagan, Rel.Engin.Sys.Safety **91**, 1290 (2006).

[15] D. Higdon, J. Gattiker, B. Williams, and M. Rightley, J.Amer.Stat.Assoc. **103** (2008).

[16] D. Higdon, J. D. McDonnell, N. Schunck, J. Sarich, and S. M. Wild, (2014), arXiv:1407.3017 [physics.data-an].

[17] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning* (MIT Press, 2006).

[18] F. A. Gomez, C. E. Coleman-Smith, B. W. O'Shea, J. Tumlinson, and R. Wolpert, Astrophys.J. **760**, 112 (2012).

[19] J. Novak, K. Novak, S. Pratt, J. Vredevoogd, C. Coleman-Smith, *et al.*, Phys.Rev. **C89**, 034917 (2014).

[20] S. Pratt, C. Ratti, and W. P. McCormack, (2014), arXiv:1409.2164 [nucl-th].

[21] S. Pratt, E. Sangaline, P. Sorensen, and H. Wang, (2015), arXiv:1501.04042 [nucl-th].

[22] S. Bass and A. Dumitru, Phys.Rev. **C61**, 064909 (2000).

[23] D. Teaney, J. Lauret, and E. Shuryak, (2001), arXiv:nucl-th/0110037 [nucl-th].

[24] T. Hirano, U. W. Heinz, D. Kharzeev, R. Lacey, and Y. Nara, Phys.Lett. **B636**, 299 (2006).

[25] C. Nonaka and S. A. Bass, Phys.Rev. **C75**, 014902 (2007).

[26] H. Petersen, J. Steinheimer, G. Burau, M. Bleicher, and H. Stocker, Phys.Rev. **C78**, 044901 (2008).

[27] C. Shen, Z. Qiu, H. Song, J. Bernhard, S. Bass, *et al.*, (2014), arXiv:1409.8164 [nucl-th].

[28] C. Shen, S. A. Bass, T. Hirano, P. Huovinen, Z. Qiu, *et al.*, J.Phys. **G38**, 124045 (2011).

[29] H. Song, S. A. Bass, U. Heinz, T. Hirano, and C. Shen, Phys.Rev. **C83**, 054910 (2011).

[30] B. Schenke, P. Tribedy, and R. Venugopalan,

Phys.Rev.Lett. **108**, 252301 (2012).

[31] H.-J. Drescher, A. Dumitru, A. Hayashigaki, and Y. Nara, Phys.Rev. **C74**, 044905 (2006).

[32] M. L. Miller, K. Reygers, S. J. Sanders, and P. Steinberg, Ann.Rev.Nucl.Part.Sci. **57**, 205 (2007).

[33] J. S. Moreland, J. E. Bernhard, and S. A. Bass, (2014), arXiv:1412.4708 [nucl-th].

[34] H. Song and U. W. Heinz, Phys.Rev. **C77**, 064901 (2008).

[35] P. Huovinen and P. Petreczky, Nucl.Phys. **A837**, 26 (2010).

[36] F. Cooper and G. Frye, Phys.Rev. **D10**, 186 (1974).

[37] Z. Qiu, (2013), arXiv:1308.2182 [nucl-th].

[38] S. Bass, M. Belkacem, M. Bleicher, M. Brandstetter, L. Bravina, *et al.*, Prog.Part.Nucl.Phys. **41**, 255 (1998).

[39] M. Bleicher, E. Zabrodin, C. Spieles, S. Bass, C. Ernst, *et al.*, J.Phys. **G25**, 1859 (1999).

[40] N. Borghini, P. M. Dinh, and J.-Y. Ollitrault, Phys.Rev. **C63**, 054906 (2001).

[41] A. Bilandzic, R. Snellings, and S. Voloshin, Phys.Rev. **C83**, 044913 (2011).

[42] B. B. Abelev *et al.* (ALICE Collaboration), Phys.Rev. **C90**, 054901 (2014).

[43] M. D. Morris and T. J. Mitchell, J.Stat.Plan.Inf. **43**, 381 (1995).

[44] J. L. Loeppky, J. Sacks, and W. J. Welch, Technometrics **51**, 366 (2009).

[45] J. Goodman and J. Weare, Comm.App.Math.Comp.Sc. **5**, 65 (2010).

[46] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman, PASP **125**, 306 (2013).

[47] U. Heinz, C. Shen, and H. Song, AIP Conf.Proc. **1441**, 766 (2012).

[48] C. Shen, Z. Qiu, and U. Heinz, private communication (2013).

[49] Z. Qiu, C. Shen, and U. Heinz, Phys.Lett. **B707**, 151 (2012).

[50] R. Pordes, B. Kramer, D. Olson, M. Livny, A. Roy, *et al.*, J.Phys.Conf.Ser. **78**, 012057 (2007).

[51] I. Sfiligoi, D. C. Bradley, B. Holzman, P. Mhashilkar, S. Padhi, *et al.*, WRI World Congress **2**, 428 (2009).

[52] `https://github.com/jbernhard/ebe-osg`, `https://github.com/jbernhard/mtd`, `https://github.com/jbernhard/mtd-paper`.

[53] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O'Neil, (2014), arXiv:1403.6015 [math.NA].

[54] O. Tange, ;login: **36**, 42 (2011).