Exploring Phase Space with Flow Matching

ML4Jets 2024 Paris

Timo Janßen

Campus-Intitute Data Science, Georg-August-Universität Göttingen

with Fabian Sinz and Bernhard Schmitzer

5 November 2024

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN IN PUBLICA COMMODA SEIT 1737

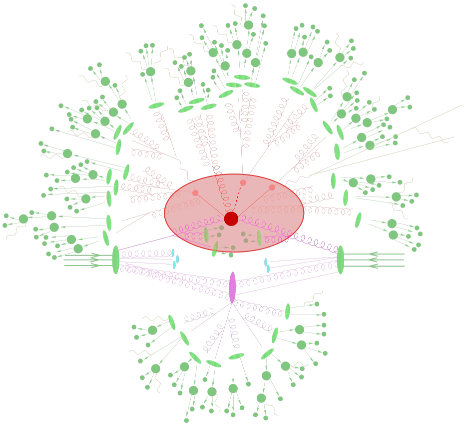
Campus-Institut Data Science

Motivation

- parton-level Monte Carlo event generation scales badly with number of particles
  - increasing number of Feynman diagrams
  - decreasing unweighting efficiency
- HL-LHC demands more simulated data

  $\rightarrow$ improve sampling efficiency
- Here: learn to sample efficiently with Continuous Normalizing Flows (CNFs)
- Note: results are preliminary, no publication yet

Monte Carlo event generators

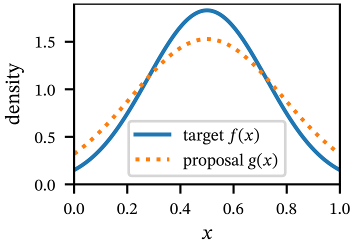


hard interaction

- full-featured simulation
- from high to low energy scales
- **hard interaction** of quarks & gluons via perturbation theory
- sample 4-momenta according to differential cross section

  → expensive, multimodal, narrow peaks, cuts
- factorize hard interaction from PDFs, parton shower, hadronization, decays, electroweak corrections, multiple interaction, …

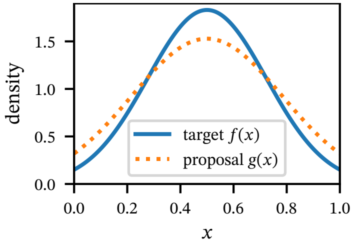
Sampling basics: importance sampling



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

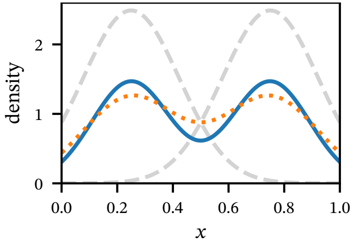
- draw points from a proposal distribution
- points come with weights $w_i = \frac{f(x_i)}{g(x_i)}$
- spread of weights is small if proposal is close to target

Sampling basics: importance sampling



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

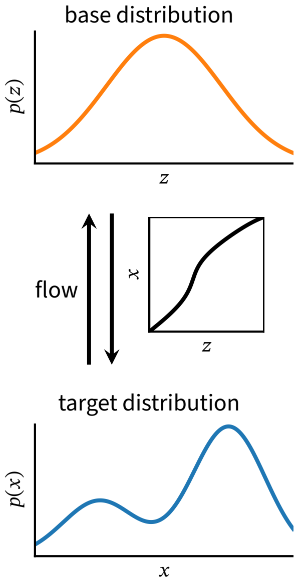
- draw points from a proposal distribution
- points come with weights $w_i = \dfrac{f(x_i)}{g(x_i)}$
- spread of weights is small if proposal is close to target



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

- 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]

Normalizing Flows



- based on change-of-variable formula:

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

→ transform simple distribution into complex one

key properties

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

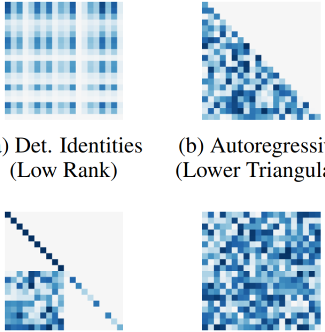
- use for adaptive importance sampling
- replacement for VEGAS [Lepage (1978), JCP **27** (2)]

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



(a) Det. Identities (Low Rank)

(b) Autoregressive (Lower Triangular)

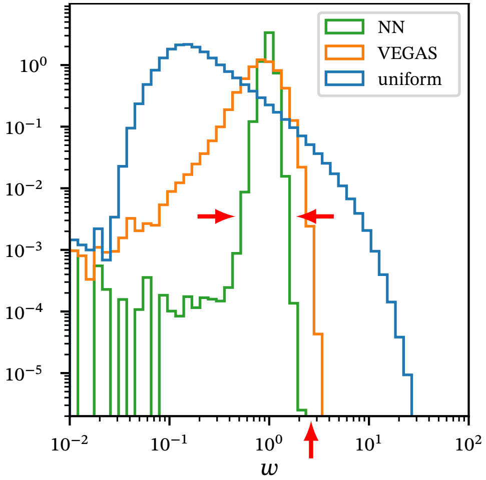
(c) Coupling (Structured Sparsity)

(d) **Unbiased Est.** (Free-form)

Chen *et al*. NeurIPS 2019

Discrete flow results: unweighted events for $gg \to 4g$

weight distribution



- presented at ML4Jets 2021
- 1st application of NFs to HEP phase space sampling
- HAAG phase space mapping [van Hameren&Papadopoulos, Eur.Phys.J.C25:563-574,2002]
- 8 dimensions, multichannel with 3 channels
- remap each channel with one normalizing flow
- non-trivial phase space cuts

unw. efficiency gain over VEGAS

$t \to e^+\nu_e b$	$e^+e^- \to t\bar{t}$	$gg \to 3g$	$gg \to 4g$
1.7	2.0	2.3	1.5

E. Bothmann, TJ, M. Knobbe, T. Schmale, S. Schumann: SciPost Phys. 8, 069 (2020)

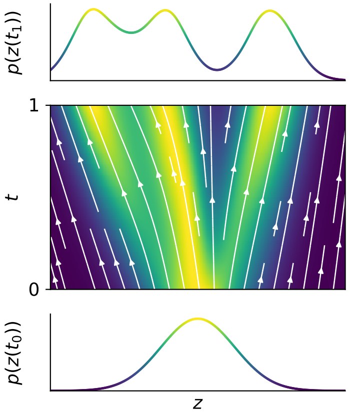
Continuous normalizing flows – Neural ODE

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:

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



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

Chen *et al*. NeurIPS 2018

© Grathwohl *et al*. ICLR 2019

Continuous normalizing flows – Neural ODE

We can calculate the log probability together with the flow trajectory by numerically solving the ODE

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

given the initial conditions

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

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

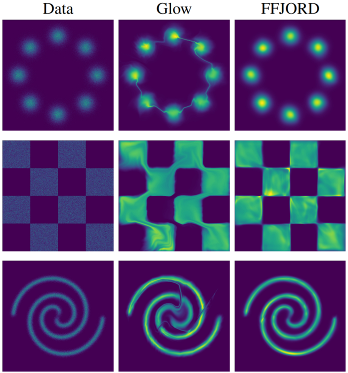
given a sample $\mathbf{x}_1$, we can compute $p_1(\mathbf{x}_1)$ by solving the ODE in reverse

Training a CNF

- approximate the target vector field with NN $v_t(\mathbf{x}; \theta)$
- train by minimizing the KL divergence between the target distribution and the generated distribution (equivalent to training of discrete flows)
- gradients for backpropagation are available through the adjoint ODE
- integrating the ODE requires many evaluation of the vector field

  $\rightarrow$ slow training and sampling

discrete flow   continuous flow



Simulation-free training (flow matching)

Regression objective for the vector field:

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

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:

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

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 $u_t(x)$

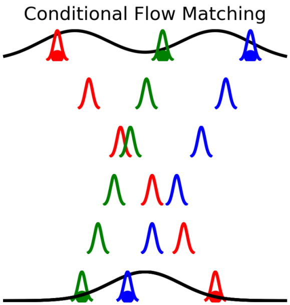
Conditional flow matching objective:

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

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

Choose Gaussian probability paths:

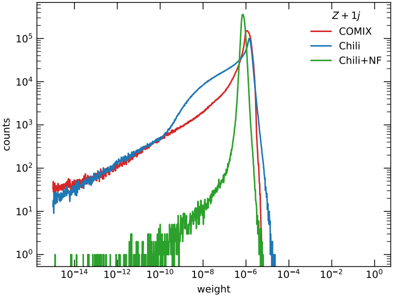
$$p_t(x|x_1) = \mathcal{N}(x \,|\, tx_1,\, (t\sigma_{\min} - t + 1)^2\, I)$$

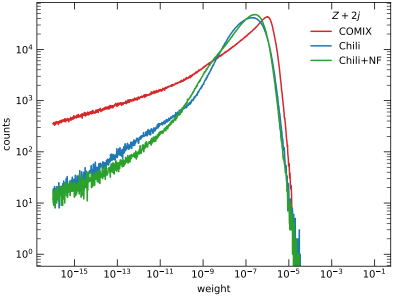


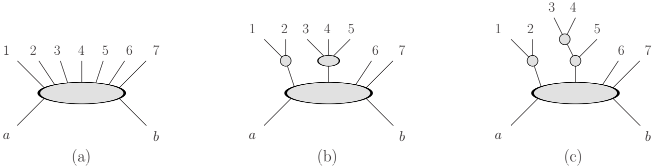
Lipman *et al.* ICLR 2023

© Tong *et al.* TMLR 03/2024

Phase space parameterization: CHILI



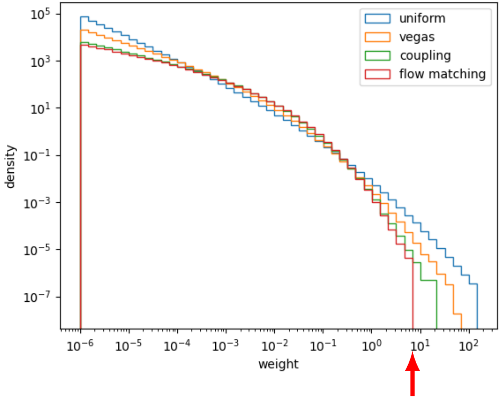




- simple yet efficient algorithm for phase space sampling
- single t-channel combined with any number of s channel decays
- used in the GPU event generator PEPPER [Bothmann *et al.* arXiv:2311.06198]
- NF leads to narrower weight distributions
- diminishing gains when more jets are added

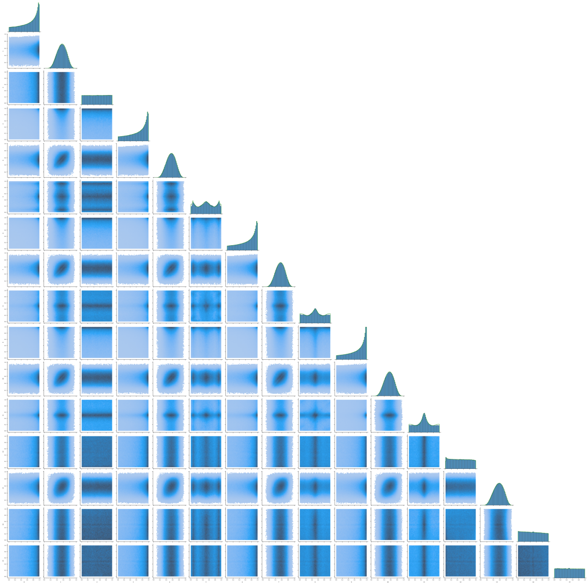
Bothmann *et al.*: SciPost Phys. 15, 169 (2023)

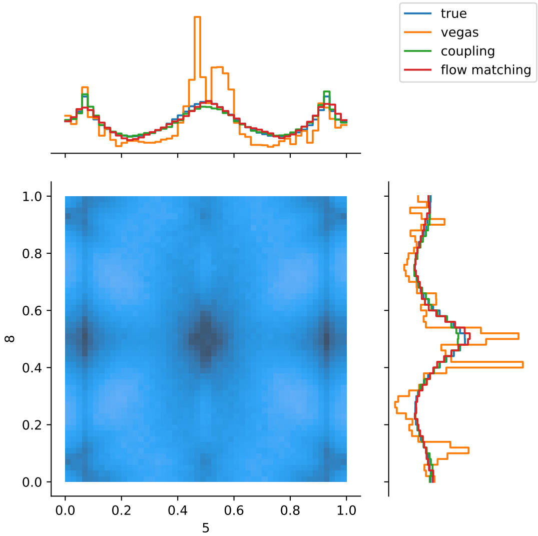
Results: $pp \to Z + 4j$ single channel with PEPPER



- partonic channel $d\bar{d} \to e^+e^-gggg$
- 16 dimensions
- distribution features non-trivial correlations
- factor 8 efficiency gain over VEGAS
- FM better than discrete flow
- high cut efficiency
- bootstrap training improves performance

Results: $pp \to Z + 4j$ single channel with PEPPER





NNLO integration with STRIPPER

- sector-improved residue subtraction scheme for higher order calc.
- collaboration with Rene Poncelet and Steffen Schumann
- developed python interface to C++ code
- select individual infrared limits in sectors where poles can be extracted and cancelled by virtual corrections

  $\to \sigma$ integral becomes sum of sector integrals

Example:

$gg \to t\bar{t}g$ single unresolved (NLO real contribution)

- example has 8 dimensions (+2 discrete inputs: sectors, polarizations)
- 2 sectors, 4 polarizations

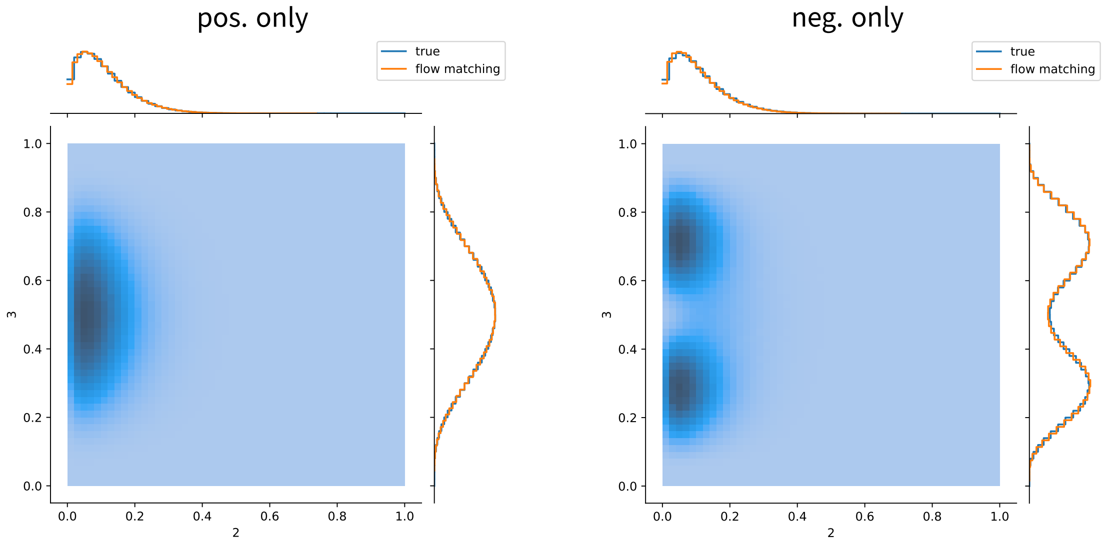
M. Czakon and D. Heymes: Nucl. Phys. B **890** (2014) 152–227

Results: $gg \to t\bar{t}g$ with STRIPPER

- have to deal with negative weights (integrand is not positive definite)
- typically one adapts $g(x)$ to $|d\sigma|$
- found that stratification into positive/negative parts is better than learning $|f|$
  $\to$ relies on flows being able to differentiate between pos/neg very efficiently
- sectors & polarizations sampled as discrete variables $\to$ conditional flow

	Integrator	MC estimate	pos. point fraction	neg. point fraction
pos.	VEGAS	1788.2 ± 1.9191	72.61%	27.38%
	IFLOW	1791.0 ± 0.52629	95.61%	4.39%
	flow matching	1790.9 ± 0.4405	98.08%	1.92%
neg.	VEGAS	-396.19 ± 0.83711	61.50%	38.49%
	IFLOW	-396.6 ± 0.17026	9.31%	90.69%
	flow matching	-396.8 ± 0.14114	4.01%	95.99%
sum	VEGAS	1392.01 ± 2.094		
	IFLOW	1394.4 ± 0.5532		
	flow matching	1393.4 ± 0.4626		
full PS	VEGAS	1393.6 ± 2.6374	71.06%	28.93%
	IFLOW	1393.4 ± 1.7923	80.86%	19.13%
	flow matching	1392.9 ± 1.8073	81.49%	18.51%

Results: $gg \to t\bar{t}g$ with STRIPPER



Outlook

- investigate scaling behaviour $\rightarrow$ go to $Z + 5$j

  When to prefer continuous flow over discrete flow?
- condition flow on flavour combinations to sample hadronic interactions with a single model
- make best use of existing multichannel samplers (as base distribution or for generating training data)
- evaluate performance for a full-featured NNLO calculation with STRIPPER

Conclusions

- Flow Matching is a surprisingly simple way of training flows with free-form Jacobian
- interesting relations to Optimal Transport theory
- training is efficient and scalable
- we find significantly improved MC variances and unweighting efficiencies for different HEP examples
- continuous flow performs slightly better than discrete flow
- normalizing flows for higher order are a promising direction for future research

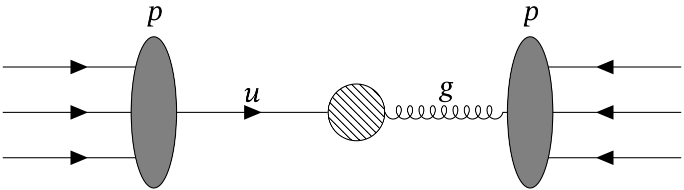
Conclusions

- Flow Matching is a surprisingly simple way of training flows with free-form Jacobian
- interesting relations to Optimal Transport theory
- training is efficient and scalable
- we find significantly improved MC variances and unweighting efficiencies for different HEP examples
- continuous flow performs slightly better than discrete flow
- normalizing flows for higher order are a promising direction for future research

Questions and suggestions are welcome!

Backup Slides

QCD cross-sections



Factorized cross-section:

$$\sigma = \sum_{i,j} \int_0^1 \mathrm{d}x_1 \int_0^1 \mathrm{d}x_2 \, f_i(x_1, \mu_F^2) \, f_j(x_2, \mu_F^2) \, \hat{\sigma}_{p_i p_j \to \{p_f\}}(x_1, x_2, \mu_F^2)$$

→ separate perturbative & non-perturbative physics

QCD cross-sections

Factorized cross-section:

$$\sigma = \sum_{i,j} \int_0^1 \mathrm{d}x_1 \int_0^1 \mathrm{d}x_2\, f_i(x_1, \mu_F^2)\, f_j(x_2, \mu_F^2)\, \hat{\sigma}_{p_i p_j \to \{p_f\}}(x_1, x_2, \mu_F^2)$$

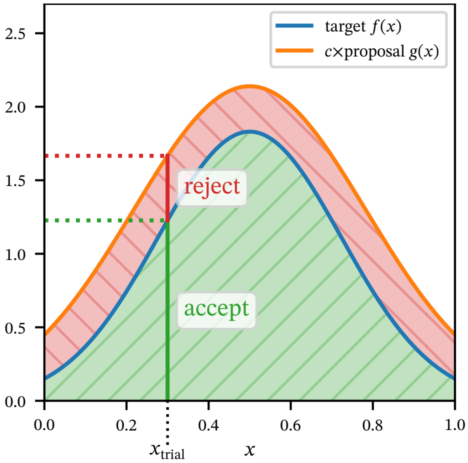
→ separate perturbative & non-perturbative physics

Partonic cross-section:

$$\mathrm{d}\hat{\sigma}_{p_i p_j \to \{p_f\}}(x_1, x_2, \mu_F^2) = \frac{1}{2E_1 E_2 |v_1 - v_2|} \left(\prod_f \frac{\mathrm{d}^3 p_f}{(2\pi)^3} \frac{1}{2E_f} \right)$$
$$\times\, |\mathcal{M}(p_i p_j \to \{p_f\})|^2\, (2\pi)^4\, \delta^{(4)}\Big(p_i + p_j - \sum p_f\Big)$$

→ can be calculated from first principles

Unweighted event generation (rejection sampling)



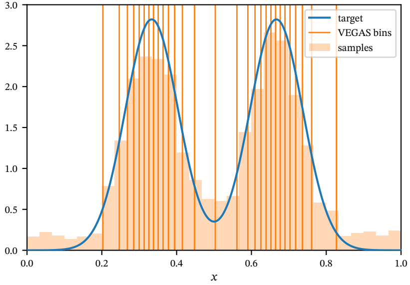
Goal

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 $p_{\text{accept}(x)} = \dfrac{w(x)}{w_{\text{max}}}$

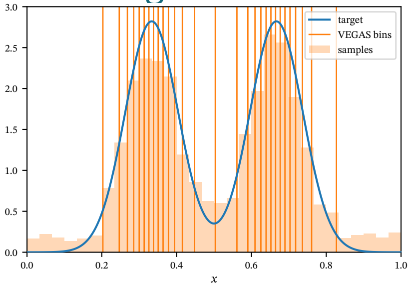
  $\rightarrow$ reduce sample size!
- unweighting efficiency: $\eta = \dfrac{\langle w \rangle}{w_{\text{max}}}$

VEGAS algorithm

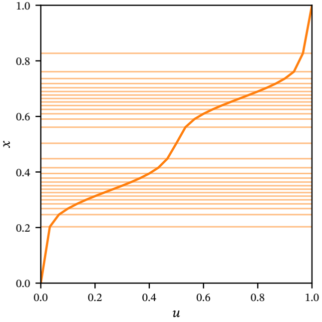


- piecewise constant density over $[0, 1)^d$
- factorized: $q(\mathbf{x}) = \prod_{i=1}^{d} q_i(x_i)$
- adapt bin boundaries to minimize variance

VEGAS algorithm



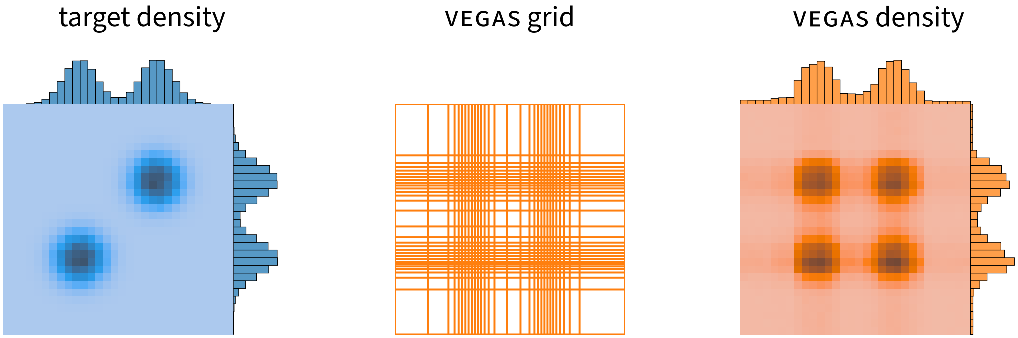
- piecewise constant density over $[0, 1)^d$
- factorized: $q(\mathbf{x}) = \prod_{i=1}^{d} q_i(x_i)$
- adapt bin boundaries to minimize variance



- piecewise linear mapping with uniform base distribution
- optimise a proposal by learning missing structure
- apply to each channel in a multichannel

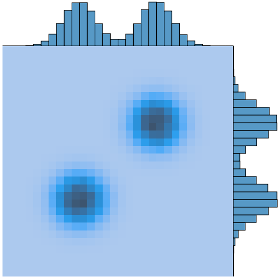
Lepage (1978), Journal of Computational Physics **27** (2)

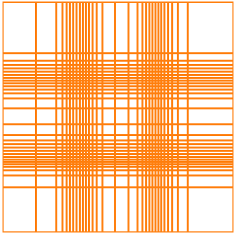
VEGAS algorithm – limitations

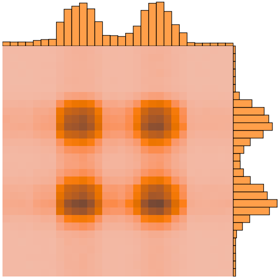


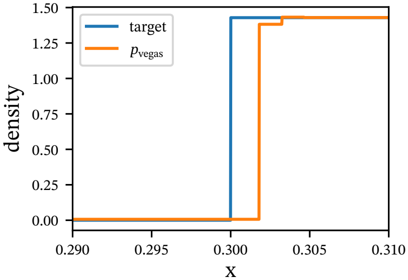
- factorised ansatz does not allow to learn correlations
  → phantom peaks

VEGAS algorithm – limitations



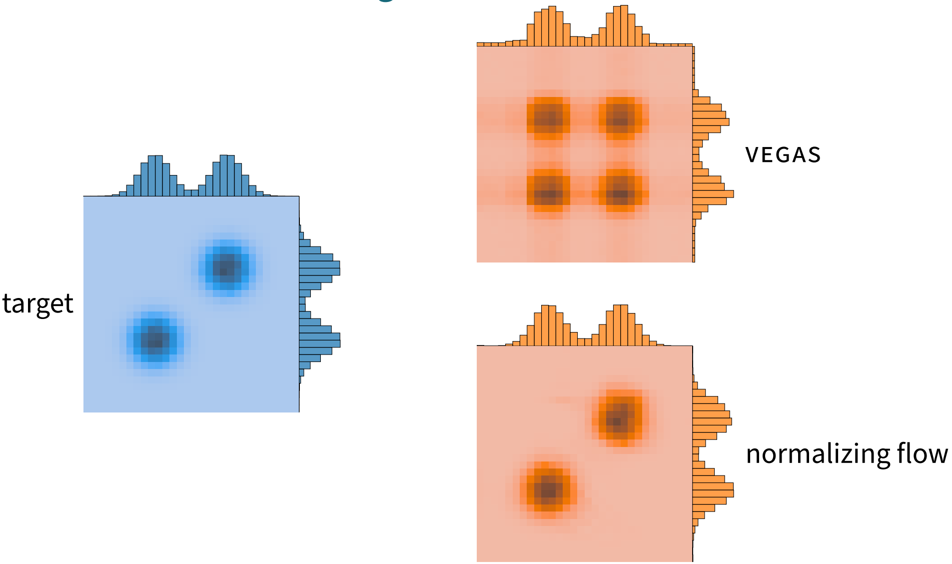






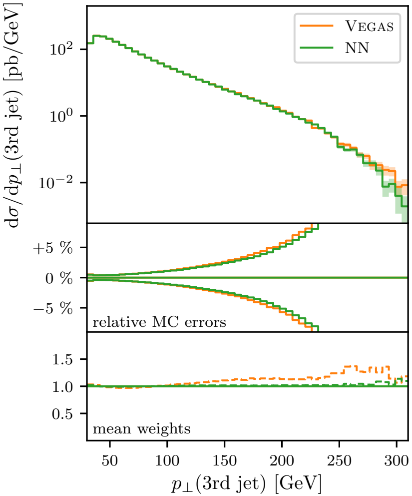
- factorised ansatz does not allow to learn correlations
  → phantom peaks
- bin boundaries are not well aligned with cuts

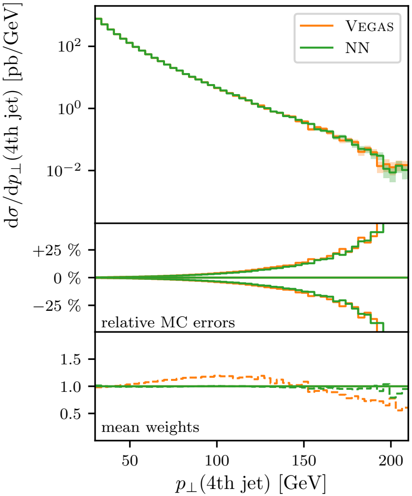
Toy example: VEGAS vs. normalizing flow



→ replacing VEGAS with normalizing flows reduces the spread of weights

Discrete flow results: unweighted events for $gg \to 4g$





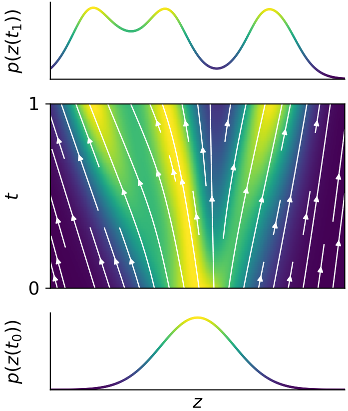
E. Bothmann, TJ, M. Knobbe, T. Schmale, S. Schumann: SciPost Phys. 8, 069 (2020)

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



© Grathwohl *et al.* ICLR 2019

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 $u_t(x)$

Recall the continuity equation:

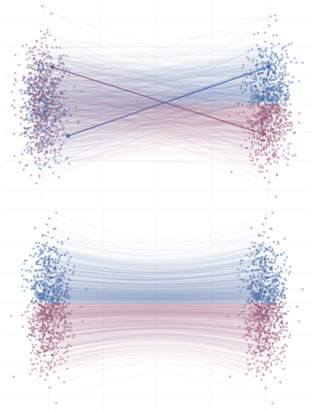
$$\frac{\mathrm{d}}{\mathrm{d}t} p_t(\mathbf{x}) + \mathrm{div}\big(p_t(\mathbf{x})\, u_t(\mathbf{x})\big) = 0$$

→ constructing p_t or u_t is equivalent

We can construct conditional probability paths with the correct marginals and their corresponding vector fields!

Lipman *et al.* ICLR 2023

Conditional probability paths



given data samples $x_1 \sim q(x_1)$, construct p_t as a mixture of simpler probability paths:

$$p_t(x) = \int p_t(x|x_1)\, q(x_1)\, \mathrm{d}x_1$$

where the conditional probability path $p_t(x|x_1)$ satisfies

$p_0(x|x_1) = p(x)$	simple base distribution
$p_1(x|x_1) = \mathcal{N}(x|x_1, \sigma^2 I)$	normal concentrated around x_1

it is generated by a conditional vector field $u_t(x|x_1)$ which has the correct marginal to generate $p_t(x)$:

$$u_t(x) = \int u_t(x|x_1) \frac{p_t(x|x_1)\, q(x_1)}{p_t(x)}\, \mathrm{d}x_1$$

(proof using continuity equation)

Gaussian probability paths

consider Gaussian probability paths:

$$p_t(x|x_1) = \mathcal{N}(x \,|\, \mu_t(x),\, \sigma_t(x_1)^2\, I)\,,$$

where

$$\mu_0(x_1) = 0 \qquad \mu_1(x1) = x_1$$
$$\sigma_0(x_1) = 1 \qquad \sigma_1(x_1) = \sigma_{\text{min}}$$

simplest flow is a linear interpolation between $t = 0$ and $t = 1$:

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

→ OT displacement map between two Gaussians

this path is generated by the vector field

$$u_t(x|x_1) = \frac{x_1 - (1 - \sigma_{\text{min}})x}{1 - (1 - \sigma_{\text{min}})t}$$