Teaching PROFESSOR new math

Anthony Austin[1,*], *Sven* Leyffer[2,**], *Steven* Mrenna[3,***], *Juliane* Müller[4,****], and *Holger* Schulz[3,5,†]

[1]Virginia Tech
[2]Argonne National Laboratory
[3]Fermi National Accelerator Laboratory
[4]Lawrence Berkeley National Laboratory
[5]University of Cincinnati

Abstract. The software package PROFESSOR [1] provides machinery to facilitate Monte-Carlo event-generator tuning. The method is based on the numerical optimization of a goodness-of-fit measure defined between measured experimental data and MC predictions. As the latter are typically very expensive to evaluate, PROFESSOR replaces them with fast to evaluate surrogates. We present improvements to the method in the numerical optimization as well as the surrogate construction. Firstly, we introduce an algorithm and implementation of a bi-level optimization problem aimed at improving scenarios were the underlying physics model is unable to describe all data to the same accuracy. Secondly, we display our implementation of an algorithm for multivariate rational approximations which are superior to the previously used polynomial surrogates.

1 Introduction

Monte-Carlo (MC) event generators are an essential tool for physics analyses. Their predictions are used e.g. to estimate background contributions, to correct for detector effects and ultimately test physics models against measured data. It is therefore desirable to achieve particle simulations that are as precise as possible. In the last decade, tremendous success was achieved in the perturbative regime of the underlying calculations. Realistic MC generators, however, also contain physics in a regime where couplings are large and field theory methods break down. Programs such as Pythia8 [2] implement non-perturbative aspects of the simulation by means of heuristic models that introduce parameters that need to be adjusted to meaningful values with the objective to reproduce nature with high fidelity. This process, which is commonly referred to as "tuning", is helped greatly by the existence of tools like Rivet [3] which conveniently analyze the simulated events to allow for immediate comparison

*e-mail: apaustin@vt.edu
**e-mail: leyffer@mcs.anl.gov
***e-mail: mrenna@fnal.gov
****e-mail: julianemueller@lbl.gov
†e-mail: hschulz@fnal.gov

with a trove of experimental data, e.g. in the form of a goodness-of-fit measure such as

$$\chi^2(p) = \sum_b^{N_{\text{bins}}} \frac{[d_b - \text{MC}_b(p)]^2}{\sigma_b^2}, \tag{1}$$

where the sum runs over individual bins, b. The measured data is given by d_b and some uncertainty σ_b. The corresponding Monte-Carlo prediction $\text{MC}_b(p)$ is dependent on model parameters, p. The optimization problem is defined such that we want to find the point $\hat{p}$ that minimizes the expression in (1). The computational cost of running the MC generators for a given point p range from minutes to days, depending on the physics process. For most cases, this is too expensive for numerical minimization. PROFESSOR instead uses surrogates f_b that are trained on a moderate number of $\text{MC}(p)$. The evaluation of $f_b(p)$ is of the order of micro-seconds which is much more amenable for numerical minimization purposes.

Ideally the models would be able to describe all measured data with one set of parameter values. However, this is generally not the case. Instead, specialized tunes are developed by biasing the optimization process to give more weight to certain datasets than others. This process typically involves many iterations in a trial-and-error fashion. We attempt to automate this procedure in Sect. 2 by formulating the problem in (1) as bilevel optimization problem.

Although fairly robust and widely applicable, the multivariate polynomial approximations currently used in PROFESSOR do have limitations if the true functional form of the to be approximated data exhibits traits of rational functions. In Sect. 3 we demonstrate our implementation of an algorithm for multivariate rational approximations.

2 Bi-level Optimization

The difficulty with solving the "classical" tuning problem arise from the following challenges:

- Incompatible datasets and mismodeling in the MC generator necessitate the introduction of tuning weights w_O for each observable
- Adjusting these weights has so far been a manual procedure in which the user iteratively optimizes the classical problem (from here on "inner optimization") and look at resulting plots

To overcome these difficulties, we propose an automated procedure to replace the manual weight adjustment. For this purpose, we reformulate the problem as bilevel optimization problem, where on the upper level ("outer optimization") we optimize over the weights-per-observable, and on the lower level ("inner optimization") we optimize over the parameters p such that the goodness of fit function is minimized given the weights from the upper level.

The resulting bilevel optimization problem then becomes:

$$\min_{\boldsymbol{w}\in[0,1]^{N_O}, \hat{\boldsymbol{p}}_{\boldsymbol{w}}\in\Omega} g(\boldsymbol{w}, \hat{\boldsymbol{p}}_{\boldsymbol{w}}) \tag{2a}$$

$$\text{s.t.} \sum_{O=1}^{N_O} w_O = 1 \tag{2b}$$

$$\hat{\boldsymbol{p}}_{\boldsymbol{w}} = \arg\min_{\boldsymbol{p}\in\Omega} \phi^2(\boldsymbol{p}, \boldsymbol{w}) \tag{2c}$$

where $\boldsymbol{w}$ represents the vector of weights per observable, N_O is the number of observables, Ω is the space in which the parameters $\boldsymbol{p}$ live, and $\hat{\boldsymbol{p}}_{\boldsymbol{w}}$ is the optimal solution (the optimal parameters) as determined by solving the inner optimization problem for a given set of weights

$\boldsymbol{w}$. The function $g(\cdot)$ represents our objective function at the outer level (more details below). Thus, in order to compute the objective function at the upper level (2a), we have to solve the optimization problem at the inner level (2c), which is defined as

$$\phi^2(\boldsymbol{p}, \boldsymbol{w}) = \sum_{O=1}^{N_O} w_O^2 \cdot \sum_{b \in O} \frac{(f_b(\boldsymbol{p}) - d_b)^2}{\sigma_b^2}. \tag{3}$$

For each $\hat{\boldsymbol{p}}_{\boldsymbol{w}}$, we can calculate the per-observable goodness-of fit

$$\nu_O(\hat{\boldsymbol{p}}_{\boldsymbol{w}}) = \frac{1}{N_{\text{bins}}(O)} \sum_{b \in O} \frac{(f_b(\hat{\boldsymbol{p}}) - d_b)^2}{\sigma^2}, \; O = 1, \ldots, N_O \tag{4}$$

With N_O such measures, we can calculate the mean and the standard deviation over the goodness of fit values of all observables:

$$\mu(\hat{\boldsymbol{p}}_{\boldsymbol{w}}) = \frac{1}{N_O} \sum_{O=1}^{N_O} \nu_O(\hat{\boldsymbol{p}}_{\boldsymbol{w}}) \tag{5}$$

and

$$s^2(\hat{\boldsymbol{p}}_{\boldsymbol{w}}) = \frac{1}{N_O} \sum_{O=1}^{N_O} \left[\nu_O(\hat{\boldsymbol{p}}_{\boldsymbol{w}}) - \mu(\hat{\boldsymbol{p}}_{\boldsymbol{w}})\right]^2. \tag{6}$$

We then define the objective function of the outer optimization problem as:

$$g(\boldsymbol{w}, \hat{\boldsymbol{p}}_{\boldsymbol{w}}) = \mu(\hat{\boldsymbol{p}}_{\boldsymbol{w}}) + \lambda s^2(\hat{\boldsymbol{p}}_{\boldsymbol{w}}), \tag{7}$$

where the dependency on the weights $\boldsymbol{w}$ arises from determining $\hat{\boldsymbol{p}}_{\boldsymbol{w}}$.

The minimization of the outer objective in (7) is done utilizing a radial basis function (RBF) which after a training phase will iteratively suggest new points in the weight-space until certain termination criteria of the algorithm are met. We notice that the results have a dependence on the initial guess, which is why we apply a multi-start strategy. Although not ideal this is certainly defensible in the light of the fast evaluation of the surrogates and therefore quick convergence of the inner optimization ((3)). In figure 1 we investigate the behaviour of the *outer* optimization, showing the portfolio objective as a function of the number of iterations in the algorithm. We typically find the program to terminate after $O(100)$ iterations per initial set of weights.

Example

As a first example, we are looking at a typical tuning problem, namely the so-called underlying event (UE). This effect, which is present at any hadron-collider data is modelled using various parameters, steering such things as the hadronic matter-overlap, an energy evolution of the phenomenon as well as a necessary low-energy cut-off. Many dedicated measurements exist that are designed to be particularly sensitive to UE physic modelling. We have chosen the data presented in [4]. The modelling in Pythia8, which we shall use as our physics simulator, when compared to the data shows some level of discrepancy. Data at different collider energies can not necessarily be described with the same modelling parameters. Further, there is some tension between the number and energy of particles generated in the simulation. In the plots shown in figure 2 we compare our results obtained with the bi-level optimization to the standard tuning approach. We would argue that in the presence of mis-modelling, the bi-level optimization is able to yield a balanced prediction without human intervention.

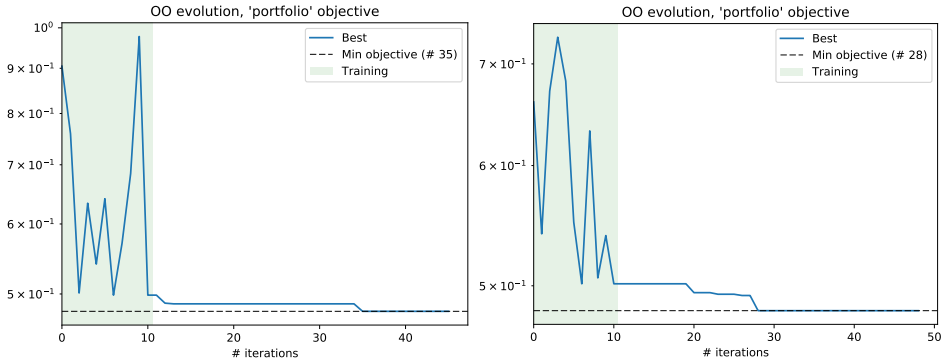

Figure 1. Performance of the radial-basis function bases optimization for two different initial conditions. The training phases is the shaded area, the dashed horizontal line indicates the minimum objective found. The blue line shows the smallest known objective after each iteration.

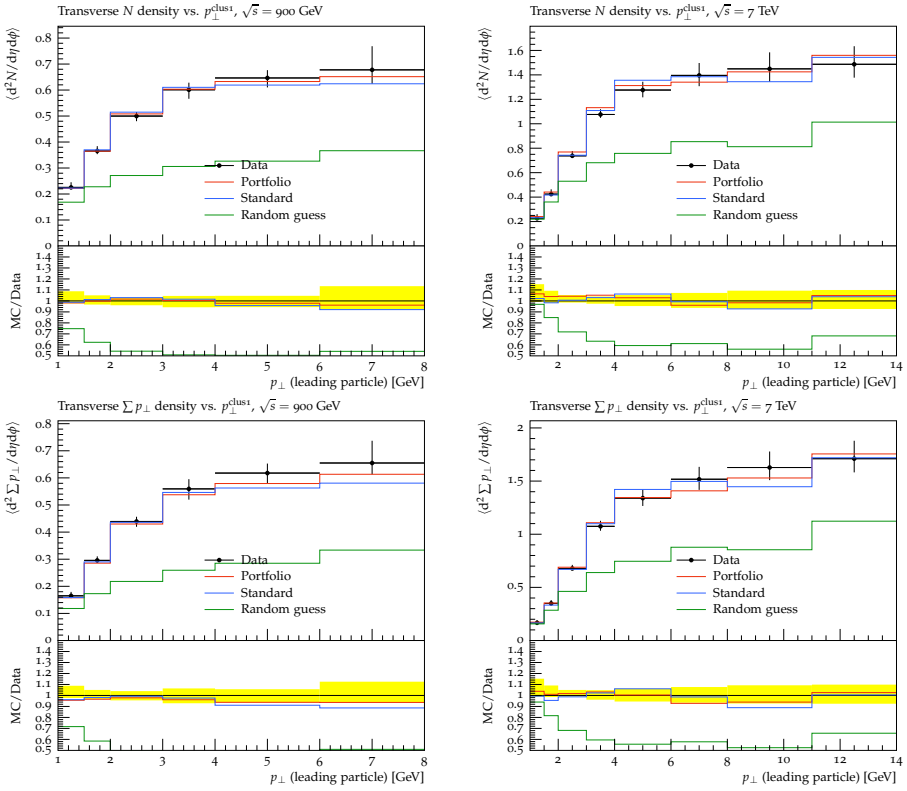

Figure 2. Comparison of results obtained with the bi-level optimization and the standard approach. The physics modelling in Pythia8 is such that there is some level of incompatibility from left to right as well as top to bottom in those 4 plots, meaning that one can get a very good description for each of the distributions alone — but not together. We would argue that the bi-level optimization with the portfolio objective is able to yield a more balanced prediction of these data-sets.

3 Rational approximations

Rational approximations can be seen as natural extensions of the polynomial approximations used so far in PROFESSOR. Denoting the degrees of the numerator and denominator polynomial as m and n respectively, we can formally define the rational approximation as

$$f^{(m,n)}(p) = \frac{h^{(m)}(p)}{g^{(n)}(p)}. \tag{8}$$

Polynomial approximations in this picture are thus simply the class of rational approximations with $n = 0$.

3.1 The algorithm

For simplicity, we present the algorithm for a one-dimensional parameter space. We are interested in deriving a rational approximation:

$$f : \mathbb{R} \to \mathbb{R}, \quad f(p) \simeq \frac{h(p)}{g(p)} \tag{9}$$

by fitting the coefficients of $h(p), g(p)$ to data, $(p_l, f(p_l)$. Denoting the polynomials as

$$h(p) = a_0 + a_1 p + a_2 p^2 + \ldots + a_m p^m \quad \text{and} \quad g(p) = b_0 + b_1 p + b_2 p^2 + \ldots + b_n p^n, \tag{10}$$

we observe that we have $K = (m+1)+(n+1)-1$ degrees of freedom (one less, because we can scale either $a_0 = 1$ or $b_0 = 1$). Assuming that we have K data points, $(p_l, f(p_l)), l = 1, \ldots, K$ that are "nicely" chosen, the fitting problem can be written as

$$f(p_l) = \frac{h(p_l)}{g(p_l)} \quad \Leftrightarrow \quad g(p_l) f(p_l) = h(p_l) \quad \forall l = 1, \ldots, K \tag{11}$$

provided that $g(p_l) \neq 0$. We note, that (11) is a square linear system of equations in K unknowns. To solve this system, we provisionally add coefficients up to order K to $h(p)$ and $g(p)$ (which we will enforce to be zero later). Thus, we write the polynomials as

$$\begin{array}{rcl} h(p) & = & a_0 + a_1 p + a_2 p^2 + \ldots + a_m p^m + \ldots + a_K p^K \\ \text{and} & & \\ g(p) & = & b_0 + b_1 p + b_2 p^2 + \ldots + b_n p^n + \ldots + b_K p^K. \end{array} \tag{12}$$

We let the Vandermonde matrix of order K be

$$V := \begin{bmatrix} 1 & p_1 & p_1^2 & \cdots & p_1^K \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & p_K & p_K^2 & \cdots & p_K^K \end{bmatrix} \tag{13}$$

and we define the diagonal matrix $F := \mathrm{diag}(f(p_1), \ldots, f(x_K))$, and vectors of coefficients $a = (a_0, \ldots, a_K)^T$ and $b = (b_0, \ldots, b_K)^T$. Then, we can write (11) compactly as

$$FVb = Va.$$

If we assume that V is invertible (which imposes a condition on the interpolation points), then we can rewrite this system as

$$a = V^{-1} FVb =: Zb,$$

where $Z = V^{-1}FV$. Now recall, that $b_{n+1} = \ldots = b_K = 0$, and denote the first n components of b by $\hat{b}$, and similarly define $\hat{Z} := Z[:, 1:n]$ using Matlab/Python notation. Then it follows that

$$a = \hat{Z}\hat{b},$$

which is a "skinny" system of equations. If we also enforce the condition that $a_{m+1} = \ldots = a_K = 0$, then we can write this system as

$$\left(\begin{array}{c} a_0 \\ \vdots \\ a_m \\ \hline a_{m+1} \\ \vdots \\ a_K \end{array}\right) = \left[\begin{array}{c} \bar{Z} \\ \hline \tilde{Z} \end{array}\right] \left(\begin{array}{c} b_0 \\ \vdots \\ b_n \end{array}\right) \quad \text{or} \quad \hat{a} = \bar{Z}\hat{b}, \quad \text{and} \quad 0 = \tilde{Z}\hat{b}$$

We now observe that the last set of equations, $0 = \tilde{Z}\hat{b}$, has $K - m - 1 = m + n + 1 - m - 1 = n$ equations and $n + 1$ unknowns. Thus, if the equations are consistent, we can choose $\hat{b}$ to be a vector that lies in the null-space of $\tilde{Z}$. Forming an SVD of $\tilde{Z}$

$$\tilde{Z} = \tilde{U}\tilde{\Sigma}\tilde{V}^T,$$

we choose $\hat{b} := \tilde{V}[:, n+1]$ as the last singular vector, then obtain $\hat{a} = \bar{Z}\hat{b}$.

The generalization to multi-dimensional parameter spaces is straight forward.

3.2 Example

We consider the function

$$f(x) = \frac{1+x}{1+x+x^2}, \quad x \in (0, 100).$$

The calculation of a rational approximation of type $(m = 1, n = 2)$ requires the determination of $N_{\text{coeff}} = 6$ coefficients, which is the same number of coefficients required to build a 5^{th} order polynomial in 1 dimension. Both approximations to the same input data are shown in Fig. 3. The qualitative gain when using rational approximations is quite evident. Rather importantly for physics use-cases, where purely positive definite quantities are encountered frequently, we observe that the polynomial approximations have a tendency to oscillate, leading to negative predictions which can be nonphysical therefore making the application of the latter at least questionable.

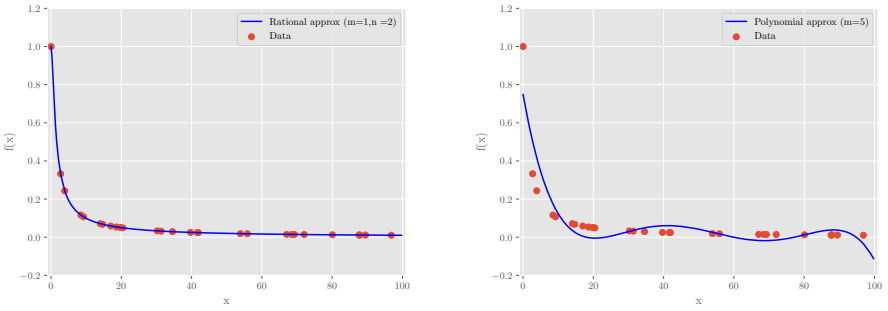

Figure 3. Comparison of rational and polynomial approximation to input data that exhibits traits of a rational function. Left: rational approximation with 6 coefficients. Right: polynomial approximation with 6 coefficients.

4 Summary and outlook

We presented recent progress in the methodology of of Monte-Carlo event-generator tuning. A bi-level optimization strategy seems promising toward more automatic event-generator tuning, especially in the presence of mis-modelling or otherwise existing tension between data-sets. The extension of the surrogate models from ordinary polynomials to rational approximations looks promising and is expected to widen the applicability of the tuning method. The application to multi-dimensional parameter spaces will be thoroughly discussed in an upcoming publication.

References

[1] A. Buckley, H. Hoeth, H. Lacker, H. Schulz, J.E. von Seggern, Eur. Phys. J. **C65**, 331 (2010), 0907.2973

[2] T. Sjostrand, S. Mrenna, P.Z. Skands, Comput.Phys.Commun. **178**, 852 (2008), 0710.3820

[3] A. Buckley, J. Butterworth, L. Lonnblad, D. Grellscheid, H. Hoeth, J. Monk, H. Schulz, F. Siegert, Comput. Phys. Commun. **184**, 2803 (2013), 1003.0694

[4] G. Aad et al. (ATLAS), Eur. Phys. J. **C71**, 1636 (2011), 1103.1816

Acknowledgement

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and Office of HEP, Scientific Discovery through Advanced Computing (SciDAC) program.