Efficiency improvements in MC event generation

Stefan Höche[1]

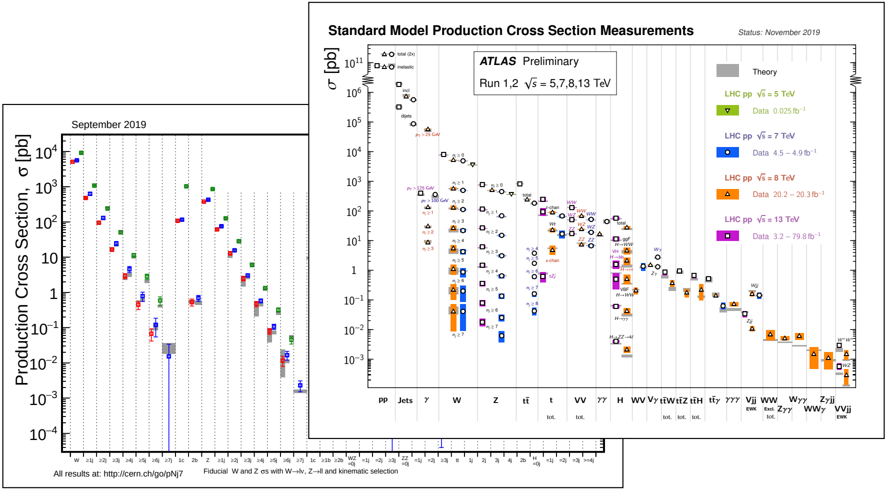
Fermi National Accelerator Laboratory

JLab Computing Round Table

Newport News, 05/05/2020

[1] based on work with Christina Gao, Joshua Isaacson, Claudius Krause, Stefan Prestel and Holger Schulz

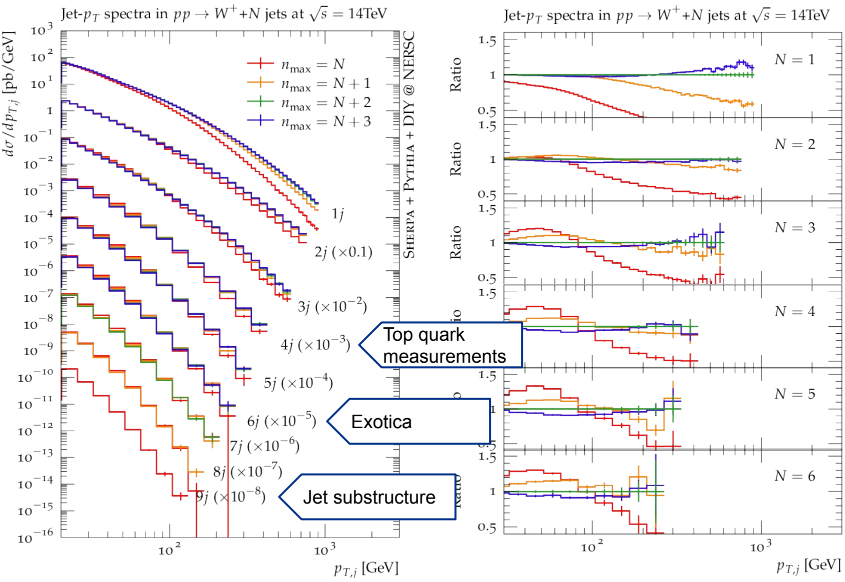
What do we need to compute?



[ATLAS] https://twiki.cern.ch/twiki/bin/view/AtlasPublic/StandardModelPublicResults

[CMS] https://twiki.cern.ch/twiki/bin/view/CMSPublic/PhysicsResultsCombined

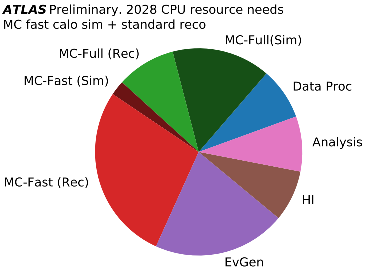
What is our pain threshold?

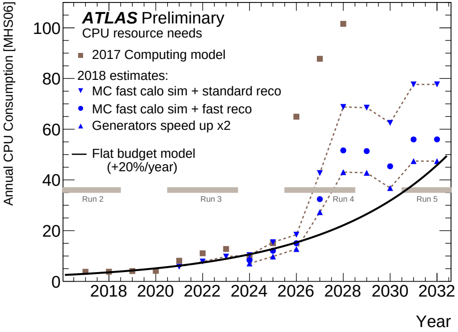


Why improve generation efficiency?

[HSF Generator WG] arXiv:2001.10028

- Event generation will consume significant fraction of resources at LHC soon
- Need to scrutinize both generator usage and underlying algorithms
- Dedicated effort in HEP Software Foundation (HSF)



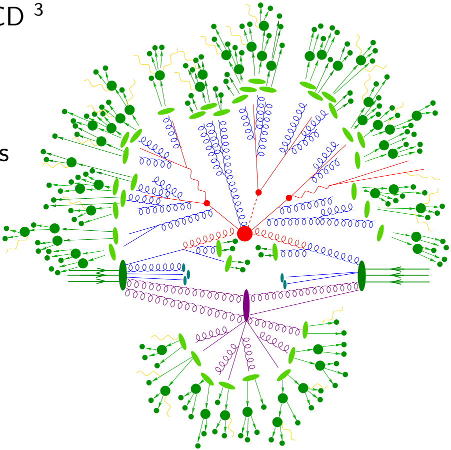


[ATLAS] https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ComputingandSoftwarePublicResults

How does the calculation work?

- **Hard interaction**
  LO, NLO QCD/EW [2], NNLO QCD [3]
  Various ME generators . . .
- **Radiative corrections**
  Parton / Dipole showers,
  YFS resummation / QED showers
- **Multiple interactions**
  Sjöstrand-Zijl / eikonal models
- **Hadronization**
  String / Cluster model
- **Hadron Decays**
  Phase space or EFTs,
  YFS QED corrections

[Buckley et al.] arXiv:1101.2599

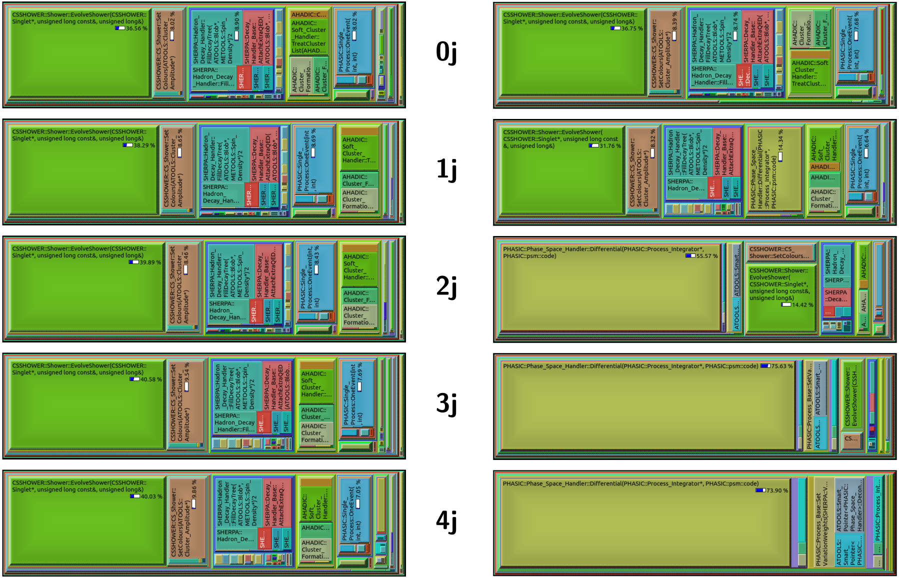


[2] via interfaces to loop generators
[3] $pp \to Z/W^{\pm}/h/W^{\pm}W^{\mp}$

Timing distribution: LO merging example

Weighted W+0..≤4j@LO

Unweighted W+≤0..4j@LO

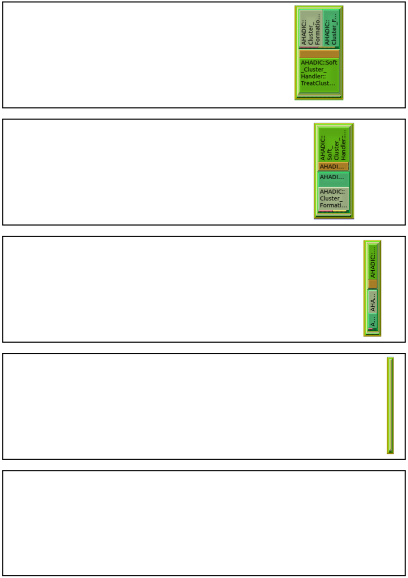


Timing distribution: LO merging example

Weighted W+0..≤4j@LO

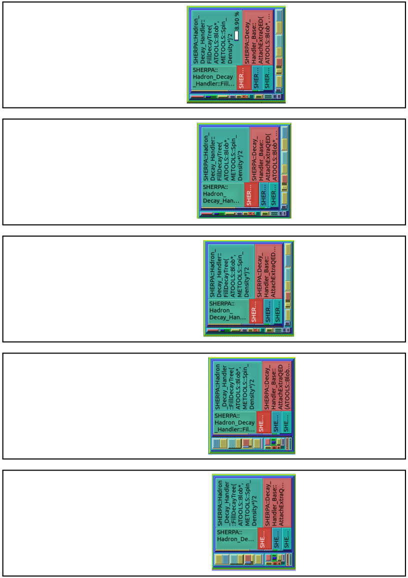
Hadronization

Unweighted W+≤0..4j@LO



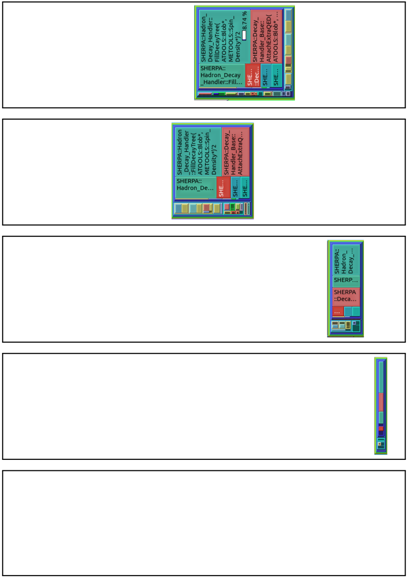
Timing distribution: LO merging example

Weighted W+0..≤4j@LO



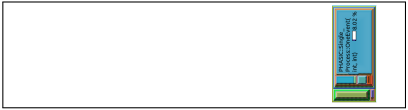
Hadron decays + QED radiation

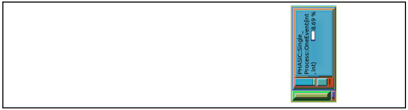
Unweighted W+≤0..4j@LO

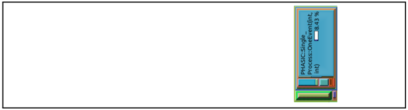


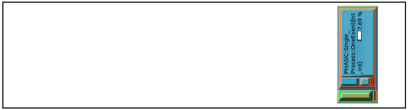
Timing distribution: LO merging example

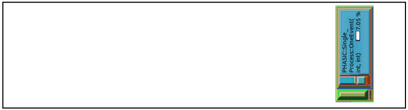
Weighted W+0..≤4j@LO





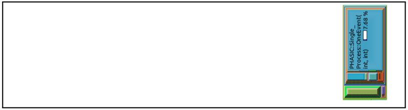


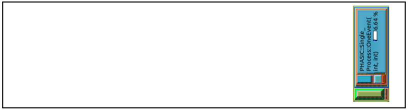


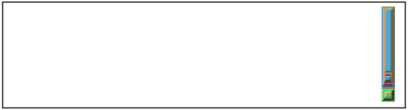


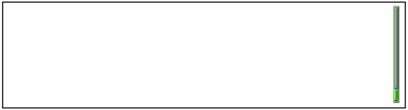
Underlying event simulation

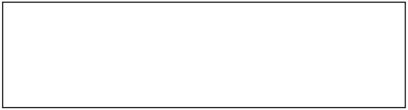
Unweighted W+≤0..4j@LO





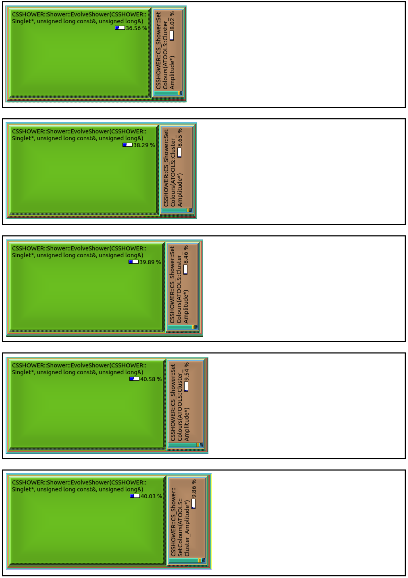






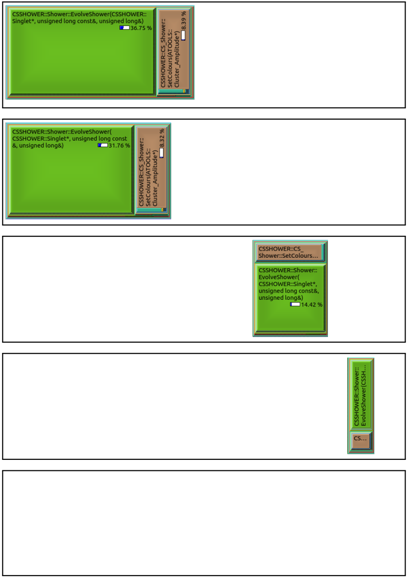
Timing distribution: LO merging example

Weighted W+0..≤4j@LO



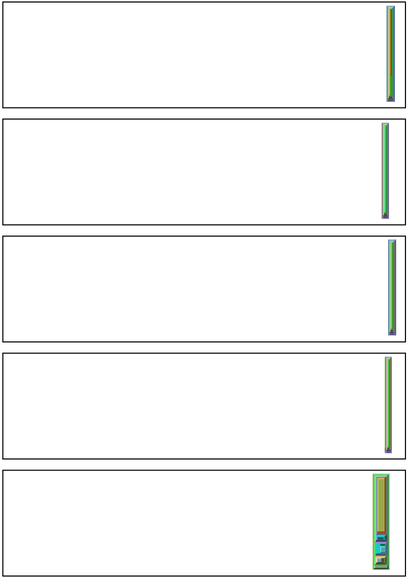
Parton Shower evolution

Unweighted W+≤0..4j@LO



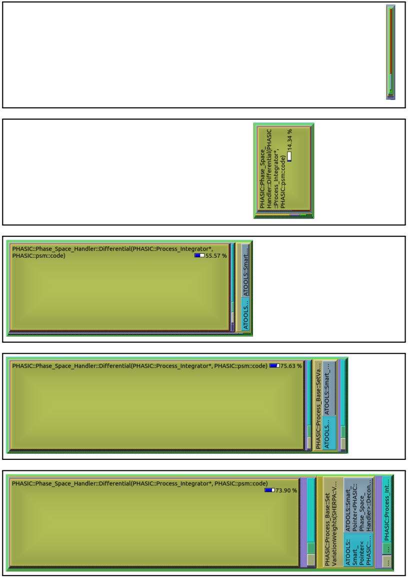
Timing distribution: LO merging example

Weighted W+0..≤4j@LO



Matrix Element generation

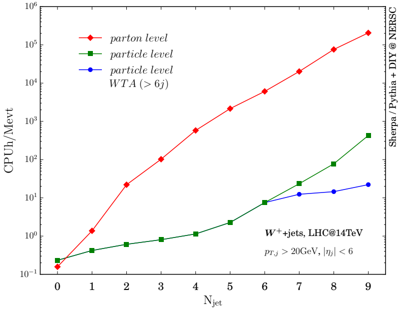
Unweighted W+≤0..4j@LO



Timing distribution: Details on scaling with multiplicity

Observation

- Hard scattering simulation much more demanding than parton shower & hadronization in multi-jet merged simulations
- Complexity of merging ME&PS grows quickly due to inherent N! scaling of algorithms, but still negligible compared to hard scattering calculation



Where does this hierarchy come from?

- Best case ME computation naively scales as $\approx \mathcal{O}(3^N)$
- Monte-Carlo unweighting efficiency degrades quickly due to high dimensionality of phase-space integral (3N-4)
- Overall scaling $\approx \mathcal{O}(4^N)$ if not taking cuts into account
- Cut inefficiencies worsen the picture significantly

The problem at matrix-element level

- We want to compute the integral

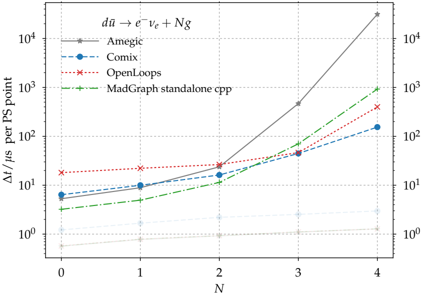
$$\langle O \rangle = \sum_{a,b \in \{q,g\}} \int dx_1 \int dx_2 f_a(x_1, Q^2) f_b(x_2, Q^2)\, \hat{\sigma}_{ab}(Q^2)\, O$$

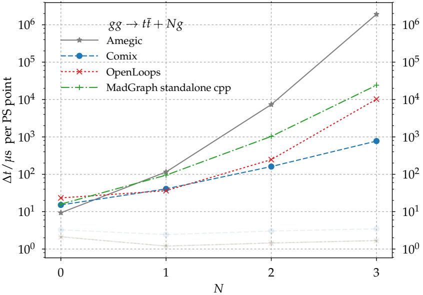
  where O is our observable, and $\hat{\sigma}$ is the partonic cross section

$$\hat{\sigma} = \frac{1}{SF} \int d\Phi_{2\to n}\, |M_{2\to n}|^2$$

- $d\Phi_{2\to n}$ - differential phase-space element: **3n-4 dimensional**
- $|M_{2\to n}|^2$ - matrix element squared: **exhibits poles and strong peaks**
- Cuts to remove singularities create **discontinuous integrand**
- Need adaptive MC methods to perform computation
- During adaptation stage, find maximum $\langle O \rangle$
  During event generation stage, hit-or-miss against max

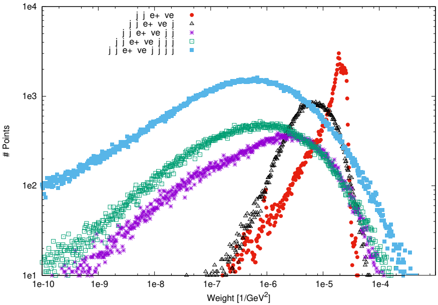
Timing of integrand





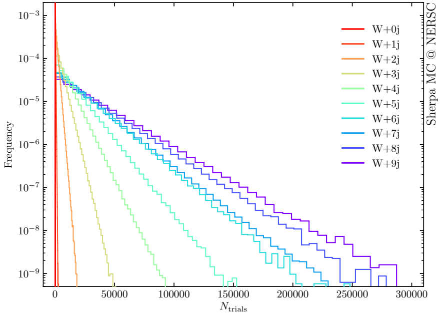
- Amegic hep-ph/0109036 $\to$ Feynman diagrams
  Worst case scaling factorial with particle multiplicity
- Comix arXiv:0808.3674 $\to$ Color-dressed recursion
  Worst case scaling exponential with particle multiplicity
- MadGraph arXiv:1405.0301 $\to$ Feynman diagrams
  Worst case scaling factorial with particle multiplicity
- OpenLoops arXiv:1907.13071 $\to$ Color-ordered recursion
  Worst case scaling $\sim$ factorial with particle multiplicity

Weight distributions and number of trials



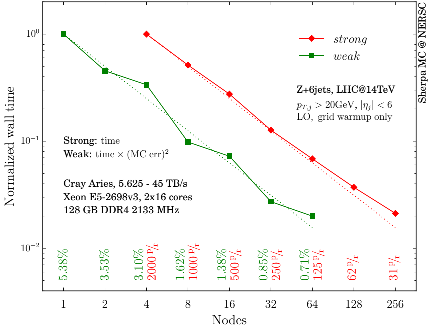
- Unweighted events obtained by performing hit-or-miss against maximum of weight distribution
- If maximum artificially low (or current weight exceeds known max) events acquire relative weight

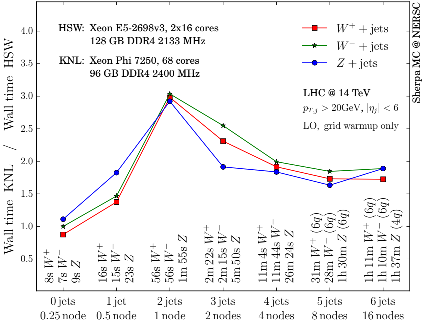
- Higher multiplicity events have broader weight distribution
- Leads to much reduced unweighting efficiency
- Clearly, if we improve this, we gain more than by just accelerating the integrand



Scaling with MPI and on different architectures

[Prestel,Schulz,SH] arXiv:1905.05120

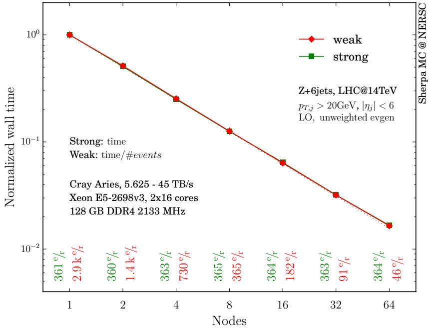


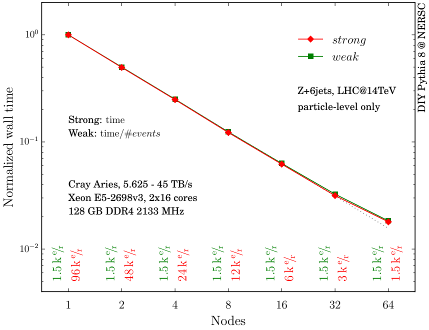


- Traditionally, integration is performed with Vegas or similar algorithms
- Grid warmup step scales (strong & weak) up to $\sim$ 2048 cores
- Performance limited by number of events being processed per rank
- Acceptable (though limited) performance on KNL w/o modifications

Scaling with MPI and on different architectures

[Prestel,Schulz,SH] arXiv:1905.05120





- Scaling of event generation step (strong & weak) up to $\sim$ 2048 cores
- Performance limited by number of events being processed per rank (Average timing can only be expected if statistics is large enough)
- Performance of particle-level simulation limited by I/O speed Good results with Cori burst buffer, but room for improvement

Improving event generation efficiency with Neural Networks

- Neural Networks can be used in different ways to improve event generation

Surrogate model technique

- Generate events with GANs
  arXiv:1707.00028, arXiv:1901.00875, arXiv:1901.05282, arXiv:1903.02433, arXiv:1907.03764, arXiv:1912.08824, arXiv:1909.01359, arXiv:1909.04451, ...
- Several orders of magnitude faster
- Generates unweighted events
- Needs existing sample to train
- Biased results if not trained right

Variabe transformation technique

- Learn integrand to improve importance sampling
  arXiv:1707.00028, arXiv:1810.11509, arXiv:2001.05478 arXiv:2001.05486
- Insufficient training yields high uncertainties, but no bias
- Events generated from scratch no pre-existing sample required
- Resulting events still need to be unweighted

Event generation with Normalizing Flows

- Straightforward MC integral estimator

$$I = \int_\Omega f(x)\,\mathrm{d}x = \frac{\Omega}{N}\sum_{i=1}^{N} f(x_i) = \Omega\,\langle f\rangle_x\ , \qquad \sigma_I = \Omega\sqrt{\frac{\langle f^2\rangle_x - \langle f\rangle_x^2}{N-1}}$$

- After variable transformation $\mathrm{d}x \to \mathrm{d}x\, g(x) = \mathrm{d}G(x)$

$$I = \int_\Omega \frac{f(x)}{g(x)}\,\mathrm{d}G(x) = \Omega\,\langle f/g\rangle_G\ , \qquad \sigma_I = \Omega\sqrt{\frac{\langle (f/g)^2\rangle_G - \langle f/g\rangle_G^2}{N-1}}$$

- Multi-dimensional integrals: $\mathrm{d}\vec{x} \to \mathrm{d}\vec{x}'\,|\mathrm{d}\vec{x}(\vec{x}')/\mathrm{d}\vec{x}'|$
  $\to$ Jacobian changes from $1/g(x)$ to $|\mathrm{d}\vec{x}'/\mathrm{d}\vec{x}|^{-1}$
- For bijective map g of random variable $\vec{x}$ drawn from base distribution q_0, the variable $\vec{x}' = g(\vec{x})$ follows distribution inferred by chain rule:

$$q_1(\vec{x}') = q_0(g^{-1}(\vec{x}'))\,|\left|\frac{\partial g^{-1}(\vec{x}')}{\partial \vec{x}'}\right|| = q_0(\vec{x})\,|\left|\frac{\partial g(\vec{x})}{\partial \vec{x}}\right|^{-1}|$$

- In CS literature, this transformation is called a "Normalizing Flow"

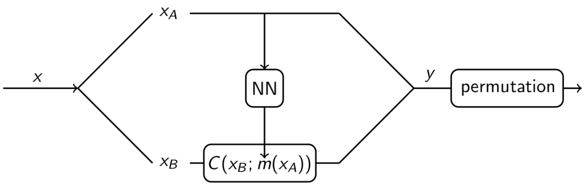
Event generation with Normalizing Flows

- If we use a Neural Network to learn g, we need to compute its gradient during inference. This is veeery slow. No, really! It's even slower!
- "Coupling layers" are special bijectors to avoid these gradients:
  - Input variables $\vec{x} = \{x_1, .., x_D\}$ partitioned into two subsets, $\vec{x}_A$ and $\vec{x}_B$

$$x'_A = x_A, \qquad x'_B = C(x_B; m(\vec{x}_A))$$

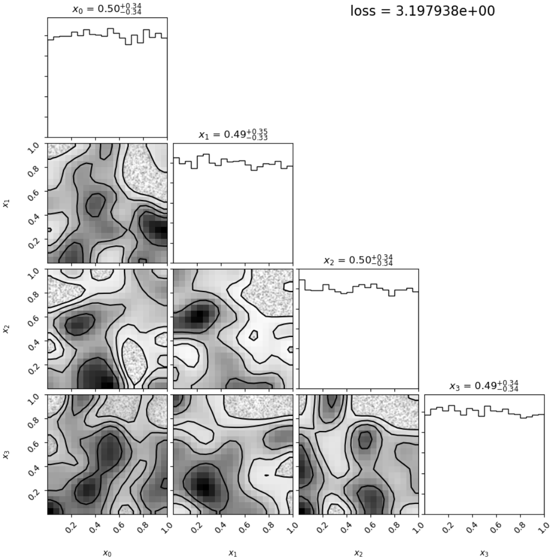
  - m is output of a Neural Network taking x_A as inputs and returning parameters of "Coupling Transform" C that will be applied to x_B
  - Inverse map is simple, leading to simple Jacobian (no $\partial m / \partial \vec{x}_A$!)

$$\left|\frac{\partial g(\vec{x})}{\partial \vec{x}}\right|^{-1} = \left|\begin{pmatrix} \vec{1} & 0 \\ \frac{\partial C}{\partial m}\frac{\partial m}{\partial \vec{x}_A} & \frac{\partial C}{\partial \vec{x}_B} \end{pmatrix}\right|^{-1} = \left|\frac{\partial C(\vec{x}_B; m(\vec{x}_A))}{\partial \vec{x}_B}\right|^{-1} .$$



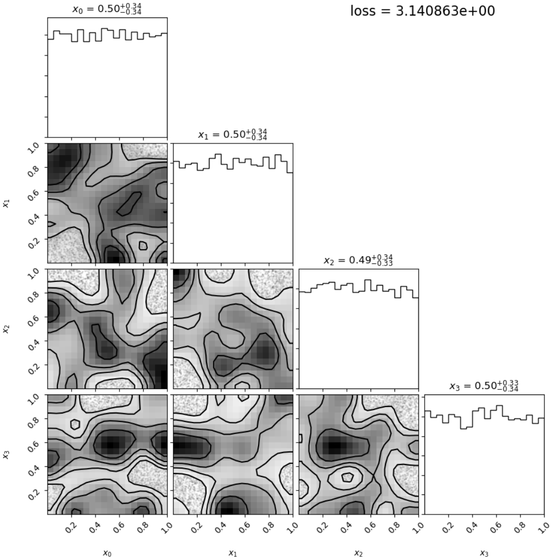
Toy example - 4-dimensional camel function

[Gao,Isaacson,Krause] arXiv:2001.05486



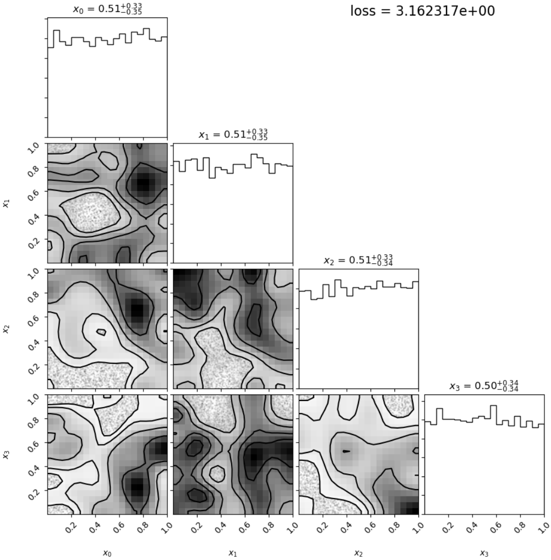
Toy example - 4-dimensional camel function

[Gao,Isaacson,Krause] arXiv:2001.05486



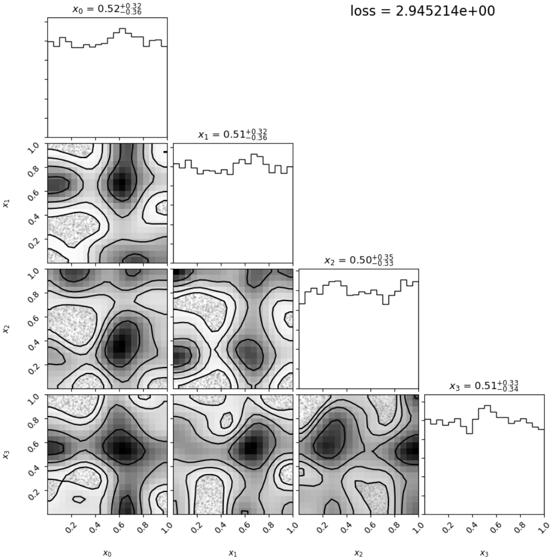
Toy example - 4-dimensional camel function

[Gao,Isaacson,Krause] arXiv:2001.05486



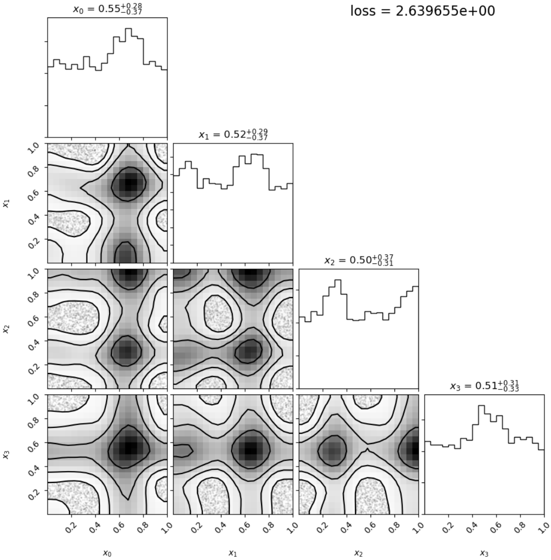
Toy example - 4-dimensional camel function

[Gao,Isaacson,Krause] arXiv:2001.05486



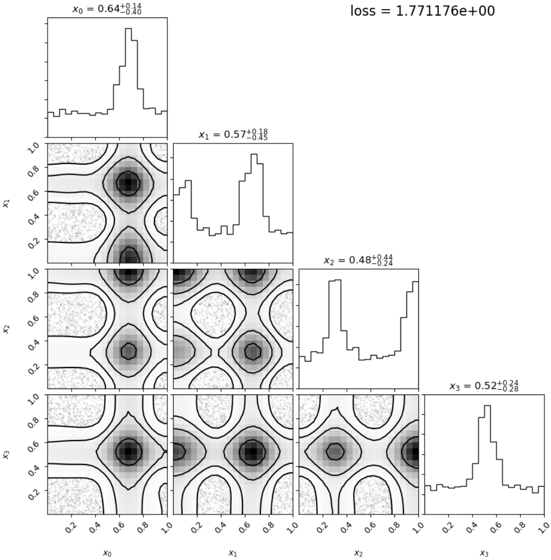
Toy example - 4-dimensional camel function

[Gao,Isaacson,Krause] arXiv:2001.05486



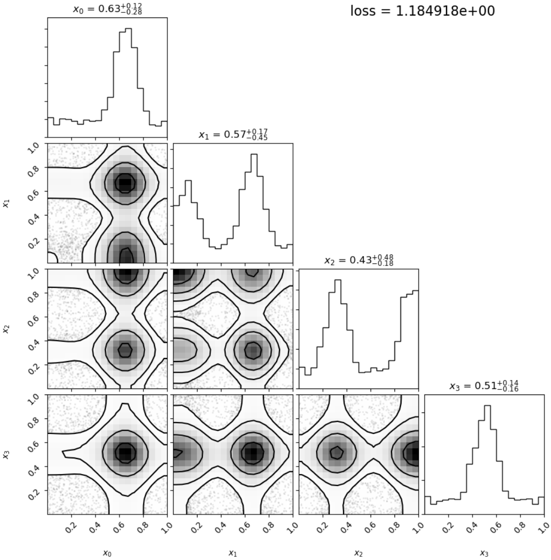
Toy example - 4-dimensional camel function

[Gao,Isaacson,Krause] arXiv:2001.05486



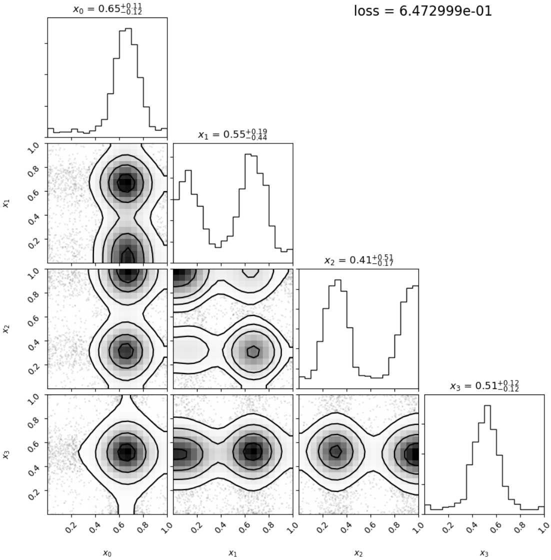
Toy example - 4-dimensional camel function

[Gao,Isaacson,Krause] arXiv:2001.05486



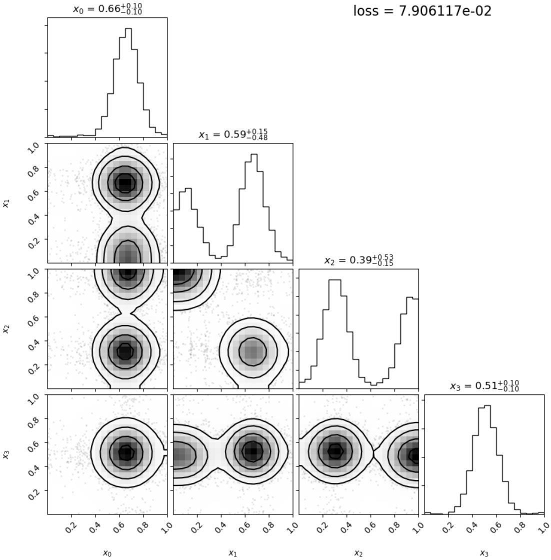
Toy example - 4-dimensional camel function

[Gao,Isaacson,Krause] arXiv:2001.05486



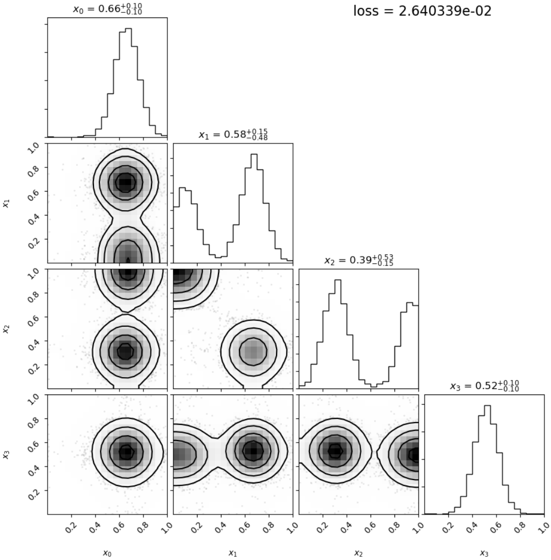
Toy example - 4-dimensional camel function

[Gao,Isaacson,Krause] arXiv:2001.05486



Toy example - 4-dimensional camel function

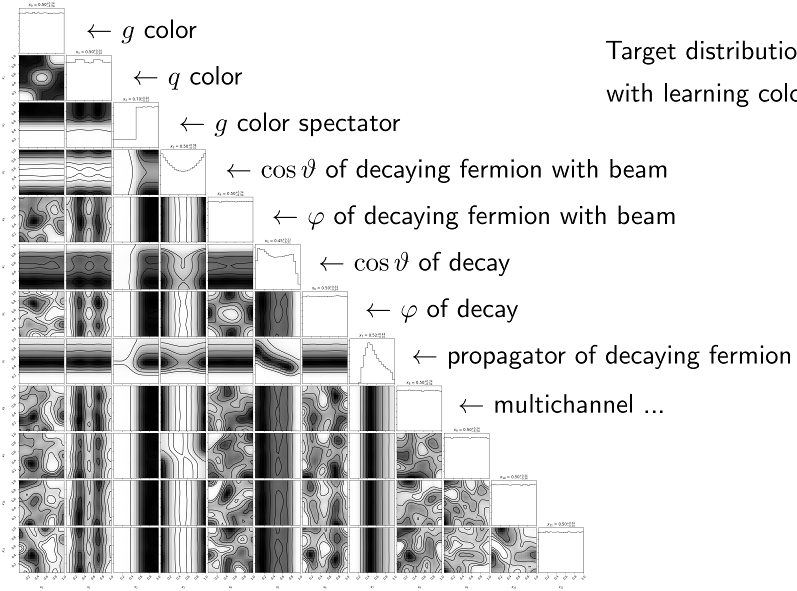
[Gao,Isaacson,Krause] arXiv:2001.05486



Real-life example: $e^+e^- \to qqg$

[Gao,Isaacson,Krause,Schulz,SH] arXiv:2001.10028

Target distribution with learning color



Real-life example: $pp \to V + jets$

[Gao,Isaacson,Krause,Schulz,SH] arXiv:2001.10028

- Check results in most performance-critical applications
- To make unweighting efficiency independent of weight outliers, define max as median of maxima in bootstrap approach [Campbell,Neumann] arXiv:1909.09117

unweighting efficiency		LO QCD					NLO QCD (RS)	
$\langle w \rangle / w_{\max}$		n =0	n =1	n =2	n =3	n =4	n =0	n =1
$W^+ + n$ jets	Sherpa	$2.5 \cdot 10^{-1}$	$3.4 \cdot 10^{-2}$	$6.7 \cdot 10^{-3}$	$1.7 \cdot 10^{-3}$	$6.6 \cdot 10^{-4}$	$6.5 \cdot 10^{-2}$	$2.9 \cdot 10^{-3}$
	NN+NF	$5.8 \cdot 10^{-1}$	$1.2 \cdot 10^{-1}$	$8.8 \cdot 10^{-3}$	$1.6 \cdot 10^{-3}$	$8.9 \cdot 10^{-4}$	$1.7 \cdot 10^{-1}$	$4.0 \cdot 10^{-3}$
	Gain	2.3	3.6	1.3	0.99	1.4	2.7	1.4
$W^- + n$ jets	Sherpa	$2.4 \cdot 10^{-1}$	$3.9 \cdot 10^{-2}$	$8.4 \cdot 10^{-3}$	$1.7 \cdot 10^{-3}$	$8.8 \cdot 10^{-4}$	$6.0 \cdot 10^{-2}$	$3.3 \cdot 10^{-3}$
	NN+NF	$6.2 \cdot 10^{-1}$	$1.3 \cdot 10^{-1}$	$1.2 \cdot 10^{-2}$	$2.3 \cdot 10^{-3}$	$9.8 \cdot 10^{-4}$	$1.6 \cdot 10^{-1}$	$3.8 \cdot 10^{-3}$
	Gain	2.6	3.2	1.5	1.4	1.17	2.8	1.2
$Z + n$ jets	Sherpa	$4.3 \cdot 10^{-1}$	$4.3 \cdot 10^{-2}$	$1.3 \cdot 10^{-2}$	$2.7 \cdot 10^{-3}$	$1.1 \cdot 10^{-3}$	$1.1 \cdot 10^{-1}$	$4.9 \cdot 10^{-3}$
	NN+NF	$5.1 \cdot 10^{-1}$	$1.1 \cdot 10^{-1}$	$1.3 \cdot 10^{-2}$	$2.6 \cdot 10^{-3}$		$1.8 \cdot 10^{-3}$	$4.9 \cdot 10^{-3}$
	Gain	1.2	2.6	1.1	0.97		1.7	1.0

Summary and Outlook

- New workflow to generate LO-merged event samples on CPU-based HPC resources
- Working towards similar workflow at NLO
- Novel technique to improve event generation with Neural Network based adaptive integration techniques
- All this heavily reliant on detailed understanding of underlying physics models and algorithms

There are no free lunches!