Tuning of Merged Pythia

Leif Gellersen

PhD Student at Lund University

leif.gellersen@thep.lu.se

January 23rd, 2019

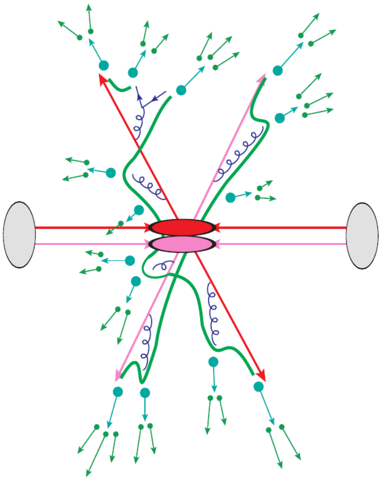
Overview

MC Event Generators

Predict fully exclusive final state

- Perturbative methods well known
  - Hard interaction: Matrix elements (LO/NLO)
  - Radiative Corrections: Parton shower in initial and final state
- Non-perturbative models
  - Multiple interactions
  - Hadronization
  - Hadron decays

Models include many parameters

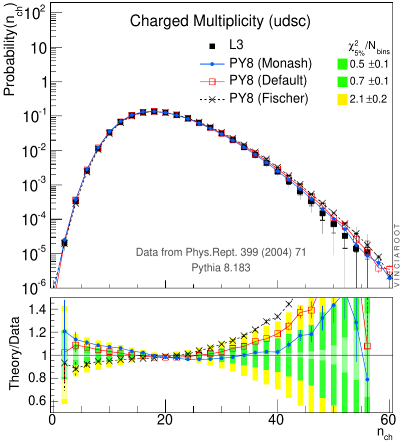


Borrowed from S. Prestel

Tuning: General Idea

- Optimize parameters based on well-measured data
- Factorize as much as possible (assuming universality)

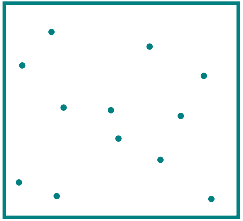
FSR	e^+e^- data: LEP event shapes
Hadronization	Many parameters, model dependent. Use LEP identified particle spectra
ISR and UE	Use hadron collider data

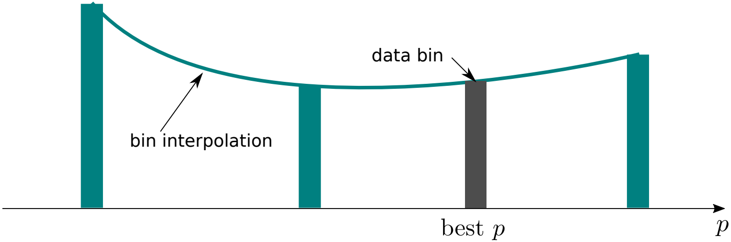


arXiv:1404.5630, P. Skands et al., 2014

How to Tune

- Generate MC pseudodata $f_b(\vec{p})$, compare to experimental data bin $\mathcal{R}_b$
- Iterative MC event generation slow $\rightarrow$ Use bin-wise parametrization of MC generator response





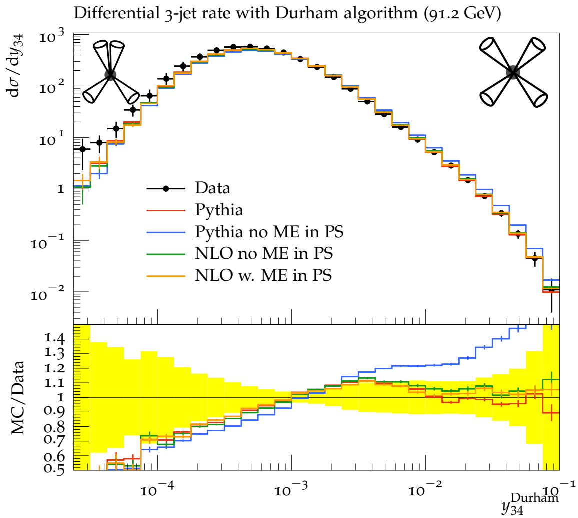
- Minimize $\chi^2(\vec{p}) = \sum_b w_b \frac{(f^{(b)}(\vec{p}) - \mathcal{R}_b)^2}{\Delta_b^2}$, with data uncertainty Δ_b, bin weights w_b
- Professor: Python package for MC tuning, highly automated, includes validation tools
  arXiv:0907.2973, A. Buckley et al., 2009

Pythia Matching & Merging Tune work with Stefan Prestel and Malin Sjödahl

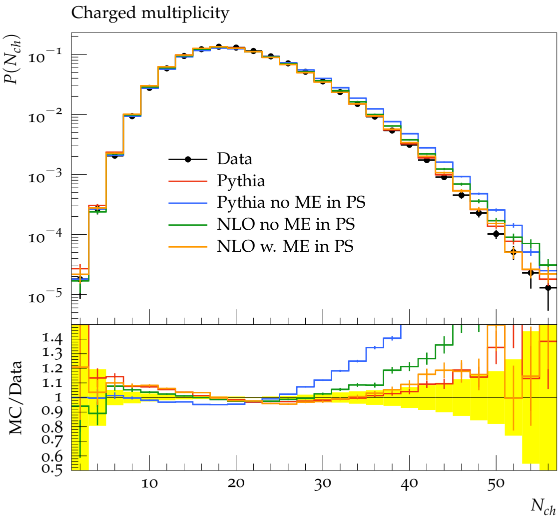
- Matching and Merging: use additional pQCD input

  Multi-jet Merging Higher multiplicity matrix elements → improved radiation pattern

  NLO Matching NLO matrix elements → higher fixed order accuracy
- Problem: Monash tune based on LO matrix elements with ME corrections
- Good for many observables in matched & merged calculations
- More precise perturbative calculation → less freedom to tune
- Retuning might allow for improvements, more universal tune



Data from Jade & Opal Collaborations, 2000
arXiv:hep-ex/0001055



Data from L3 Collaboration, 2004
arXiv:hep-ex/0406049

Scale Variations in UMEPS Merging

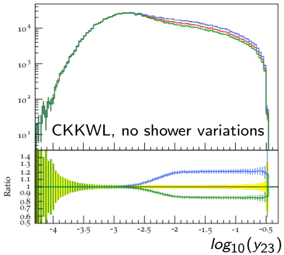
In unitarized multi-jet merging, observables $\mathcal{O}$ are calculated by

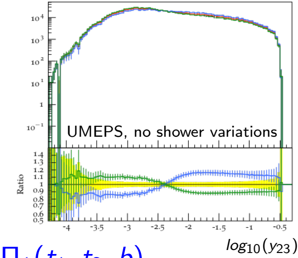
$$\langle\mathcal{O}\rangle = \int d\phi_0 \left\{ \mathcal{O}_0 \left[B_0 - \int_s B_{1\to 0} w_1 - \int_s B_{2\to 0} w_2 \right] + \int d\phi_1 \mathcal{O}_1 \left[B_1 w_1 - \int_s B_{2\to 1} w_2 \right] + \int d\phi_1 \int d\phi_2 \mathcal{O}_2 B_2 w_2 \right\}$$

with weights

$$w_1 = \frac{\alpha_s(bt_1)}{\alpha_s(\mu_R)} \Pi_0(t_0, t_1, b) \quad \text{and} \quad w_2 = \frac{\alpha_s(bt_1)}{\alpha_s(\mu_R)} \frac{\alpha_s(bt_2)}{\alpha_s(\mu_R)} \Pi_0(t_0, t_1, b) \Pi_1(t_1, t_2, b)$$

Vary $b \to b/2, 2b$ in weights and in trial shower generating Sudakovs Π_i





Scale Variations in UNLOPS

UNLOPS: Combine NLO matching and UMEPS merging:

$$\begin{aligned}\langle\mathcal{O}\rangle = \int d\phi_0 \Big\{ \mathcal{O}_0 \Big[\bar{B}_0 - \int_s \bar{B}_{1\to 0} - \int_s B_{1\to 0}(w_1 - w_1|_{\mathcal{O}(\alpha_s)}) - \int_s B_{2\to 0} w_2 \Big] \\ + \int d\phi_1 \mathcal{O}_1 \Big[\bar{B}_1 + B_1(w_1 - w_1|_{\mathcal{O}(\alpha_s)}) - \int_s B_{2\to 1} w_2 \Big] \\ + \int d\phi_1 \int d\phi_2 \mathcal{O}_2 B_2 w_2 \Big\}\end{aligned}$$

with weights w_1, w_2 as before, but $\bar{B}_i$ come with variations as well

AutoTunes work with Johannes Bellm

Problem

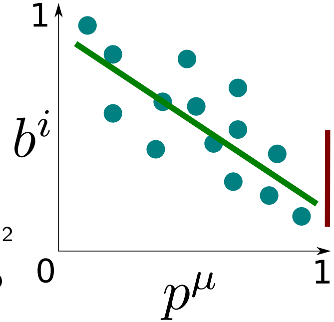
- Polynomial interpolation only possible for $\lesssim 10$ parameters
- Interpolation only good if ranges small enough
- χ^2 depends on weights $\rightarrow$ need to know data and generator

Goal

- Framework to reduce human interaction & make tune reproducible
- Tune many parameters at once: automatically divide into sub-tunes
- Set weights for observables automatically
- Allow for iterations with revised parameter ranges

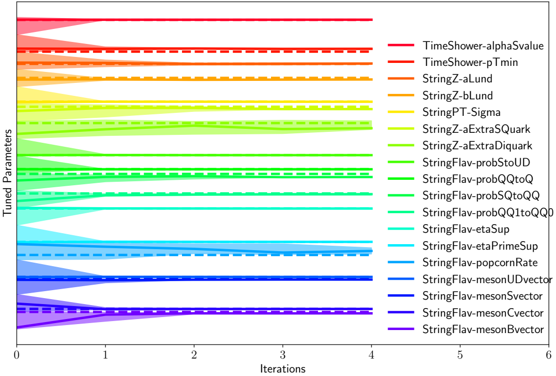
AutoTunes: The Idea

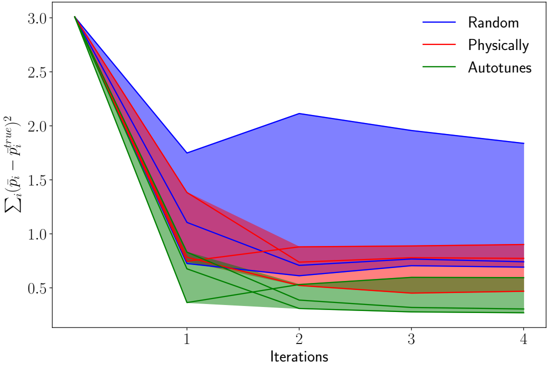
- Normalize each bin b_i and each parameter p^μ to $[0, 1]$
- Find slopes S_i^μ
- $\vec{S}_i$ vector in parameter space
- $\vec{S}_i$ points along parameters of high influence on bin
- Normalize: $N_i^\mu = \frac{S_i^\mu}{\sum_i S_i^\mu}$
- Find $\vec{J} = (1, 0, 0, 1, 0, \ldots, 1)$ that maximizes $w = \sum_i (\vec{N}_i \cdot \vec{J})^2$
  $\rightarrow$ "Most correlated" subset of parameters: tune in one step
- Use weights $w_i = \vec{N}_i \cdot \vec{J}$, emphasizes relevant data bins



Iterative Pythia Tune to Pythia Pseudodata

Try to reproduce - - - values, ≈ 6000 DOF & 18 parameters





Summary

- Matching and Merging requires tune validation and retuning of soft physics models
- Scale variations can identify well constrained hard regions of phase space
- Working on AutoTunes: Framework for more automated tuning with many parameters