HEP Simulation issues

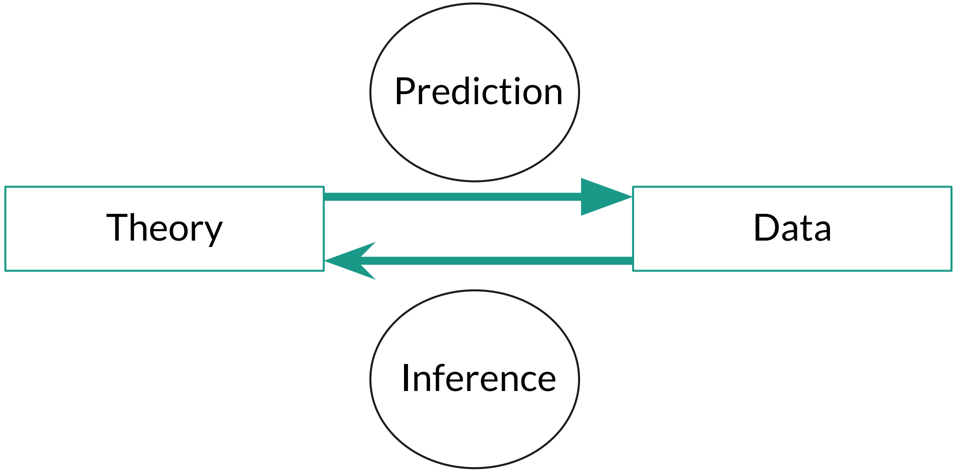
Jožef Stefan Institute

Manuel Szewc
IJS
“A Deep Learning Era of Particle Theory”
MITP, 01/07/2022

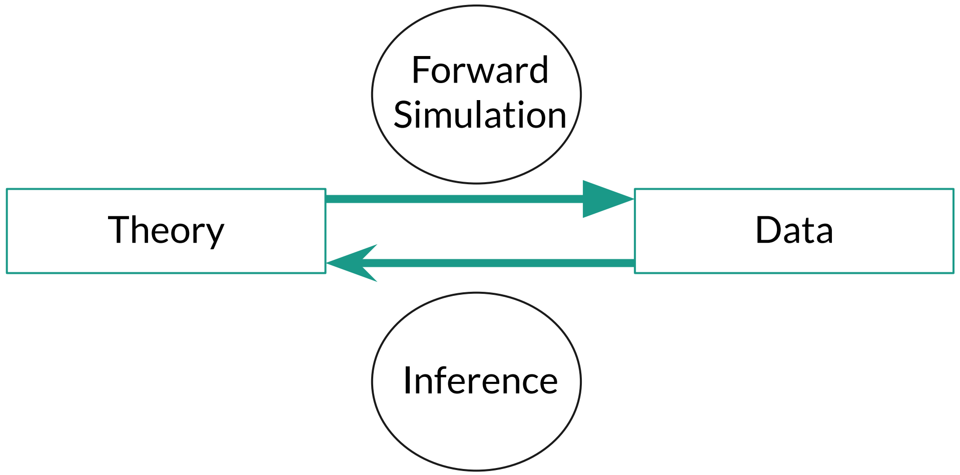
In this talk

- Why we need Event generators
- What are the problems with simulations?
- What are the best ways to tackle those problems?

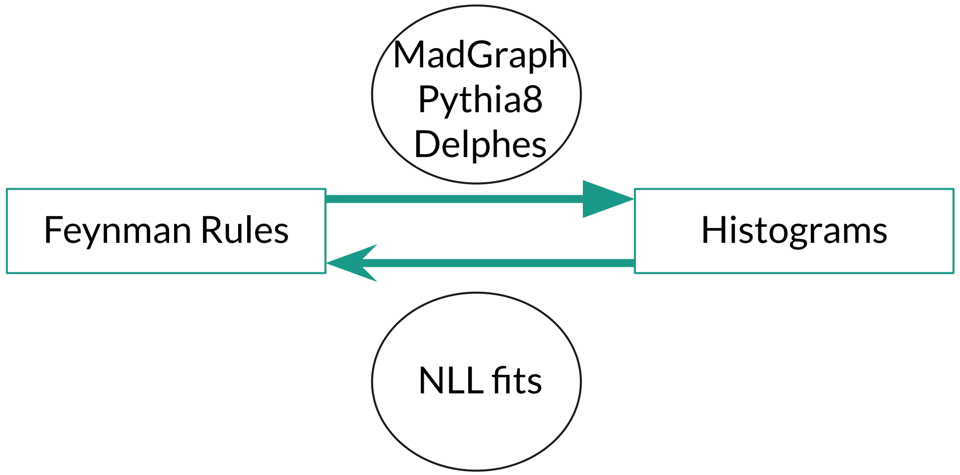
Why we need Event generators



Why we need Event generators



An example


MadGraph
Pythia8
Delphes
Feynman Rules
Histograms
NLL fits

Specific needs from event generators

- Physical simulations: we need the simulated events to follow as close as possible the modelling hypothesis (e.g. QFT predictions at a particle collider from Standard Model Lagrangian)
- Speed: We need simulations to be reasonably fast
- Size: Ideally they should be storable and shareable

Focusing on collider experiments

Event generators need to consider many different physical phenomena, with different scales and different tools:

- Partons originating from colliding hadrons sampled through PDFs
- Hard scattering amplitudes calculation and phase space integration
- ISR/FSR
- Hadronization
- Final state interactions
- Underlying event effects
- Detector effects

Making use of factorization theorems, different dedicated softwares have achieved incredible sophistication but still face difficulties

What are the problems with simulations?

High-dimensional parameter space that models empirically different non-perturbative effects plus assume exact factorization theorems:

- Expensive tunes
- Expensive and difficult treatment of uncertainties
- Additional systematics due to modelling
- Computational bottlenecks
- Numerical instabilities
- Cross-cutting from factorization theorems' breakdowns
- Self-consistency of parameter tunes

In particular, hadronization

Inherently non perturbative process: phenomenological models needed, no ab initio calculations

Two main models: Lund String model and the clustering model

These models are impressive, highly tuned, but can struggle to describe consistently and appropriately different phenomena

(Some) Hadronization modelling issues

O(20%) to O(50%) discrepancies between proton-proton and ion-ion collisions

Heavy particle composition as a function of event multiplicity is mismodelled at high event multiplicities

Mismodelling of the mass dependence of the average transverse momentum

Minimum bias description can be incompatible because of low transverse momentum mismodelling

Ridge in pp collisions missing in Pythia (and in general long range correlations are hard to model)

Charged particle multiplicity spectrum is very sensitive to color reconnections and MPI modelling

What can ML offer here?

ML is already helping a lot!

In particular, by reducing computational bottlenecks by replacing different steps of the machinery with surrogate models. See for example (there are many, many others):

- MLHAD / HADML for Pythia8 / Herwig surrogate models
- CaloGAN, CaloFlow for Geant4 calorimeter surrogate models
- OTUS, DijetGAN for End-to-end surrogate models

What can ML offer here?

Surrogate models can make for a much easier parameter inference and unfolding.

See for example:

- MLPF for particle reconstruction from calorimeter and trackers
- OmniFold and cINN for unfolding
- MadMiner and the Matrix Element Method for parameter inference

What can ML offer here?

ML opens the door to learning hadronization from data in a more direct way. To do that, we need

- Flexibility: we want to capture very complicated physics
- Efficiency: we want the procedure to be as fast as possible so learning and optimizing is possible

So what should we do? Surrogate models? Differentiable programming? Where should we train? What are the relevant metrics?

MLHAD for Pythia

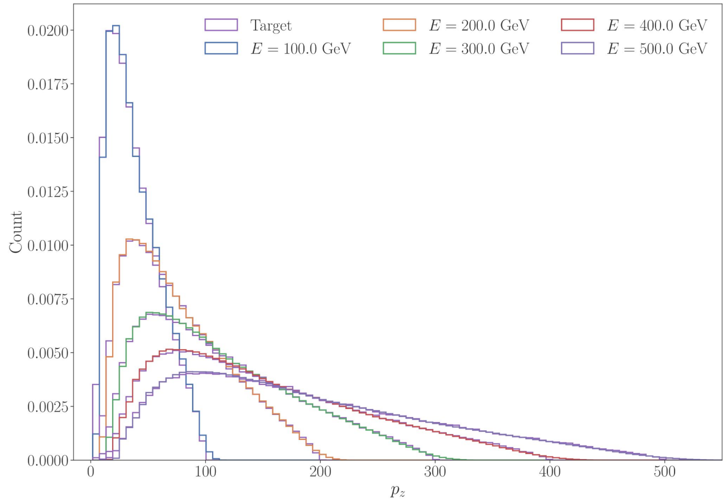
arxiv:2203.04983 by P. Ilten, T. Menzo, A. Youssef, J. Zupan

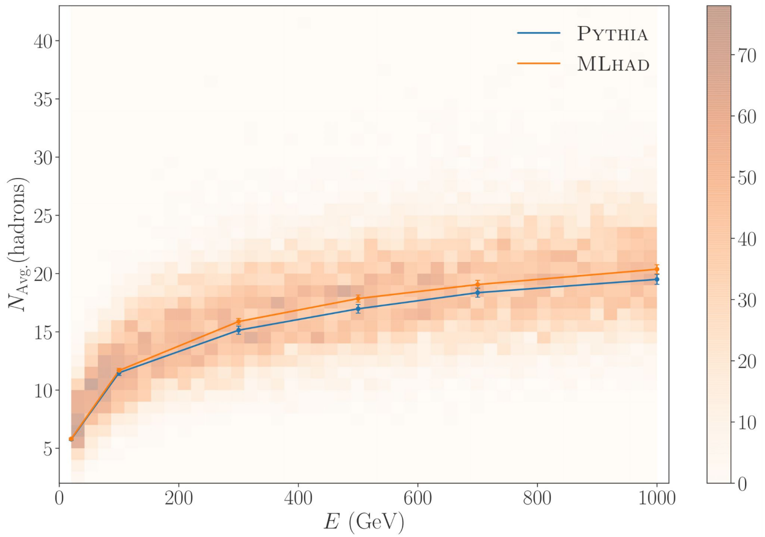
A first step towards using surrogate models for hadronization.

Learn first hadronization probability distribution from Pythia using a cSWAE

Will hear more about this next week

MLHAD for Pythia





Many issues to sort in general

Uncertainty quantification is still an issue. Also incorporation of the full set of systematic uncertainties

Surrogate training time

Trustworthiness needs to be assessed. Should we pay more attention to interpretability?

What is the best way to take advantage of current state of the art event generators when incorporating ML?

What do I mean by interpretability?

Suppose we have a very good surrogate model for a given dataset.

How do we know how well it translates to real data beyond usual cross-validation checks?

How do we know if any new tunes make sense?

One has to remember we are interested in obtaining statistical statements regarding real, underlying objects. Not only in training metrics.

Bayesian techniques

They already permeate event generators and their surrogate models extensions: NNPDFs, Bayesian Neural Networks, cINN surrogate models

Bayesian inference allows for a better evaluation of uncertainties and for improved unfolding (as long as one can make probabilistic statements)

Intuitively, Bayesian always sounds good… Now it is becoming more possible. Differentiable programming is a key development. So are different approaches to inference like Black Box Variational Inference and simulation-based inference where the intractabilities are somewhat sidestepped.

Where could we go further?

Could we lessen the modularity assumption? Start from traditional MC approaches and treat cross-cuttings from a Bayesian perspective.

Treat our simulator as a non-parametric bayesian model? As a prior over an empirical model?

Start thinking more about marginal posteriors as swyft does for astro-cosmo. Likelihood-free inference.

What about Multilevel model emulators?

Could we implement model comparison / model combination Bayesian techniques to asses and combine different tunes and even different generators together and resolve inconsistencies?

Bayesian techniques for modelling

There is a vast array of literature about Bayesian model building we can take advantage of. Already a lot of examples in HEP (e.g. GANs, VAEs, Shower Deconstruction, LDA, Topic models in general ...)

Always need to keep in mind the same requirements as for event generators:

- Physically meaningful: harder to achieve than with event generators. Requires physical bias to be baked into the model.
- Speed + size: training should not be too hard nor require so much data as to render simulations preferable

Learning four tops

arxiv:2107.00668, E. Alvarez, B.M. Dillon, D. A. Faroughy, J.F. Kamenik, F. Lamagna, MS

We can resume our goal as the following:

Consider our imperfect simulations of the (known) physical processes for four-tops as **prior knowledge** and **update** our knowledge using the **measured data** $\rightarrow$ Bayesian framework

Putting it all together: Generative process

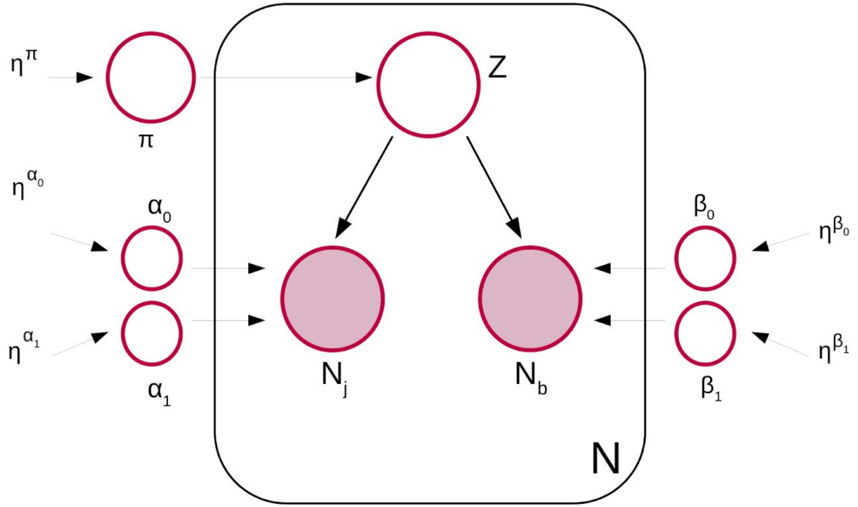
Sample fractions $\pi_0, \pi_1 \sim \text{Dir}(\eta^{\pi})$

For t=1,2:

- Sample light jet multinomials $\alpha_t \sim \text{Dir}(\eta^{\alpha t})$
- Sample b-jet $\beta_t \sim \text{Dir}(\eta^{\beta t})$

For event n=1,...,N:

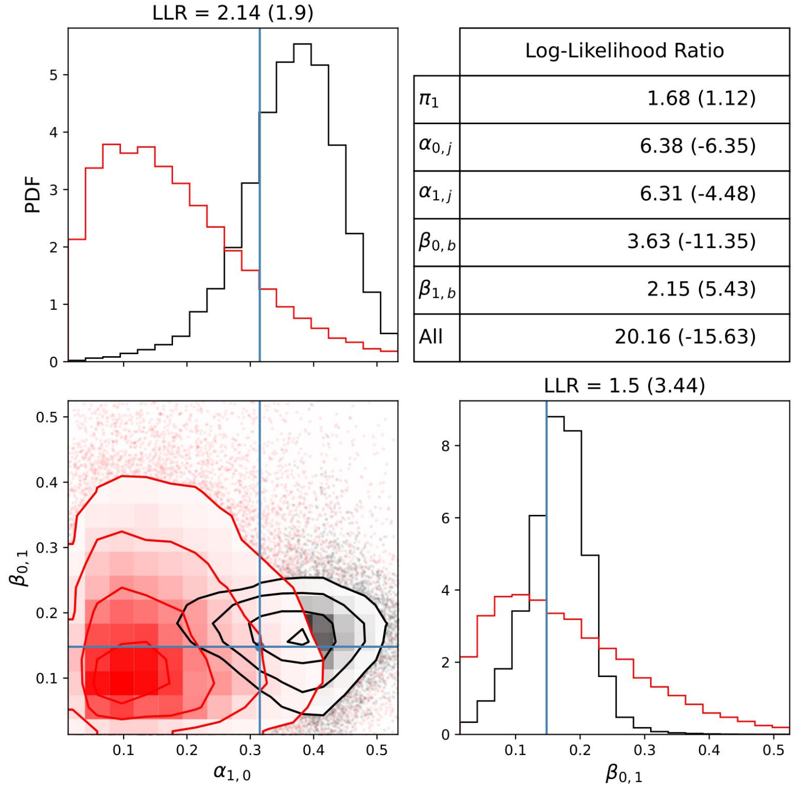
- Sample event assignment $z_n \sim \text{Multi}(\pi_0, \pi_1)$
- Sample $j_n \sim \text{Multi}(\alpha_{zn})$
- Sample $b_n \sim \text{Multi}(\beta_{zn})$



Corner plots

For each parameter, we show its 1d histogram and its 2d correlations with every other parameter

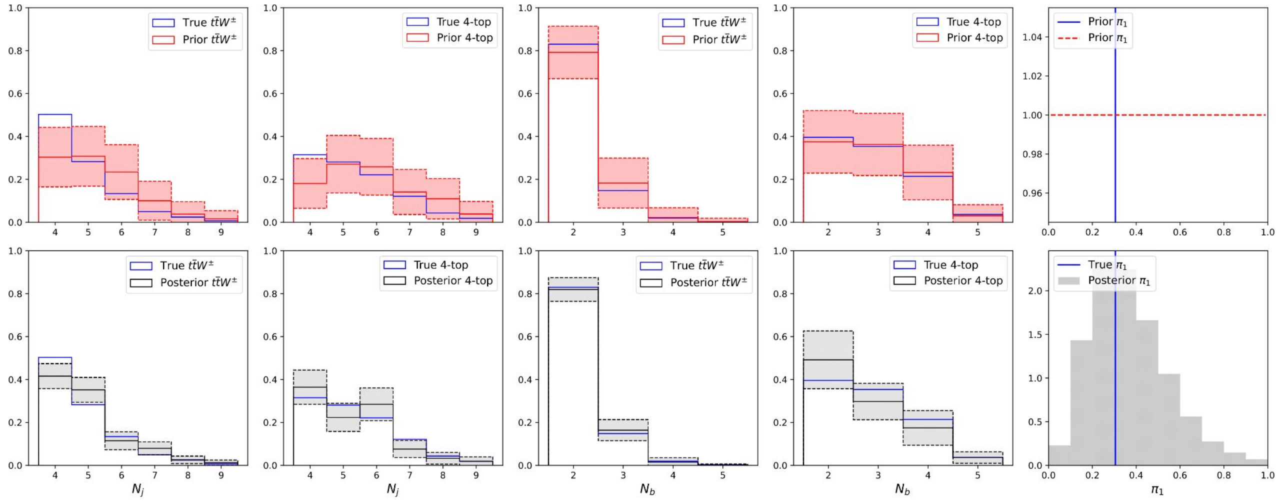
This is a particularly good example where we capture correlations



	Log-Likelihood Ratio
π_1	1.68 (1.12)
$\alpha_{0,j}$	6.38 (-6.35)
$\alpha_{1,j}$	6.31 (-4.48)
$\beta_{0,b}$	3.63 (-11.35)
$\beta_{1,b}$	2.15 (5.43)
All	20.16 (-15.63)

Grouping parameters together

Now we can compare prior vs posterior



All in all

State of the art measurements need state of the art predictions

To better our predictions, already a vast literature of ML applications

What’s the best way to validate said applications?

Thank you!