Sampling (up to) NNLO QCD with Normalizing Flows

KISS Meeting 2025 @ LMU

Timo Janßen

Institut für Theoretische Physik, Georg-August-Universität Göttingen

with Enrico Bothmann, Max Knobbe, Rene Poncelet, Bernhard Schmitzer, Steffen Schumann and Fabian Sinz

12 March 2025

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN IN PUBLICA COMMODA SEIT 1737

KISS

GEFÖRDERT VOM

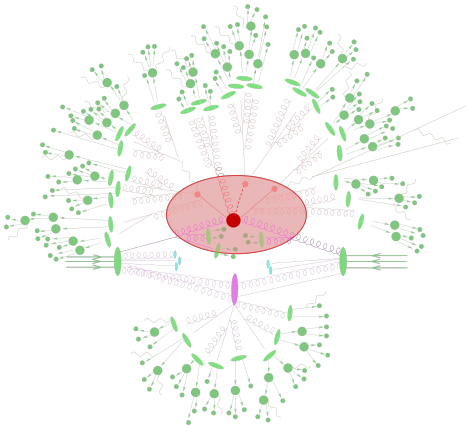
Bundesministerium für Bildung und Forschung

Goals

- increase unweighting efficiency in calculations relevant for LHC
- for NNLO: reduce variance of total cross section estimator
- don't introduce new systematic errors / biases
- scale to highest multiplicities relevant for LHC
- keep it simple

→ avoid cumbersome integration into existing codes with many dependencies that need to be maintained

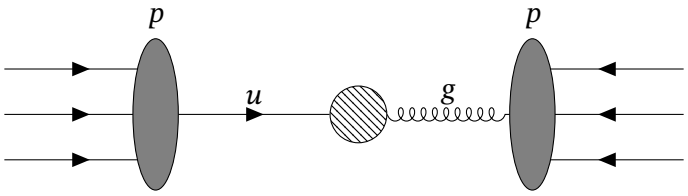
Monte Carlo event generators



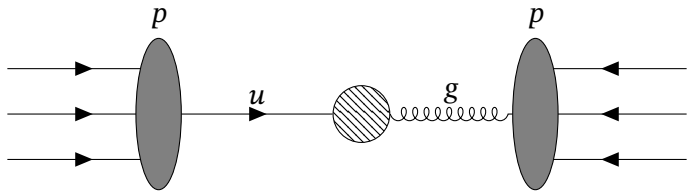
hard interaction

- full-featured simulation
- from high to low energy scales
- **hard interaction** of quarks & gluons via perturbation theory
- factorize hard interaction from PDFs, parton shower, hadronization, decays, electroweak corrections, multiple interaction, ...

Higher order QCD cross sections



Higher order QCD cross sections



Hadronic cross-section:

$$\sigma\big(h(P_1)\,h(P_2) \to X\big) = \sum_{ab} \iint_0^1 \mathrm{d}x_1 \mathrm{d}x_2 \; f_a(x_1)\, f_b(x_2)\, \hat{\sigma}_{ab}(x_1 P_1, x_2 P_2)$$

- separate perturbative & non-perturbative physics

Higher order QCD cross sections

Hadronic cross-section:

$$\sigma\big(h(P_1)\,h(P_2) \to X\big) = \sum_{ab} \iint_0^1 \mathrm{d}x_1 \mathrm{d}x_2 \; f_a(x_1)\, f_b(x_2)\, \hat{\sigma}_{ab}(x_1 P_1, x_2 P_2)$$

- separate perturbative & non-perturbative physics

Partonic cross-section:

$$\hat{\sigma}_{ab} = \hat{\sigma}_{ab}^{(0)} + \hat{\sigma}_{ab}^{(1)} + \hat{\sigma}_{ab}^{(2)} + \mathcal{O}\Big(\alpha_s^{n_{(0)}} \alpha_s^3\Big)$$

- can be calculated from first principles
- $n_{(0)}$: tree-level coupling order

Higher order QCD cross sections

Hadronic cross-section:

$$\sigma\big(h(P_1)\,h(P_2) \to X\big) = \sum_{ab} \iint_0^1 \mathrm{d}x_1 \mathrm{d}x_2 \; f_a(x_1)\, f_b(x_2)\, \hat{\sigma}_{ab}(x_1 P_1, x_2 P_2)$$

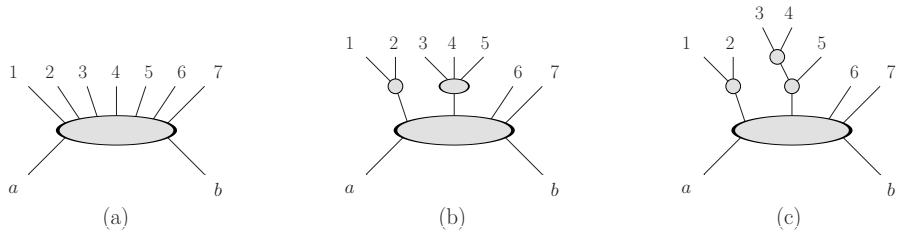
Partonic cross-section:

$$\hat{\sigma}_{ab} = \hat{\sigma}_{ab}^{(0)} + \hat{\sigma}_{ab}^{(1)} + \hat{\sigma}_{ab}^{(2)} + \mathcal{O}\Big(\alpha_s^{n_{(0)}} \alpha_s^3\Big)$$

Leading Order (LO) cross-section:

$$\hat{\sigma}_{ab}^{(0)}(x_1 P_1, x_2 P_2) = \frac{1}{2E_1 E_2 |v_1 - v_2|} \int \Big(\prod_f \frac{\mathrm{d}^3 p_f}{(2\pi)^3} \frac{1}{2E_f} \Big)$$
$$\times \, |\mathcal{M}(p_a p_b \to \{p_f\})|^2 \, (2\pi)^4 \, \delta^{(4)}\Big(x_1 P_1 + x_2 P_2 - \sum p_f\Big)$$

LO calculations with PEPPER [Bothmann *et al.* arXiv:2311.06198, SciPost Phys. 15, 169 (2023)]



- runs natively on the GPU, extremely parallel
- uses CHILI, a simple yet efficient algorithm for phase space sampling
- single t-channel combined with any number of s channel decays
- helicity sampling by default
- distribution can look very different for different helicity states

→ learn the helicity amplitudes by making the model conditional:

$$q(x;\theta) \to q(x|\lambda_1 \dots \lambda_n;\theta)$$

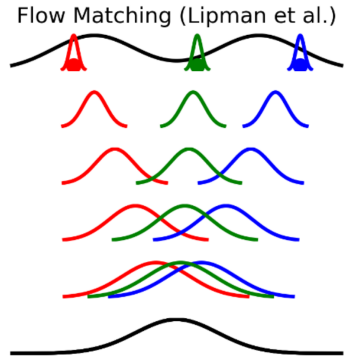
→ This is something VEGAS cannot do!

Talking to PEPPER

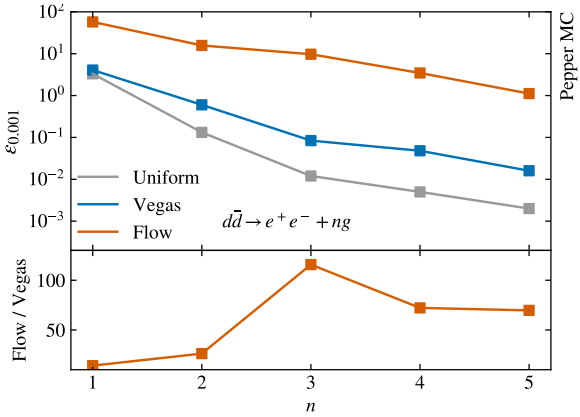
- implemented a minimal I/O interface into PEPPER
- based on HDF5 event format [Bothmann *et al.* Phys. Rev. D **109**, 014013 (2024)]
- when sampling with CHILI, write out ‘random numbers’ and weights
- read in random numbers, Jacobians, helicities from file and pass through CHILI mapping

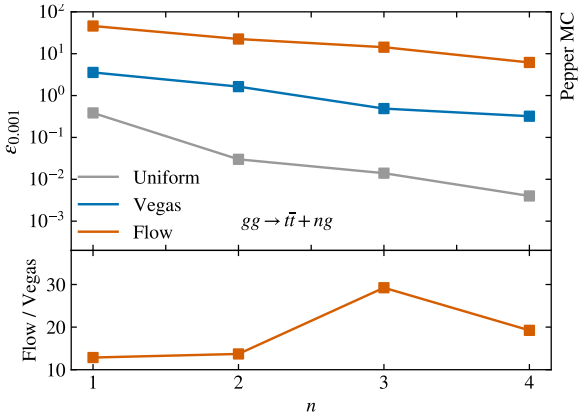
LO results

- consider two LHC examples:
  $pp \to Z + n$ jets ($n \leq 5$) and $pp \to t\bar{t} + n$ jets ($n \leq 4$)
- specific partonic channels:
  $d\bar{d} \to e^+e^- + ng$ and $gg \to t\bar{t} + ng$
- dimensionalities from 7 to 19
- number of helicity states from 8 to 256
- baseline: VEGAS Lepage (1978), Journal of Computational Physics **27** (2) trained on many events (>400M for $Z + 5$ jets)
- use flow matching to learn a velocity field (fast to train)
- generate samples, Jacobians by integrating velocity field with ODE solver (slow to sample)
- each flow trained on 170M events in total
- cut efficiencies high and on par with VEGAS



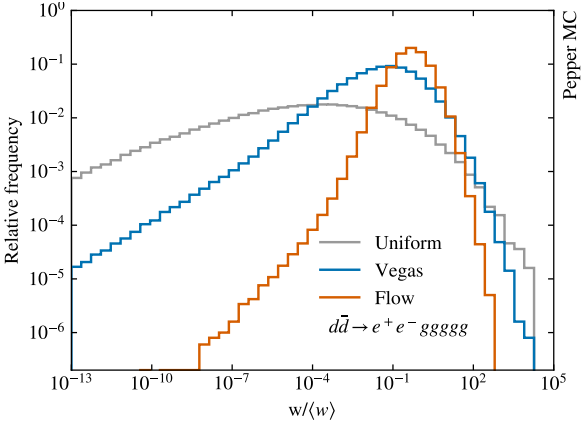
LO results: unweighting efficiency

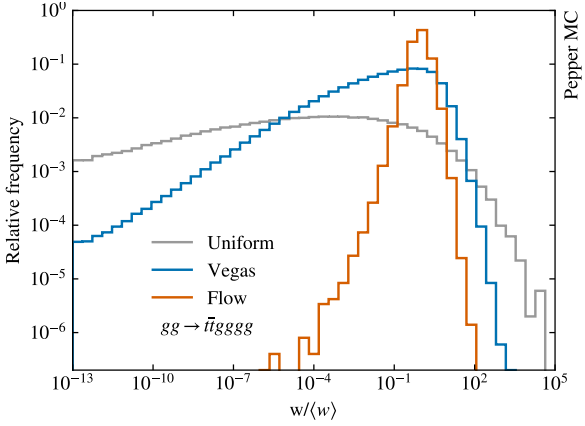




- significantly increased unw. eff. for all multiplicities
- factors up to 115
- seems to scale well

LO results: weight distributions





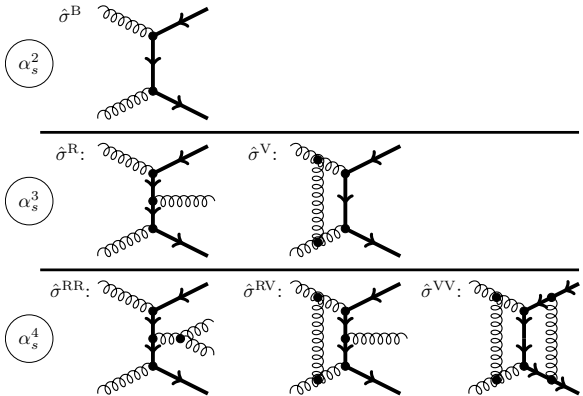
- much narrower weight distributions

Higher order contributions

Partonic cross-section:

$$\hat{\sigma}_{ab} = \hat{\sigma}_{ab}^{(0)} + \hat{\sigma}_{ab}^{(1)} + \hat{\sigma}_{ab}^{(2)} + \mathcal{O}\left(\alpha_s^{n_{(0)}}\alpha_s^3\right)$$

Example: gluonic top quark pair production



Next-to-leading order QCD

partonic NLO cross-section:

$$\hat{\sigma}_{ab}^{(1)} = \hat{\sigma}_{ab}^{R} + \hat{\sigma}_{ab}^{V} + \hat{\sigma}_{ab}^{C}$$

- $\hat{\sigma}_{ab}^{R}$ and $\hat{\sigma}_{ab}^{V}$ are separately divergent
- use dimensional regularization to replace divergences by poles

Next-to-leading order QCD

partonic NLO cross-section:

$$\hat{\sigma}_{ab}^{(1)} = \hat{\sigma}_{ab}^{R} + \hat{\sigma}_{ab}^{V} + \hat{\sigma}_{ab}^{C}$$

- $\hat{\sigma}_{ab}^{R}$ and $\hat{\sigma}_{ab}^{V}$ are separately divergent
- use dimensional regularization to replace divergences by poles

NLO subtraction idea:

$$\begin{aligned}\hat{\sigma}_{ab}^{(1)} &= \int_{m+1}\left[\mathrm{d}\hat{\sigma}_{ab}^{R} - \mathrm{d}\hat{\sigma}_{ab}^{A}\right] + \int_{m+1}\mathrm{d}\hat{\sigma}_{ab}^{A} + \int_{m}\mathrm{d}\hat{\sigma}_{ab}^{V} + \int_{m}\mathrm{d}\hat{\sigma}_{ab}^{C} \\ &= \int_{m+1}\left[\left(\mathrm{d}\hat{\sigma}_{ab}^{R}\right)_{\epsilon=0} - \left(\mathrm{d}\hat{\sigma}_{ab}^{A}\right)_{\epsilon=0}\right] + \int_{m}+\left[\mathrm{d}\hat{\sigma}_{ab}^{V} + \int_{1}\mathrm{d}\hat{\sigma}_{ab}^{A}\right]_{\epsilon=0} + \int_{m}\mathrm{d}\hat{\sigma}_{ab}^{C}\end{aligned}$$

- $\hat{\sigma}_{ab}^{A}$ is a local counterterm

The STRIPPER scheme M. Czakon and D. Heymes: Nucl. Phys. B **890** (2014) 152–227

- sector-improved residue subtraction
- introduce selector function to filter out individual infrared limits in ‘sectors’
- poles can be extracted and cancelled by virtual corrections
- challenge: evaluate sum of many integrals
- individual contributions can be of very different shape

STRIPPER NLO calculations

- at NLO we have the following contributions:

$$\hat{\sigma}_{ab}^{R} = \hat{\sigma}_{ab}^{RF} + \hat{\sigma}_{ab}^{RU}$$
$$\hat{\sigma}_{ab}^{V} = \hat{\sigma}_{ab}^{VF} + \hat{\sigma}_{ab}^{VU}$$

- phase space is mapped onto a unit hypercube
- sample over sectors and helicities

Negative weights

- NLO cross sections are not positive-definite!
- sometimes we subtract too much
- typically one adapts the sampling distribution to $|\mathrm{d}\sigma|$
- keep sign of target during integration
- found that stratification into positive/negative parts is better than learning $|f|$

$$f(\mathbf{x}) = f_+(\mathbf{x}) + f_-(\mathbf{x}), \quad \text{with} \quad f_\pm(\mathbf{x}) = \mathrm{sgn}(f(\mathbf{x}))|f(\mathbf{x})|,$$

- use two sampling densities:

$$I = \int_{\mathbf{H}_+(\mathbf{x})\in\Omega} \mathrm{d}\mathbf{H}_+ \, \frac{f_+(\mathbf{x})}{h_+(\mathbf{x})} + \int_{\mathbf{H}_-(\mathbf{x})\in\Omega} \mathrm{d}\mathbf{H}_- \, \frac{f_-(\mathbf{x})}{h_-(\mathbf{x})}.$$

- reduced variance if we can accurately sample within the strata
  → flows are able to learn boundaries between pos/neg very efficiently

Talking to STRIPPER

- we built a python interface (using pybind11) around the STRIPPER library
- use STRIPPER (sequentially on CPU) to evaluate amplitudes in individual sectors but do actual integration in python
- input ‘random numbers’ and get weights
- use GPU to generate large sample of random numbers from trained flow model and write to disk
- evaluate them asynchronously with STRIPPER

NLO results

- consider top quark pair production at LHC
- gluonic initial state
- baseline: VEGAS
- train on 10M weighted events

NLO results

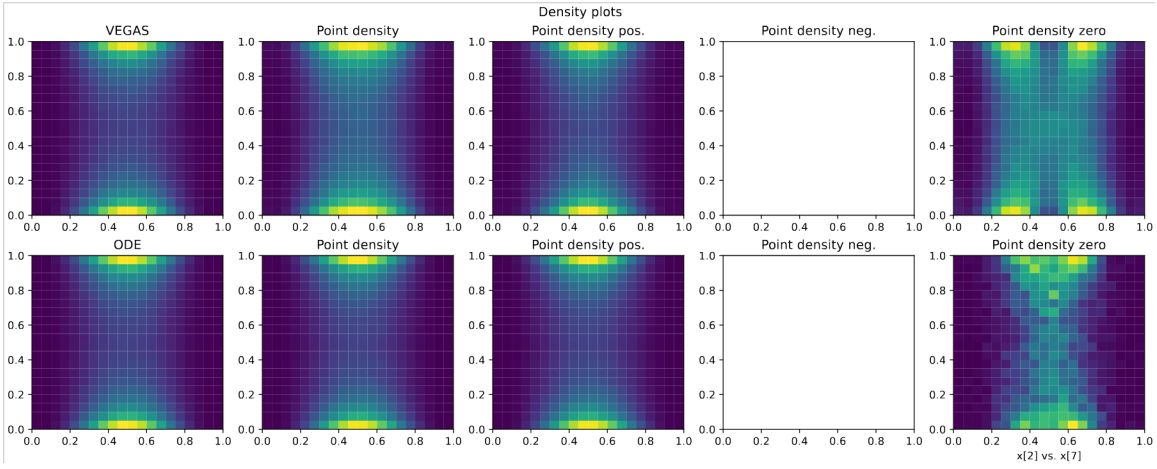
Contr.	positive		negative	
	VEGAS	ODE-Flow	VEGAS	ODE-Flow
σ^{RF}	895.50 ± 0.95	894.9 ± 0.2	-198.13 ± 0.42	-198.44 ± 0.06
σ^{RU}	443.15 ± 0.60	441.85 ± 0.15	-980.98 ± 0.77	-980.2 ± 0.2
σ^{VF}	14.030 ± 0.008	14.018 ± 0.001	-58.94 ± 0.03	-58.9315 ± 0.0050

- note: MC error scales as $\frac{1}{\sqrt{N}}$

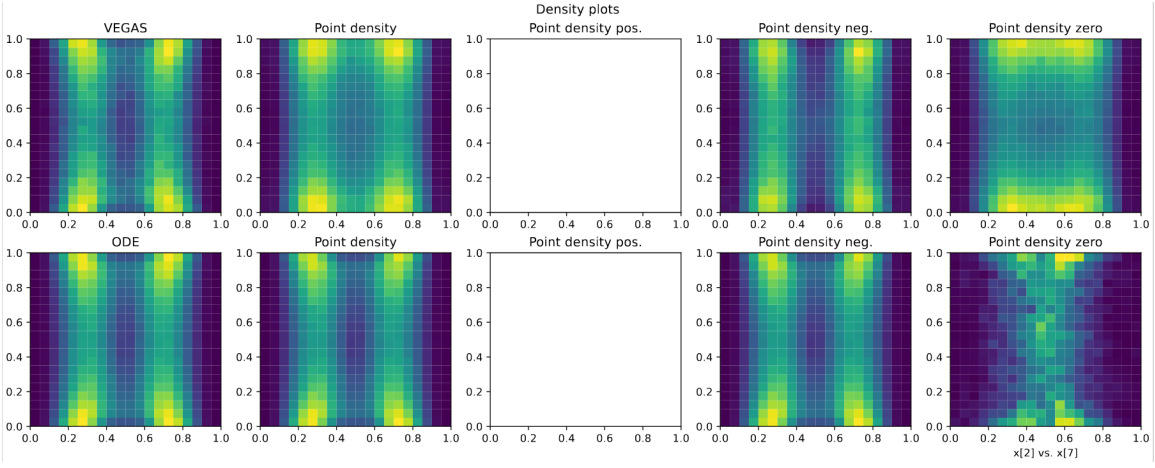
→ error reduction by $\frac{1}{x}$ means $\frac{1}{x^2}$ as many events needed

- efficiency increased by factors between 15 and 64
- flow samples within pos/neg very efficiently: $> 95\,\%$ for σ^{RF} (VEGAS: $> 60\,\%$)

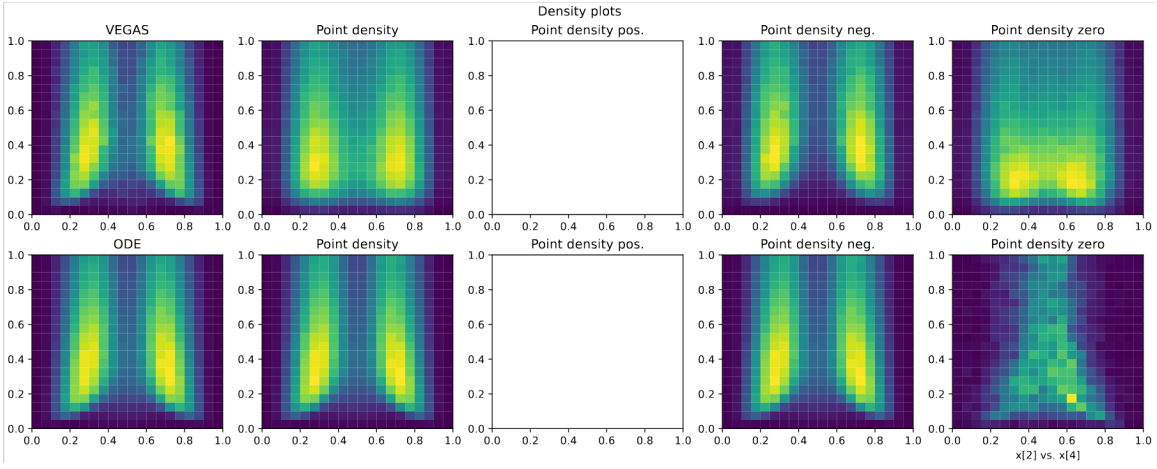
NLO results: RF pos 2d distribution



NLO results: RF neg 2d distribution

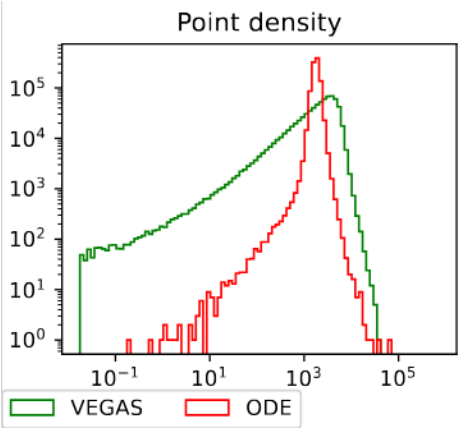


NLO results: RF neg 2d distribution

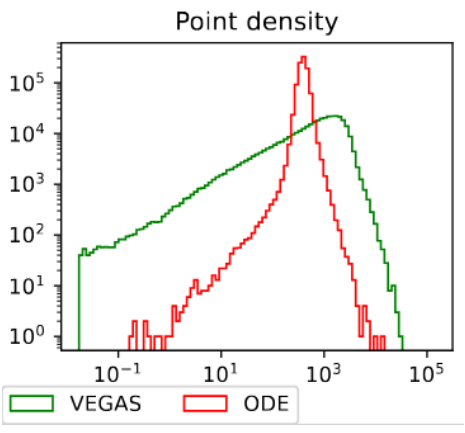


NLO results: RF weight distributions

positive



negative



STRIPPER NNLO calculations

- the NNLO correction can be written as

$$\hat{\sigma}_{ab}^{(2)} = \hat{\sigma}_{ab}^{RR} + \hat{\sigma}_{ab}^{RV} + \hat{\sigma}_{ab}^{VV} + \hat{\sigma}_{ab}^{C1} + \hat{\sigma}_{ab}^{C2}$$

- we have the following contributions:

$$\begin{aligned}
\hat{\sigma}_{ab}^{RR} &= \hat{\sigma}_{ab}^{RRF} + \hat{\sigma}_{ab}^{RRSU} + \hat{\sigma}_{ab}^{RRDU} \\
\hat{\sigma}_{ab}^{RV} &= \hat{\sigma}_{ab}^{RVF} + \hat{\sigma}_{ab}^{RVDU} + \hat{\sigma}_{ab}^{RVFR} \\
\hat{\sigma}_{ab}^{VV} &= \hat{\sigma}_{ab}^{VVF} + \hat{\sigma}_{ab}^{VVDU} + \hat{\sigma}_{ab}^{VVFR}
\end{aligned}$$

- many sectors and helicity states
- large cancellations lead to numerical difficulties

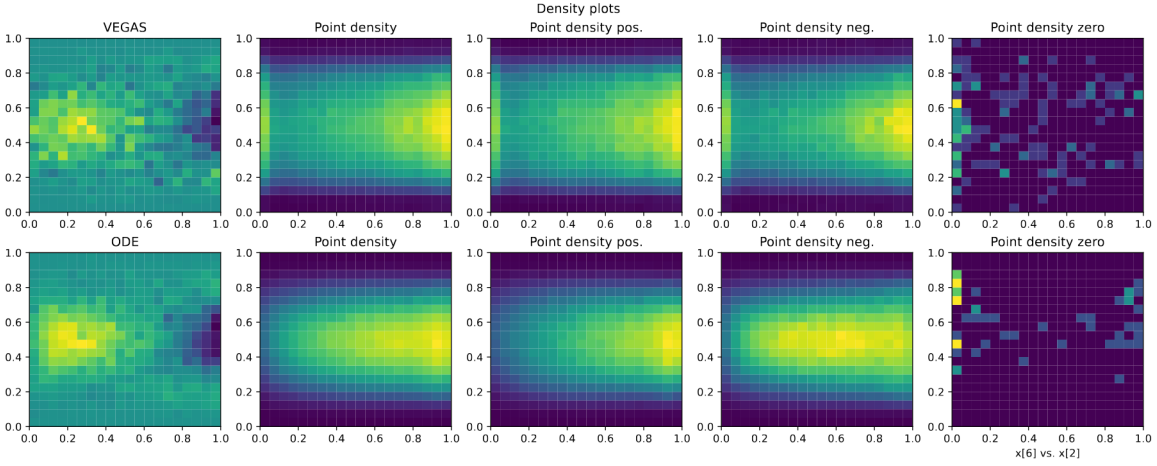
NNLO results

selected results:

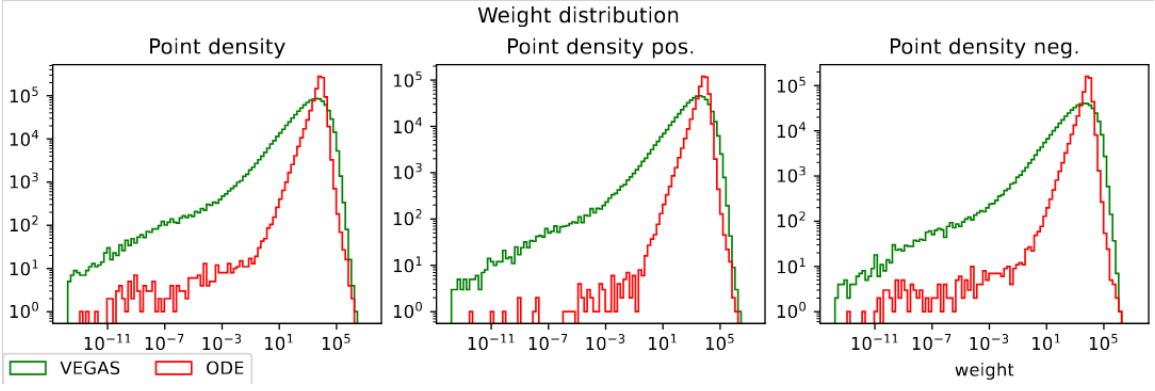
Contr.	positive		negative	
	VEGAS	ODE-Flow	VEGAS	ODE-Flow
σ^{RRF}	382.3 ± 1.3	381.2 ± 0.6	-413.5 ± 1.3	-414.19 ± 0.56
σ^{RRSU}	3717.5 ± 5.6	3700.0 ± 2.8	-881.2 ± 2.2	-884.7 ± 4.4
σ^{RRDU}	5638.9 ± 6.5	5643.9 ± 3.8	-8078.3 ± 8.5	-8071.0 ± 5.7

- find gains for almost all contributions
- so far, lower gains than for NLO
- numerical difficulties might play a role
- work in progress

NNLO results: RRF 2d distribution



NNLO results: RRF weight distribution



Conclusions

- used Flow Matching to improve sampling for LO, NLO and NNLO calculations
- model is conditioned on discrete variables (helicities, sectors)
- considered $Z + n$ jets and $t\bar{t} + n$ jets production at LHC
- implemented minimal interfaces to PEPPER and STRIPPER: exchange random numbers/weights through files
- found significant gains both for integration and unweighted event generation:

$\longrightarrow$ up to 115 times increase in unweighting efficiency at LO

$\longrightarrow$ up to 64 times more efficient integration at NLO

- working on final numbers for NNLO

Conclusions

- used Flow Matching to improve sampling for LO, NLO and NNLO calculations
- model is conditioned on discrete variables (helicities, sectors)
- considered $Z + n$ jets and $t\bar{t} + n$ jets production at LHC
- implemented minimal interfaces to PEPPER and STRIPPER: exchange random numbers/weights through files
- found significant gains both for integration and unweighted event generation:

$\longrightarrow$ up to 115 times increase in unweighting efficiency at LO

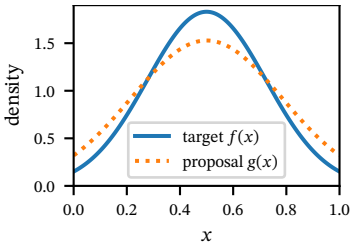
$\longrightarrow$ up to 64 times more efficient integration at NLO

- working on final numbers for NNLO

Questions and suggestions are welcome!

Backup Slides

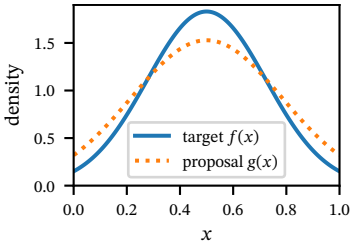
Sampling basics: importance sampling



$$\int_{[0,1]^d} f(x)\,\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{g(x_i)}, \quad x_i \sim g(x)$$

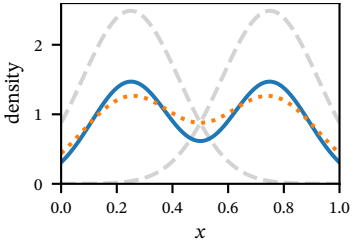
- draw points from a proposal distribution
- points come with weights $w_i = \frac{f(x_i)}{g(x_i)}$
- spread of weights is small if proposal is close to target

Sampling basics: importance sampling



$$\int_{[0,1]^d} f(x)\,\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{g(x_i)}, \quad x_i \sim g(x)$$

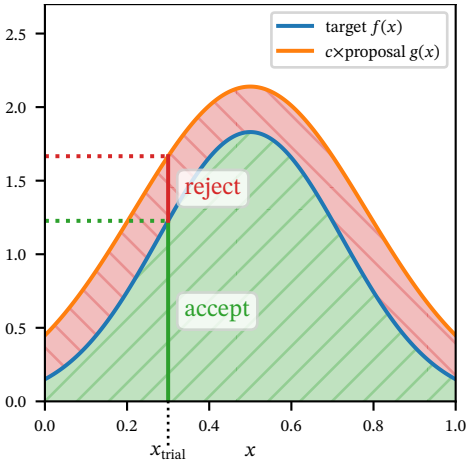
- draw points from a proposal distribution
- points come with weights $w_i = \dfrac{f(x_i)}{g(x_i)}$
- spread of weights is small if proposal is close to target



multichannel: $g(x) = \sum_{i}^{N_c} \alpha_i g_i(x), \quad \sum_{i}^{N_c} \alpha_i = 1$

- use mixture distribution for multimodal targets
- construct channels based on physics knowledge
- automatic channel weight optimization [Kleiss&Pittau Comput.Phys.Commun. 83 (1994) 141-146]

Unweighted event generation (rejection sampling)



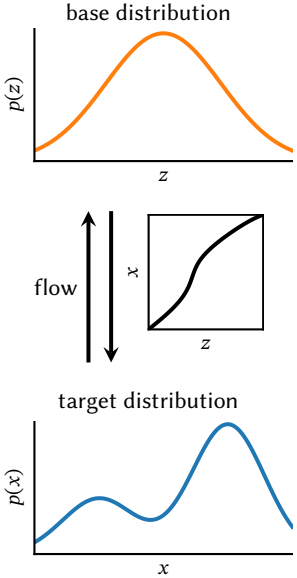
Goal

generate i.i.d. samples

- sample from proposal $g(x)$ (via inverse transform)
- determine weights $w(x) = \dfrac{f(x)}{g(x)}$
- accept events with probability $p_{\text{accept}(x)} = \dfrac{w(x)}{w_{\max}}$

  → reduce sample size!
- unweighting efficiency: $\eta = \dfrac{\langle w \rangle}{w_{\max}}$

Normalizing Flows



- generative model
- based on change-of-variable formula:

$$\mathbf{x} = g(\mathbf{z}) \quad \Rightarrow \quad p_{\mathbf{x}}(\mathbf{x}) = p_{\mathbf{z}}(\mathbf{z}) \left| \det \left(\frac{\partial g(\mathbf{z})}{\partial \mathbf{z}^T} \right) \right|^{-1}$$

→ transform simple distribution into complex one

key properties

- invertible (bijective map)
- can evaluate probability density exactly
- parameterized by NNs → train via loss minimization

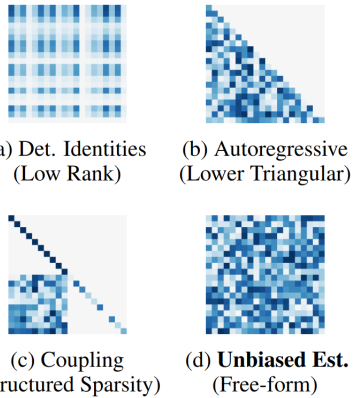
- use for adaptive importance sampling
- replacement for VEGAS [Lepage (1978), JCP **27** (2)]

Normalizing Flows

We can distinguish three kinds of normalizing flows:

- discrete flows (autoregressive, coupling)
- invertible residual networks
- continuous time flows (Neural ODEs)

Distinguishing feature: How the Jacobian is made tractable



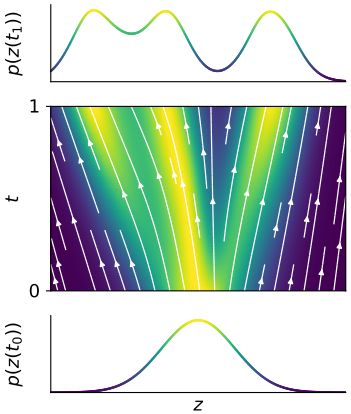
Continuous normalizing flows – Neural ODE

reconsider change of variable formula:

$$\mathbf{x} = g(\mathbf{z}) \quad \Rightarrow \quad p_{\mathbf{x}}(\mathbf{x}) = p_{\mathbf{z}}(\mathbf{z}) \left| \det \left(\frac{\partial g(\mathbf{z})}{\partial \mathbf{z}^T} \right) \right|^{-1}$$

now consider a transformation continuous in time:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = g(\mathbf{x}(t), t) \quad \Rightarrow \quad \frac{\partial \log p_{\mathbf{x}}(\mathbf{x})}{\partial t} = -\operatorname{tr}\left(\frac{\mathrm{dg}}{\mathrm{d}\mathbf{x}(t)} \right)$$



- g only needs to be Lipschitz continuous but not bijective → use a neural network
- free-form Jacobian
- trace scales better than determinant

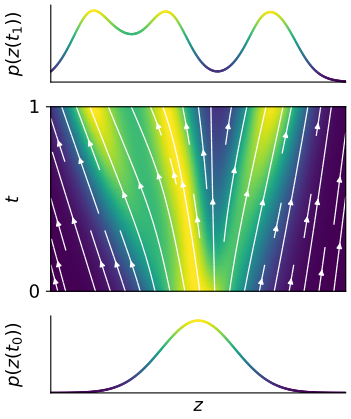
Continuous normalizing flows – Neural ODE

The instantaneous change of variable is related to the continuity equation well known in physics:

$$\frac{\mathrm{d}}{\mathrm{d}t} p_t(\mathbf{x}) + \mathrm{div}\big(p_t(\mathbf{x})\, v_t(\mathbf{x})\big) = 0$$

where v_t is a time-dependent vector field and $\mathrm{div} = \sum_{i=1}^{d} \frac{\partial}{\partial x^i}$

→ probability density 'flows' like a fluid with velocity v_t



Continuous normalizing flows – Neural ODE

Calculate the log probability together with the flow trajectory by numerically solving the ODE

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} g(\mathbf{x},t) \\ \log p_t(g(\mathbf{x},t)) \end{bmatrix} = \begin{bmatrix} v_t(g(\mathbf{x},t)) \\ -\operatorname{div}\big(p_t(\mathbf{x})\, v_t(\mathbf{x})\big) \end{bmatrix}$$

given the initial conditions

$$\begin{bmatrix} g(\mathbf{x},0) \\ \log p_0(g(\mathbf{x},0)) \end{bmatrix} = \begin{bmatrix} \mathbf{z} \\ p_{\mathbf{z}}(\mathbf{z}) \end{bmatrix}$$

→ flow from base $p_0 = p_z$ to $p_1(\mathbf{x})$ by integrating over $t \in [0,1]$

given a sample $\mathbf{x}_1$, we can compute $p_1(\mathbf{x}_1)$ by solving the ODE in reverse

Simulation-free training (flow matching)

Regression objective for the vector field:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\ x\sim p_t(x)}\|v_t(x;\theta) - u_t(x)\|^2$$

with target probability density path $p_t(x)$ generated by vector field $u_t(x)$
→ intractable since we don't know a valid choice for $u_t(x)$

Lipman *et al.* ICLR 2023

Simulation-free training (flow matching)

Regression objective for the vector field:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\ x\sim p_t(x)}\|v_t(x;\theta) - u_t(x)\|^2$$

with target probability density path $p_t(x)$ generated by vector field $u_t(x)$
→ intractable since we don't know a valid choice for $u_t(x)$

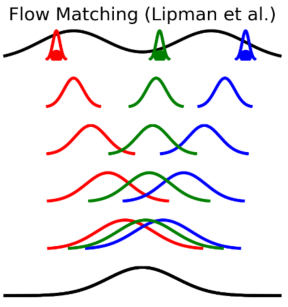
Conditional flow matching objective:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\ x_1\sim q(x_1),\ x\sim p_t(x|x_1)}\|v_t(x;\theta) - u_t(x|x_1)\|^2$$

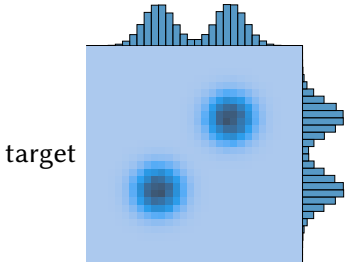
→ the gradients w.r.t. θ are the same as for $\mathcal{L}_{\text{FM}}$!

Choose Gaussian probability paths:

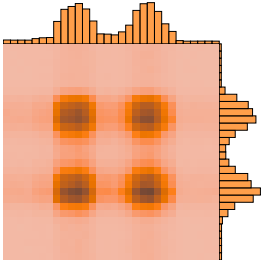
$$p_t(x|x_1) = \mathcal{N}(x \mid tx_1, (t\sigma_{\min} - t + 1)^2 I)$$



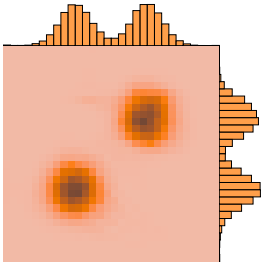
Toy example: VEGAS vs. normalizing flow



target



VEGAS



normalizing flow

→ replacing VEGAS with normalizing flows reduces the spread of weights