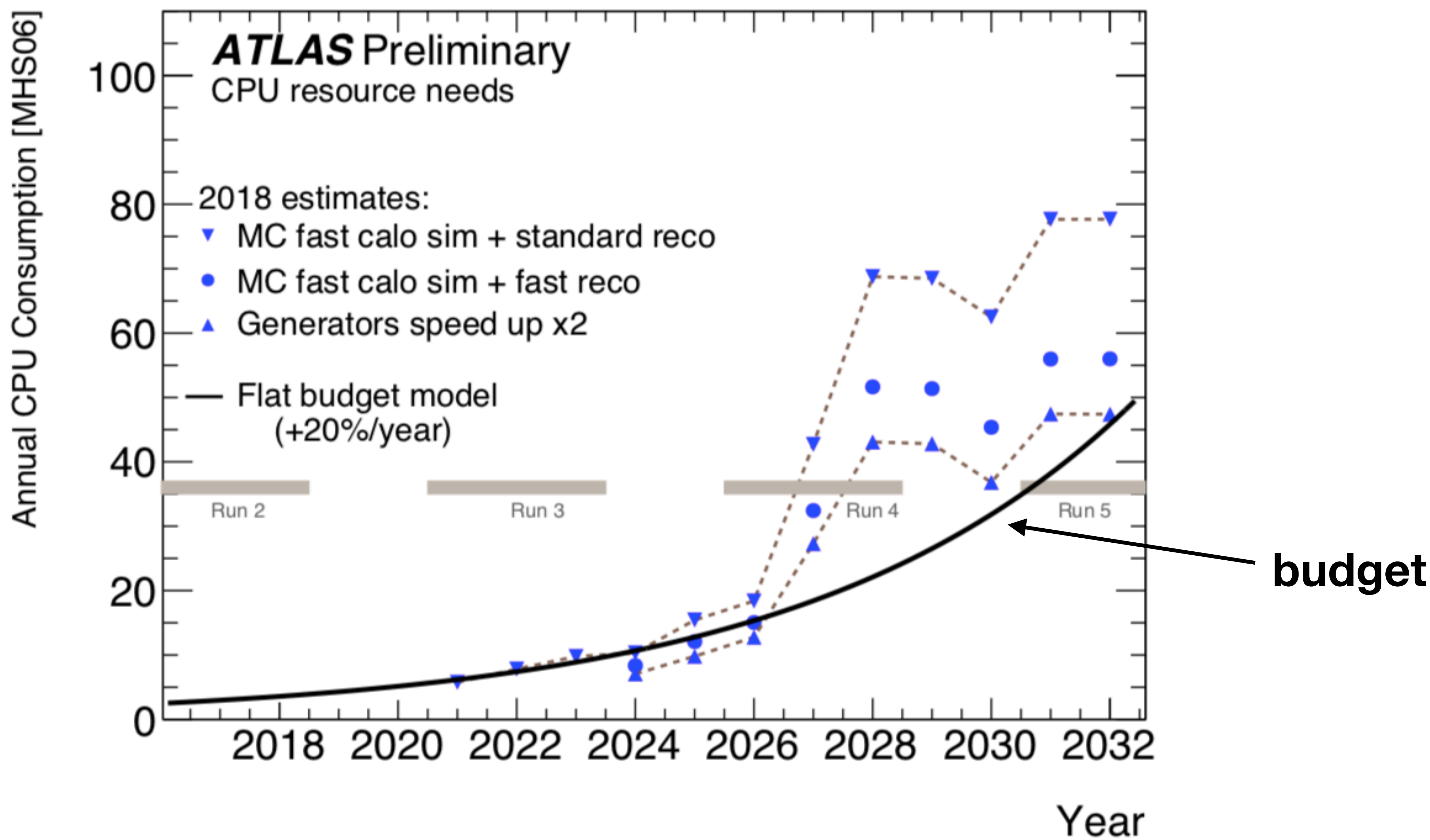
Teaching a Computer to Integrate

C. Gao, J. Isaacson, and C. Krause (2020), 2001.05486
C. Gao, S. Hoche, J. Isaacson, C. Krause, and H. Schulz (2020), 2001.10028

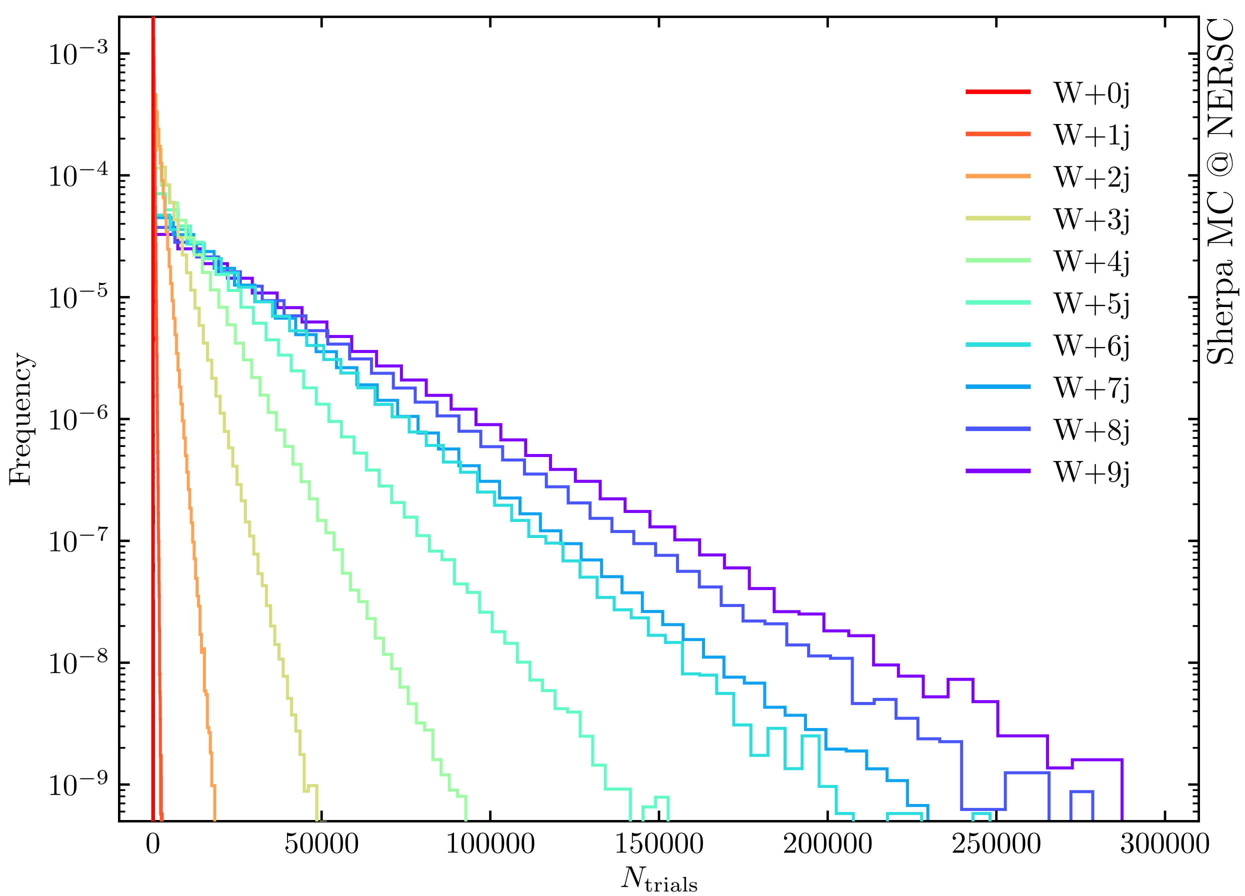
BNL Wednesday March 4th 2020

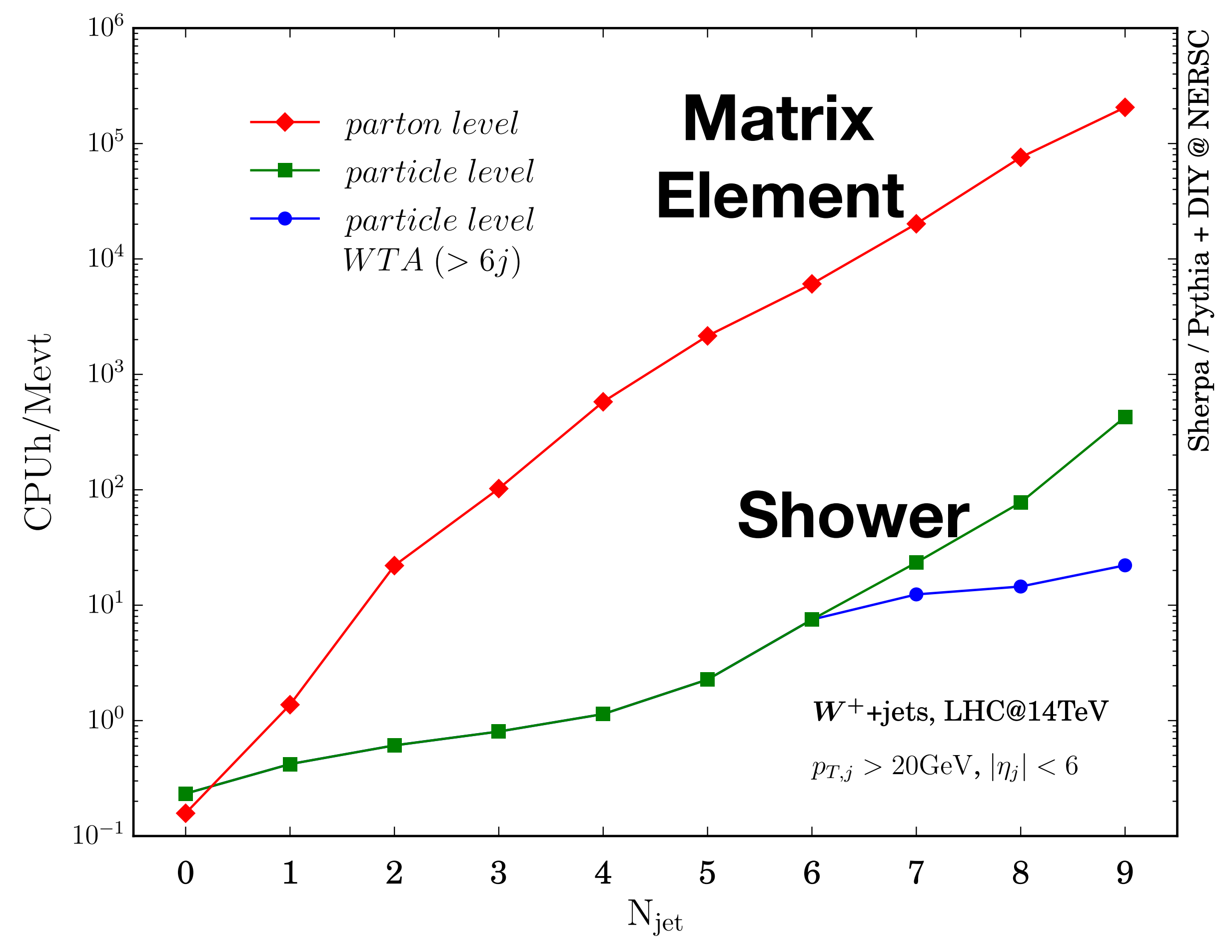
LHC requires large number of MC events



Why MC simulation so expensive

Stefan Hoche, Stefan Prestel, Holger Schulz [1905.05120;PRD]





- Matrix element evaluation is more expensive than showering
- Unweighting high-multiplicity events is expensive

Outline

- Review of MC techniques and traditional approaches
- Introduction of **i-flow**: a MC integrator based on Normalizing Flow
- Applications of **i-flow**

Monte Carlo Integration

- $I = \int_\Omega d^D x\, f\left(\overrightarrow{x}\right) \approx \frac{V}{N} \sum_i^N f\left(\overrightarrow{x}\right) \equiv V\langle f \rangle_x$

- V = volume of domain Ω

- uncertainty: $\Delta I = V\sqrt{\frac{\langle f^2 \rangle_x - \langle f \rangle_x^2}{N-1}} = \frac{\sigma_N}{\sqrt{N-1}}$

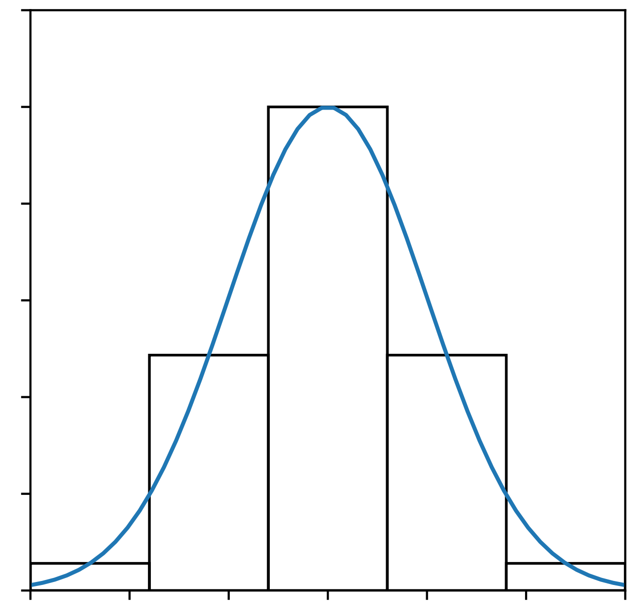
Importance Sampling

- $I = \int d^D x \, g\left(\overrightarrow{x}\right) \, \frac{f\left(\overrightarrow{x}\right)}{g\left(\overrightarrow{x}\right)} = V\langle f/g \rangle_G$
- g resembles the shape of f (ideally $g \to f/I$)
- sample uniformly in $d^D G = g\left(\overrightarrow{x}\right) d^D x$
- uncertainty: $\Delta I = V \sqrt{\frac{\langle (f/g)^2 \rangle_x - \langle f/g \rangle_x^2}{N-1}}$

MC Integrator: VEGAS

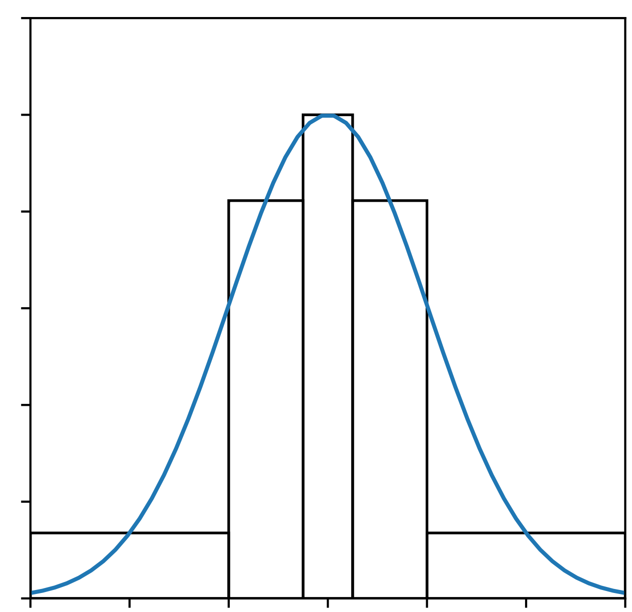
Peter Lepage 1980

- assume integrand factorizes:
  $f(\vec{x}) = f_1(x_1) f_2(x_2) \cdots f_D(x_D)$



- approximate each dimension with a histogram

- adjust the bin widths such that areas are equal

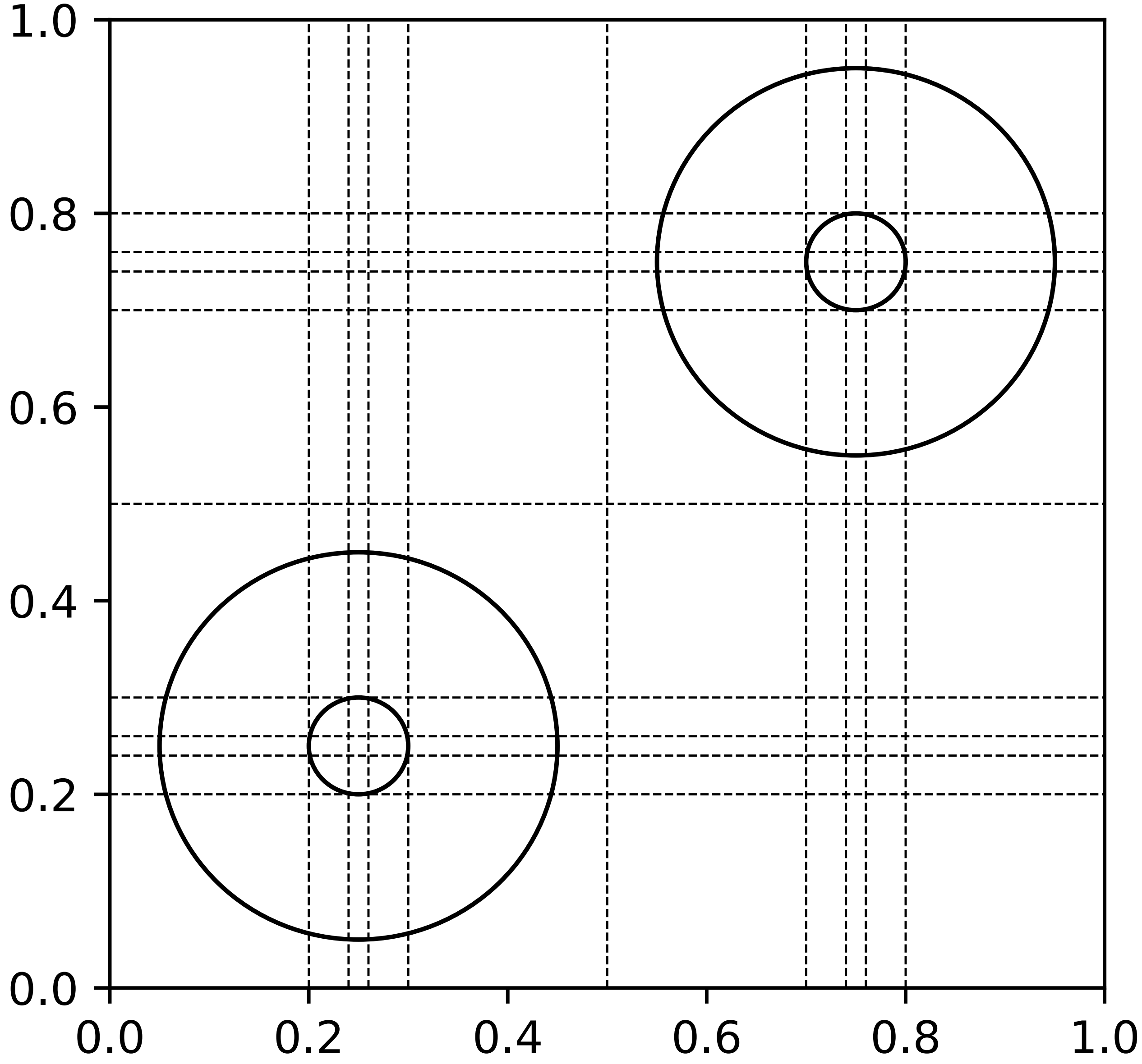


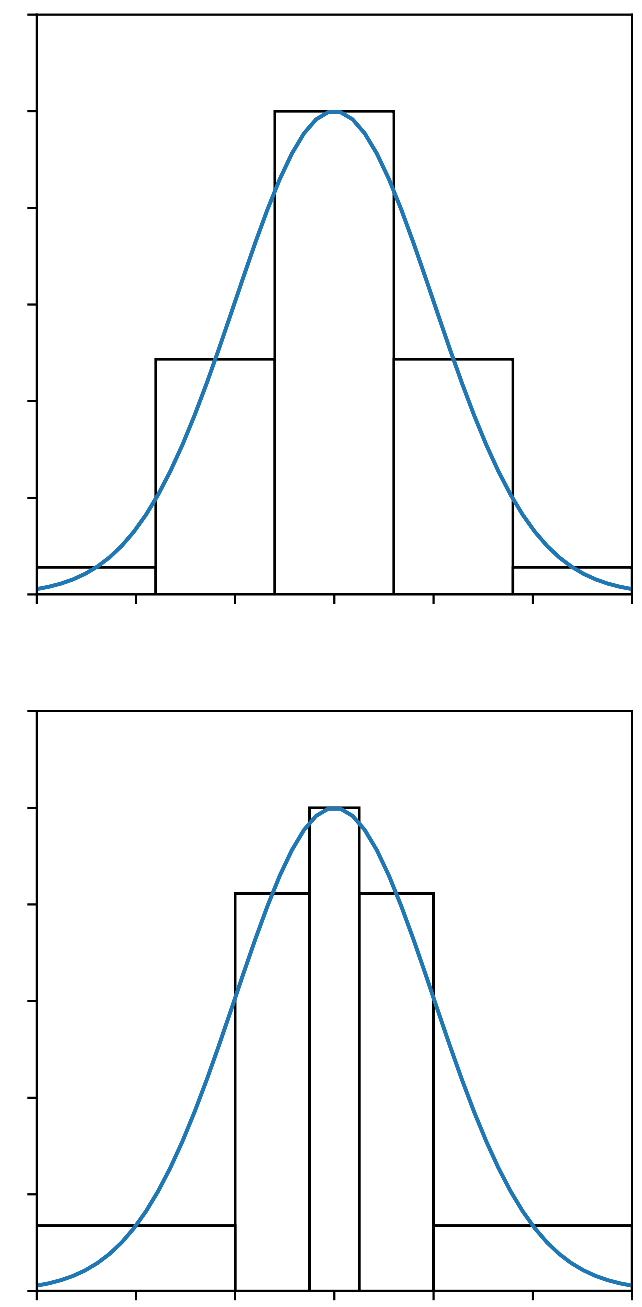
- to sample?

MC Integrator: VEGAS

Peter Lepage 1980

What if $f\left(\overrightarrow{x}\right)$ non-factorizable?





MC Integrator: FOAM

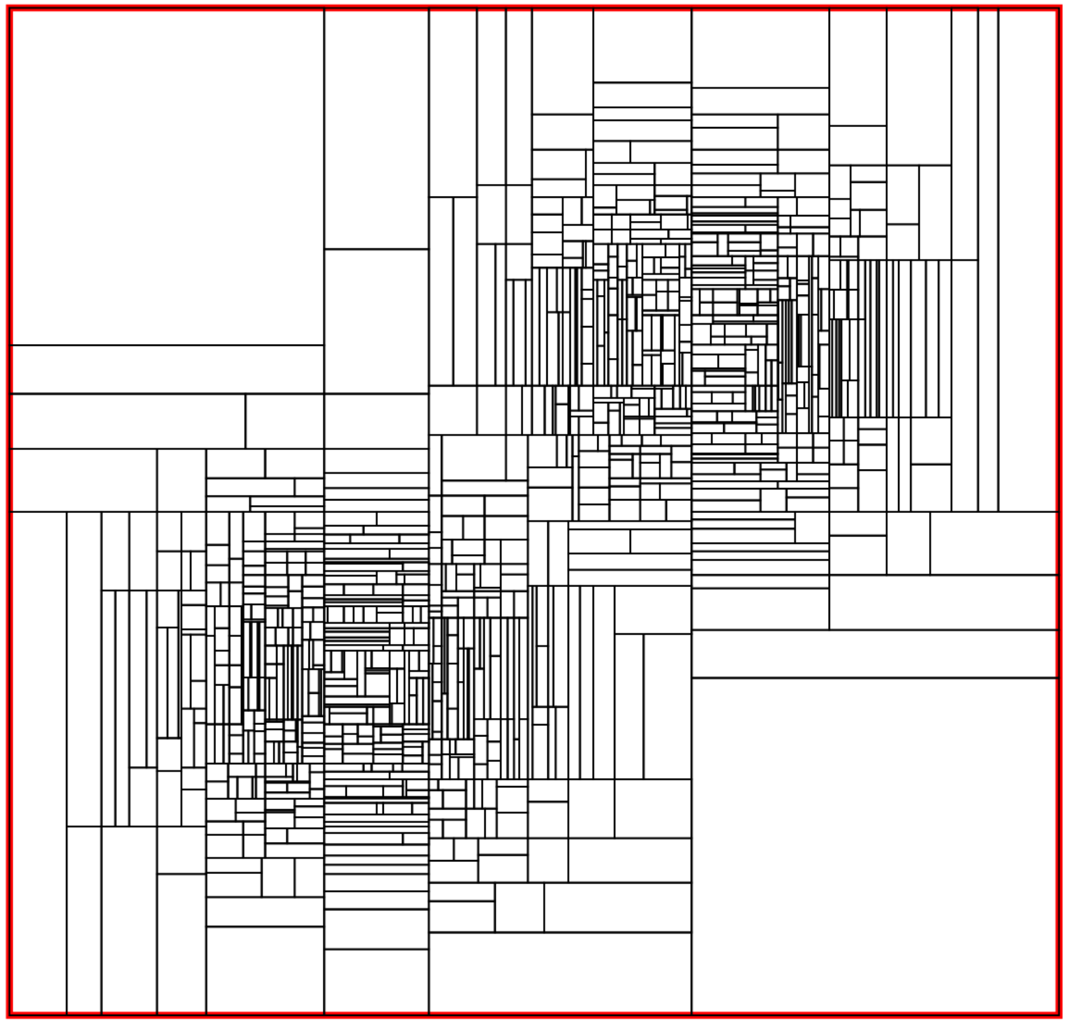
S. Jadach [arXiv:physics/0203033]

- use a cellular approximation with the first cell covering entire Ω

- build a grid by subsequent binary splits of existing cells

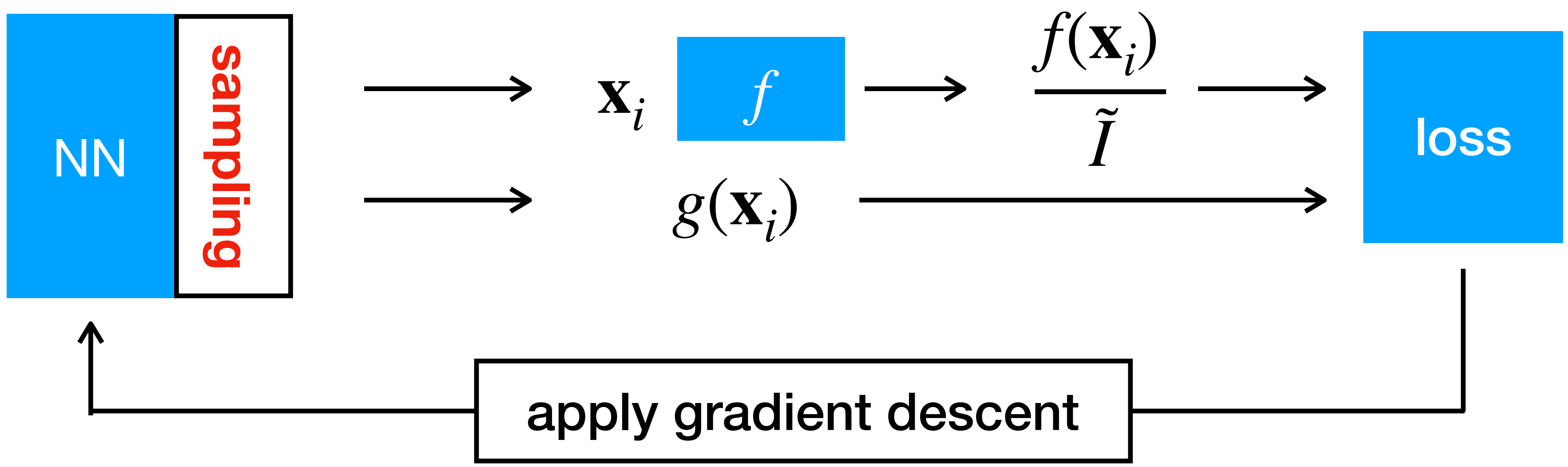
- to sample?

- but $\sim (\bar{N}_{\text{bins}})^D$ cells required



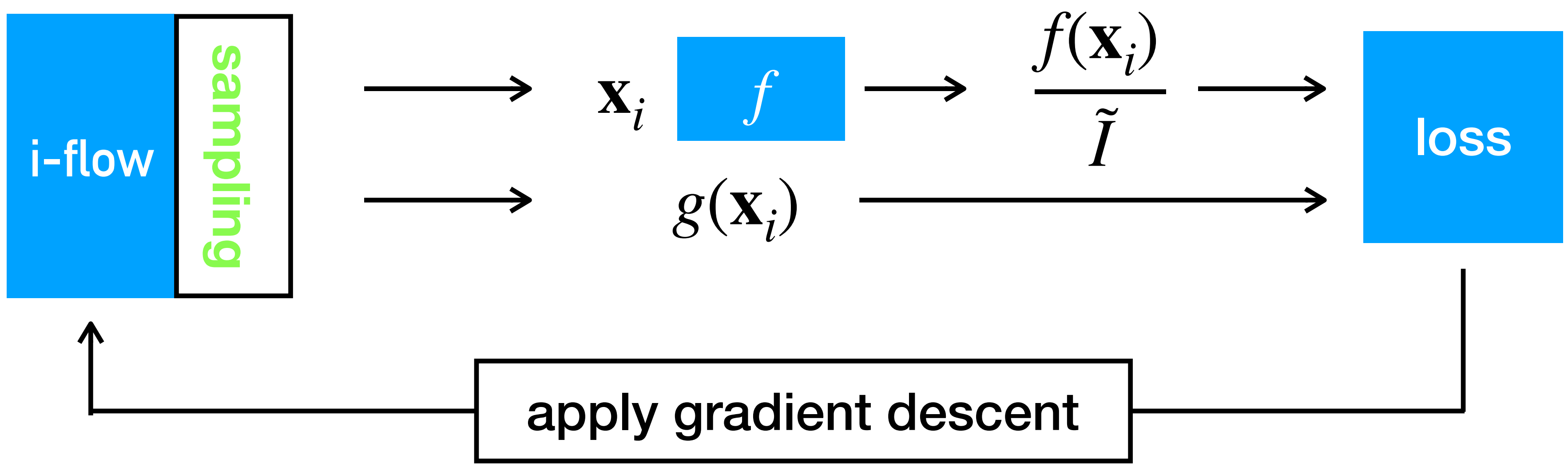
MC Integrator: NN based

Bendavid [1707.00028]
Klimek/Perelstein [1810.11509]



- g = Neural Network or BDT
- example of loss: $D_{KL} = \int dx\, f(x) \log\left(\frac{f}{g}\right)$
- but, sampling requires inverting NN (i.e. computing Jacobian determinant of a large matrix) $\sim \mathcal{O}(D^3)$

i-flow: MC Integrator with Normalizing Flows



- g = Coupling Layer based Normalizing Flow
- improves sampling efficiency $\sim \mathcal{O}(D)$
- supervised learning with an "infinite" data set

Normalizing Flow

Normalizing Flow

Rezende/Mohamed [1505.05770]
Dinh et al. [1410.8516,1605.08803]

- $\vec{x}_K = \ c_K(c_{K-1}(\cdots c_2(c_1(\vec{x}))))$, c_i is bijective

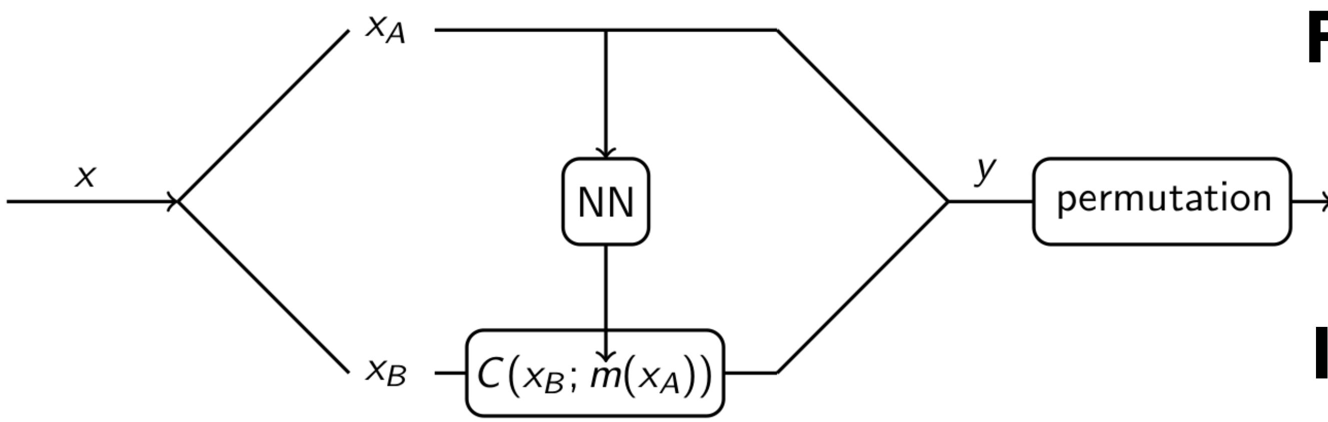
- If $x \sim g_0(x)$, then

$$x_K \sim g_K = \ g_0 \prod_{k=1}^{K} \left| \frac{\partial c_k(\vec{x}_{k-1})}{\partial \vec{x}_{k-1}} \right|^{-1} \quad , \vec{x}_0 = \vec{x}$$

- Coupling Layer is a special bijection, expressive but cheap in Jacobian computation

Coupling Layer

Dinh et al. [1410.8516,1605.08803]



Forward

$$y_A = x_A$$
$$y_B = C(x_B; m(\vec{x}_A))$$

Inverse

$$x_A = y_A$$
$$x_B = C^{-1}(y_B; m(\vec{x}_A))$$

- C is an easy, invertible Coupling Transform function
- $g_y = \left|\partial y / \partial x\right|^{-1} g_x \,, \quad \left|\frac{\partial y}{\partial x}\right|^{-1} = \left|\begin{pmatrix} \vec{1} & 0 \\ \frac{\partial C}{\partial m}\frac{\partial m}{\partial x_A} & \frac{\partial C}{\partial x_B} \end{pmatrix}\right|^{-1} = \left|\frac{\partial C(x_B; m(x_A))}{\partial x_B}\right|^{-1}$
- e.g. Affine CT: $C(x_B; s, t) = x_B \odot e^s + t \quad s, t \in \mathbb{R}^{|B|} \quad |\partial C / \partial x_B| = e^{\sum s_i}$

Coupling Transform: Piecewise Polynomial

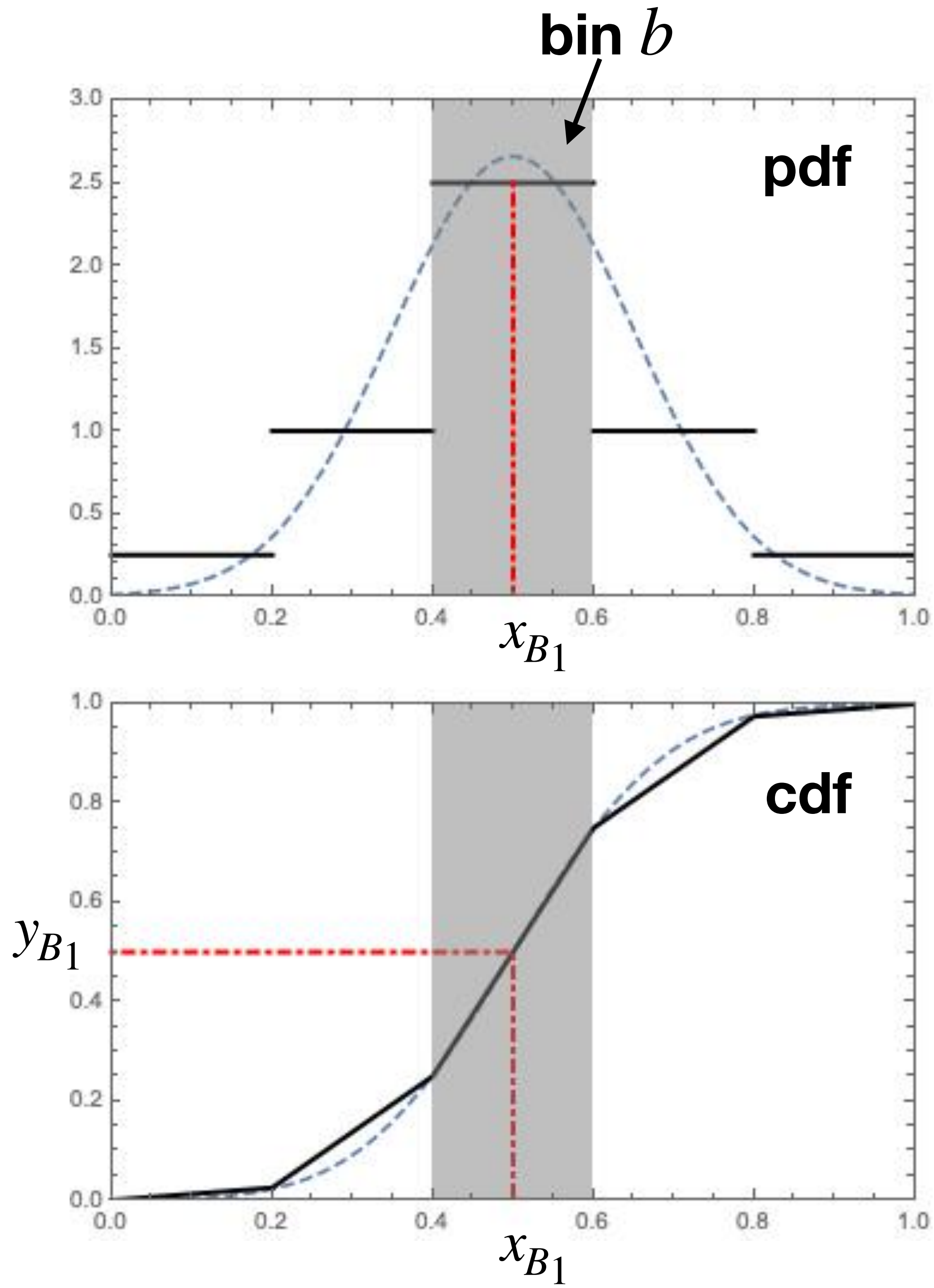
Muller et al. [1808.03856]

Forward $\begin{aligned} y_A &= x_A \\ y_B &= C(x_B; m(\vec{x}_A)) \end{aligned}$ $\quad g_y = g_x \left| \frac{\partial C(x_B; m(x_A))}{\partial x_B} \right|^{-1}$

- domain and co-domain are restricted to unit hypercube
- separability: $C(x_B; m(x_A)) = \left(C_1(x_{B_1}; m), C_2(x_{B_2}; m), \cdots, C_{|B|}(x_{B_{|B|}}; m) \right)^T$
- if $y \sim g_y$ is uniform, then C_i acts as the cumulative distribution function (CDF) of x_{B_i}: $\partial C_i(x_B; m(x_A)) = g_x \partial x_{B_i}$
- each CDF can be modeled by a piecewise monotonically increasing polynomial

Example: PW Linear

Muller et al. [1808.03856]



Given fixed bin width w, NN predicts pdf bin heights $\sim Q_i$

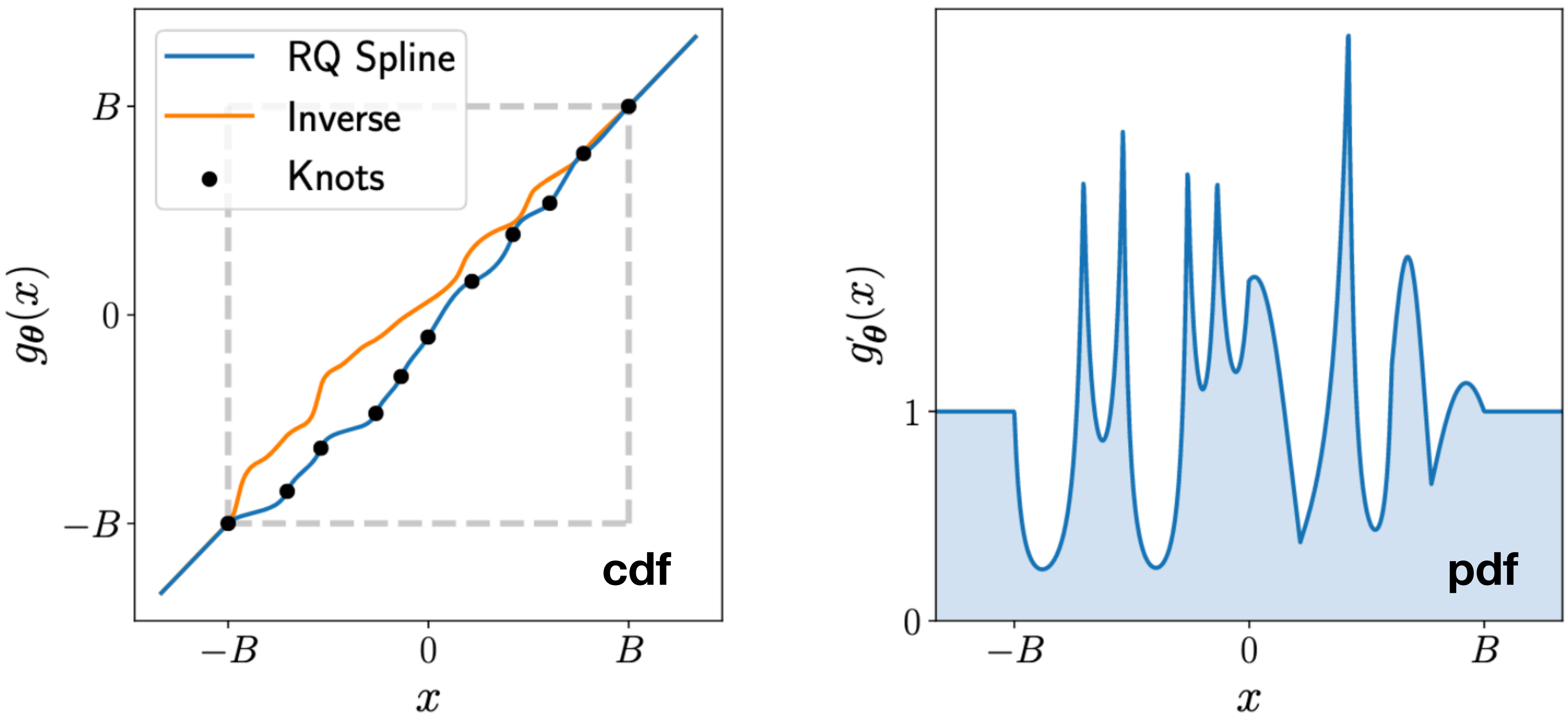
$$C_i(x_{B_i}; Q) = \alpha Q_{ib} + \sum_{k=1}^{b-1} Q_{ik}$$

$$b = \lfloor \frac{x_{B_i}}{w} \rfloor \qquad \alpha = \frac{x_{B_i} - (b-1)w}{w}$$

$$\left| \frac{\partial C(x_B; Q)}{\partial x_B} \right| = \prod_i \left| \frac{\partial C_i(x_{B_i}; Q)}{\partial x_{B_i}} \right| = \prod_i \frac{Q_{ib}}{w}$$

Coupling Transform: Rational Quadratic Spline

Durkan et al. [1906.04032]



NN predicts widths, heights, and derivatives of each knot of the spline.

How many CLs in a flow?

Normalizing Flow: $\overrightarrow{x}_K = c_K(c_{K-1}(\cdots c_2(c_1(\overrightarrow{x}))))$

c_i = NN based CL that transforms roughly half of $\overrightarrow{x}$

- capture all the correlations between every dimension
- transform (or train) each dimension equal number of times
- as few CLs as possible

How many CLs in a flow?

- minimum: 4 layers
- maximum: $2\lceil\log_2 D\rceil$, see example below.
- One means transform, zero means pass through. The transpose of the matrix and its binary negation give the max layers required.

Dimension	0	1	2	3	4	5	6	7	8	9	10	11
$\Rightarrow$	0	0	0	0	0	0	0	0	1	1	1	1
$\Rightarrow$	0	0	0	0	1	1	1	1	0	0	0	0
$\Rightarrow$	0	0	1	1	0	0	1	1	0	0	1	1
$\Rightarrow$	0	1	0	1	0	1	0	1	0	1	0	1

Finding the unique masking to capture all correlations in an $D = 12$ space

Toy Example: Integration

$$f_{gaussian}(\vec{x}) = (\alpha\sqrt{\pi})^{-n} e^{-\sum_i (x_i - \frac{1}{2})^2/\alpha^2}$$

$$f_{camel}(\vec{x}) = \frac{1}{2}(\alpha\sqrt{\pi})^{-n}\left(e^{-\sum_i (x_i - \frac{1}{3})^2/\alpha^2} + e^{-\sum_i (x_i - \frac{2}{3})^2/\alpha^2}\right)$$

- Results on 1M sample after training with 5M points
- VEGAS: 100 bins
- FOAM: 1000 points /cell
- i-flow: $2\lceil\log_2 D\rceil$ coupling layers, piecewise rational quadratic spline w. 16 bins in each dimension, DNN w. 5 layers, and other hyper-parameters

Toy Example: Integration

$$f_{gaussian}(\vec{x}) = (\alpha\sqrt{\pi})^{-n} e^{-\sum_i (x_i - \frac{1}{2})^2/\alpha^2} \qquad f_{camel}(\vec{x}) = \frac{1}{2}(\alpha\sqrt{\pi})^{-n}\left(e^{-\sum_i (x_i - \frac{1}{3})^2/\alpha^2} + e^{-\sum_i (x_i - \frac{2}{3})^2/\alpha^2}\right)$$

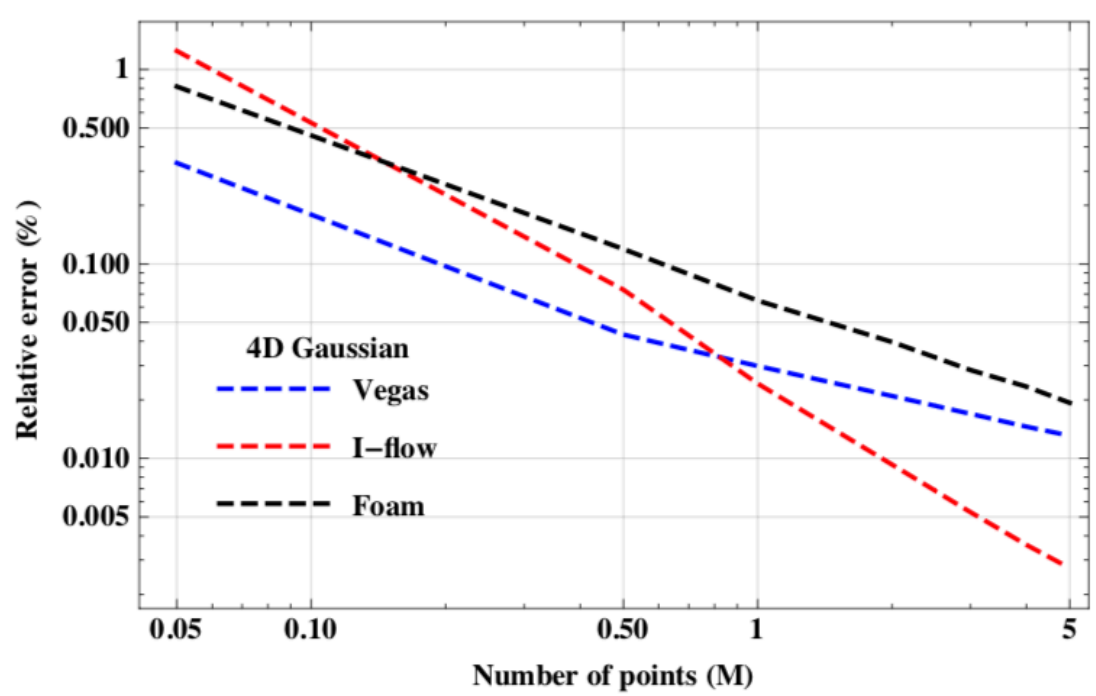
	Dim	VEGAS	Foam	i-flow	Exp
Gaussian ($\alpha = 0.2$)	2	0.99897(20)	0.99907(5)	0.99923(5)	0.999186
	4	0.99856(29)	0.99981(35)	0.99847(5)	0.998373
	8	0.99709(42)	0.99780(320)	0.99684(8)	0.996749
	16	0.99330(61)	0.72388(11428)	0.99327(23)	0.993509
Camel ($\alpha = 0.2$)	2	0.98112(89)	0.98169(5)	0.98171(4)	0.98166
	4	0.96378(222)	0.96356(30)	0.96389(25)	0.963657
	8	0.87752(759)	0.93007(142)	0.92788(44)	0.928635
	16	0.43139(25)	0.96498(17337)	0.86153(104)	0.862363

$$\frac{(I_{code} - I_{true})}{\Delta I_{code}}:$$

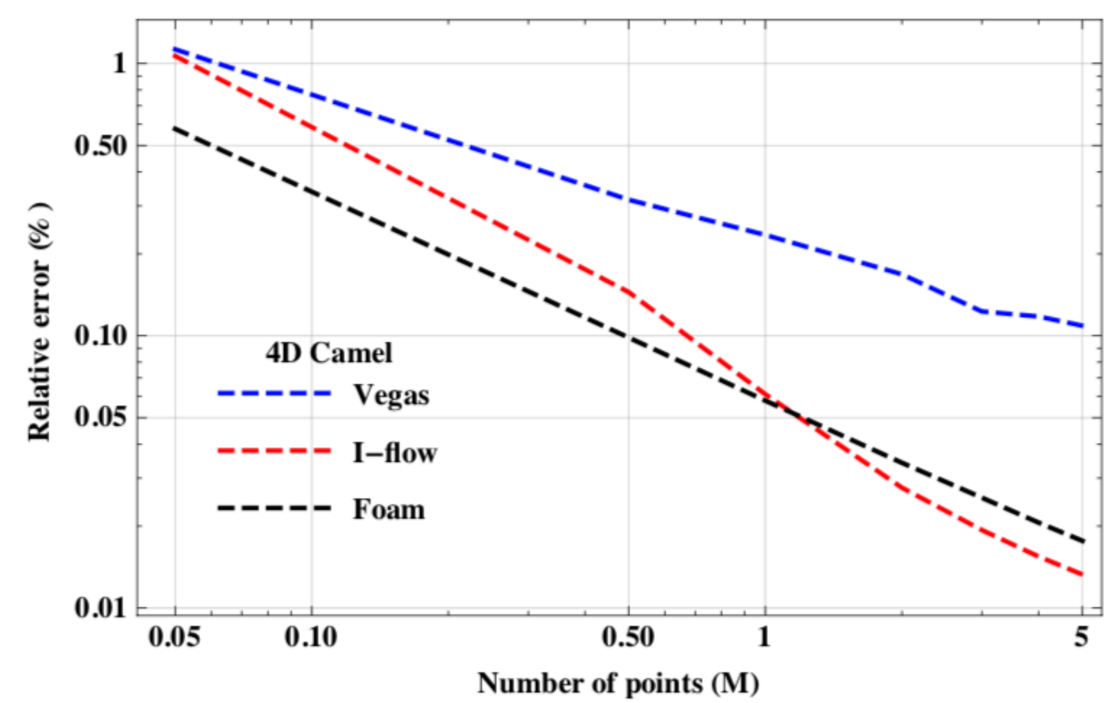
	Dim	VEGAS	Foam	i-flow
Gaussian	2	−1.08	−2.32	0.88
	4	0.65	4.11	1.94
	8	0.81	0.33	1.14
	16	−0.34	−2.36	−1.04
Camel	2	−0.61	0.6	1.25
	4	0.06	−0.32	0.93
	8	−6.73	1.01	−1.72
	16	−1723.89	0.59	−0.8

Toy Example: Integration

BUT, i-flow converges slower than VEGAS or FOAM



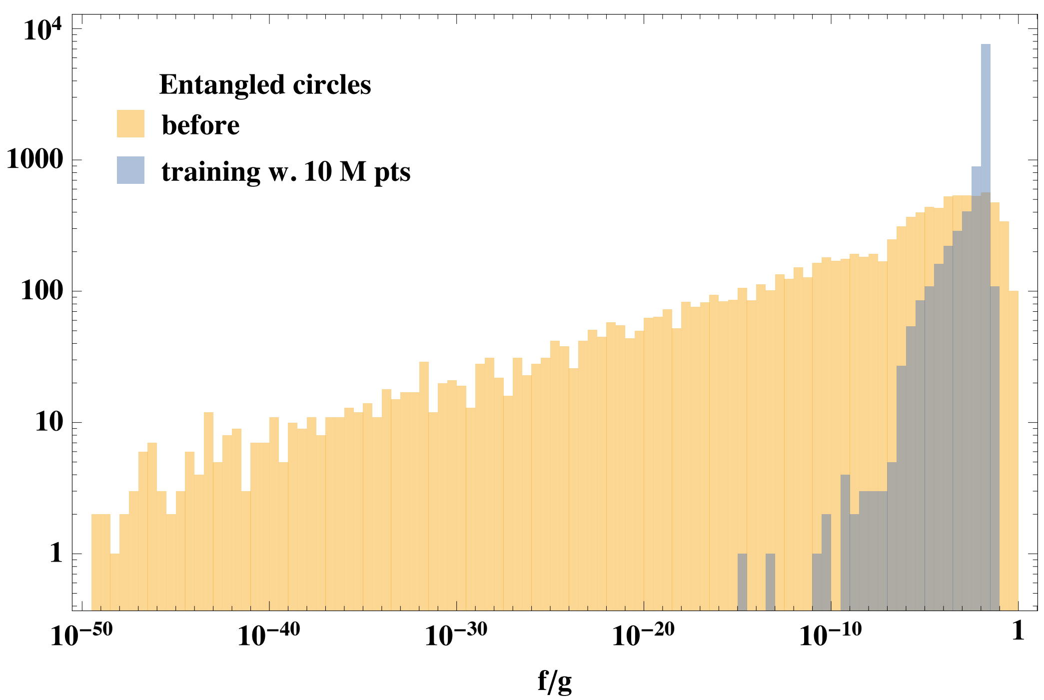
(a) 4-dimensional Gaussian

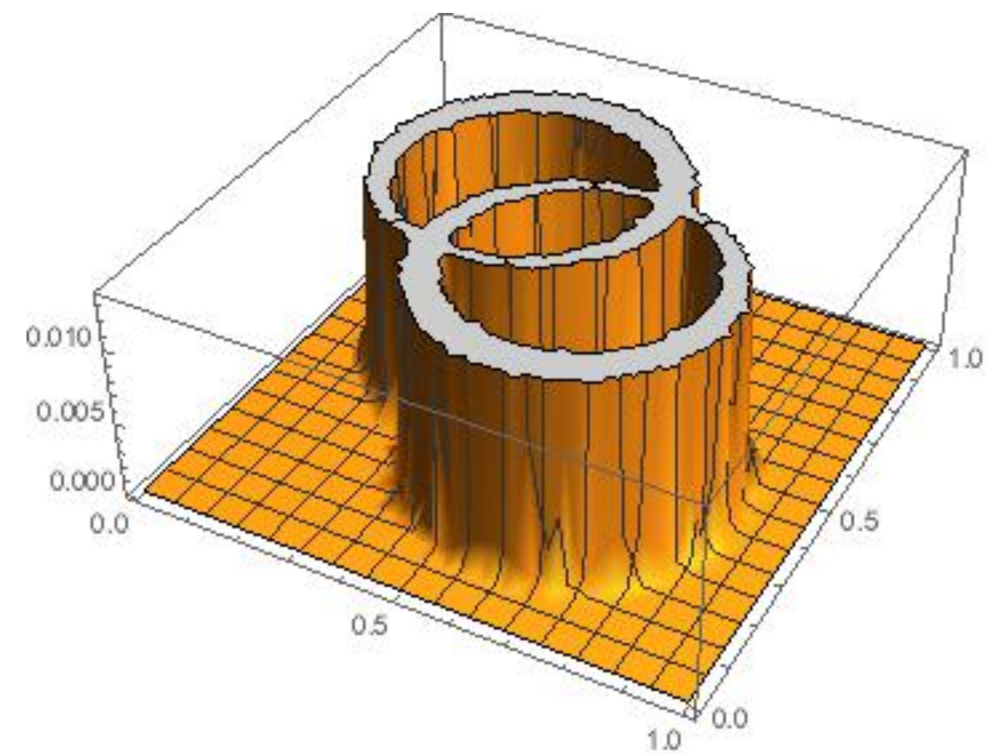


(b) 4-dimensional Camel

Toy Example: Sampling with i-flow

$$f_3(x_1, x_2) = x_2^a \exp\{-w\,|\,(x_2 - p_2)^2 + (x_1 - p_1)^2 - r^2\,|\,\}$$
$$+(1 - x_2)^a \exp\{-w\,|\,(x_2 - 1 + p_2)^2 + (x_1 - 1 + p_1)^2 - r^2\,|\,\}$$





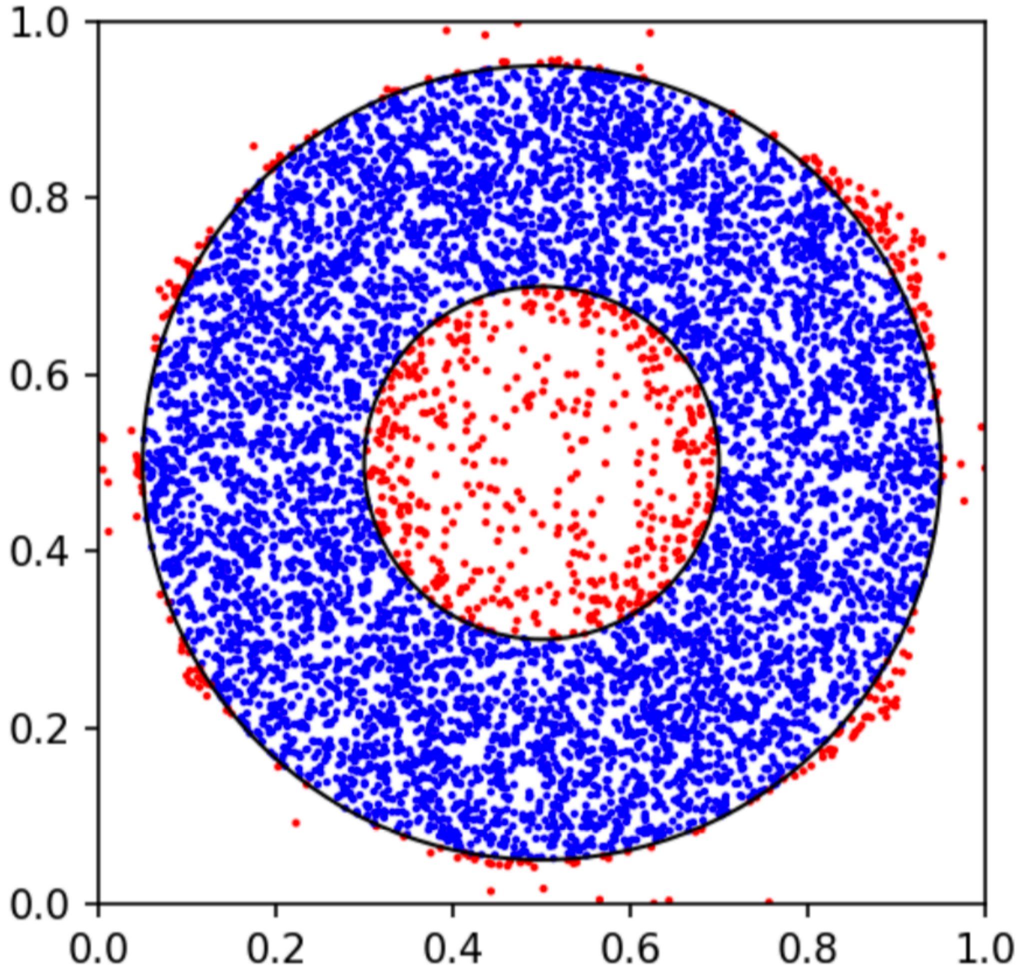
Weights of 1M points sampled,

unweighting efficiency: $\dfrac{\langle f/g \rangle}{max(f/g)} = 19.5\,\%$

$(p_1 = 0.4\ , p_2 = 0.6\ , r = 0.25\ , w = 1/0.004\ , a = 3)$

Toy Example: Sampling with i-flow

$$f_4(x_1, x_2) = \begin{cases} 1 & 0.2 < \sqrt{x_1^2 + x_2^2} < 0.45 \\ 0 & \textbf{else} \end{cases}$$



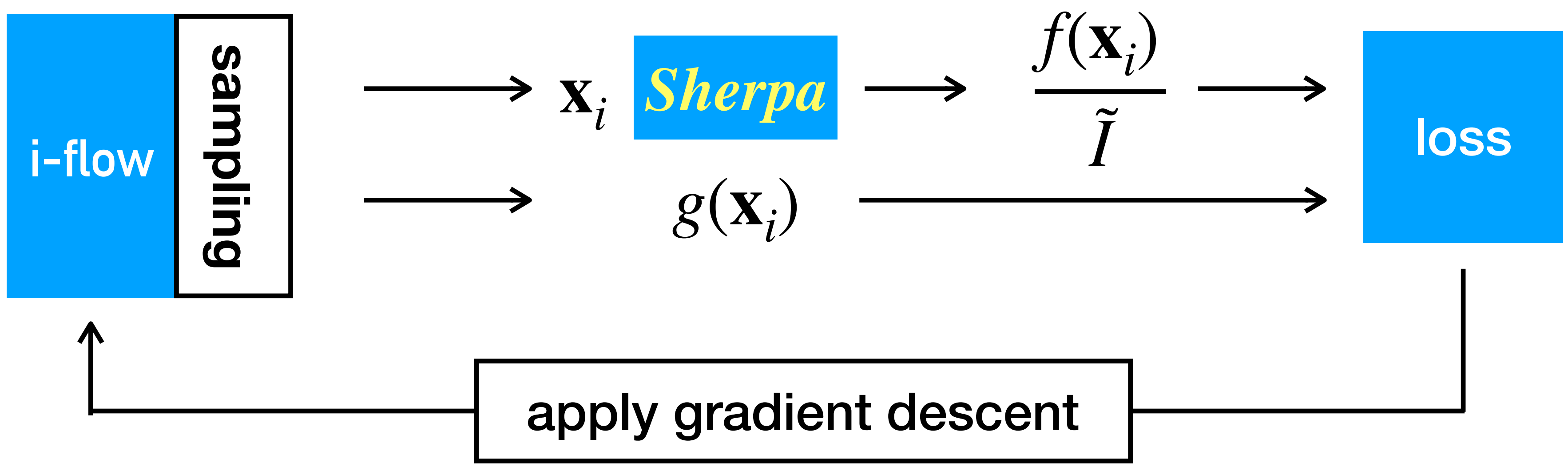
7500 points sampled:

6720 inside, 780 outside,

nearly 90% cut efficiency

Physics Application

i-flow + Sherpa: Phase Space Integration

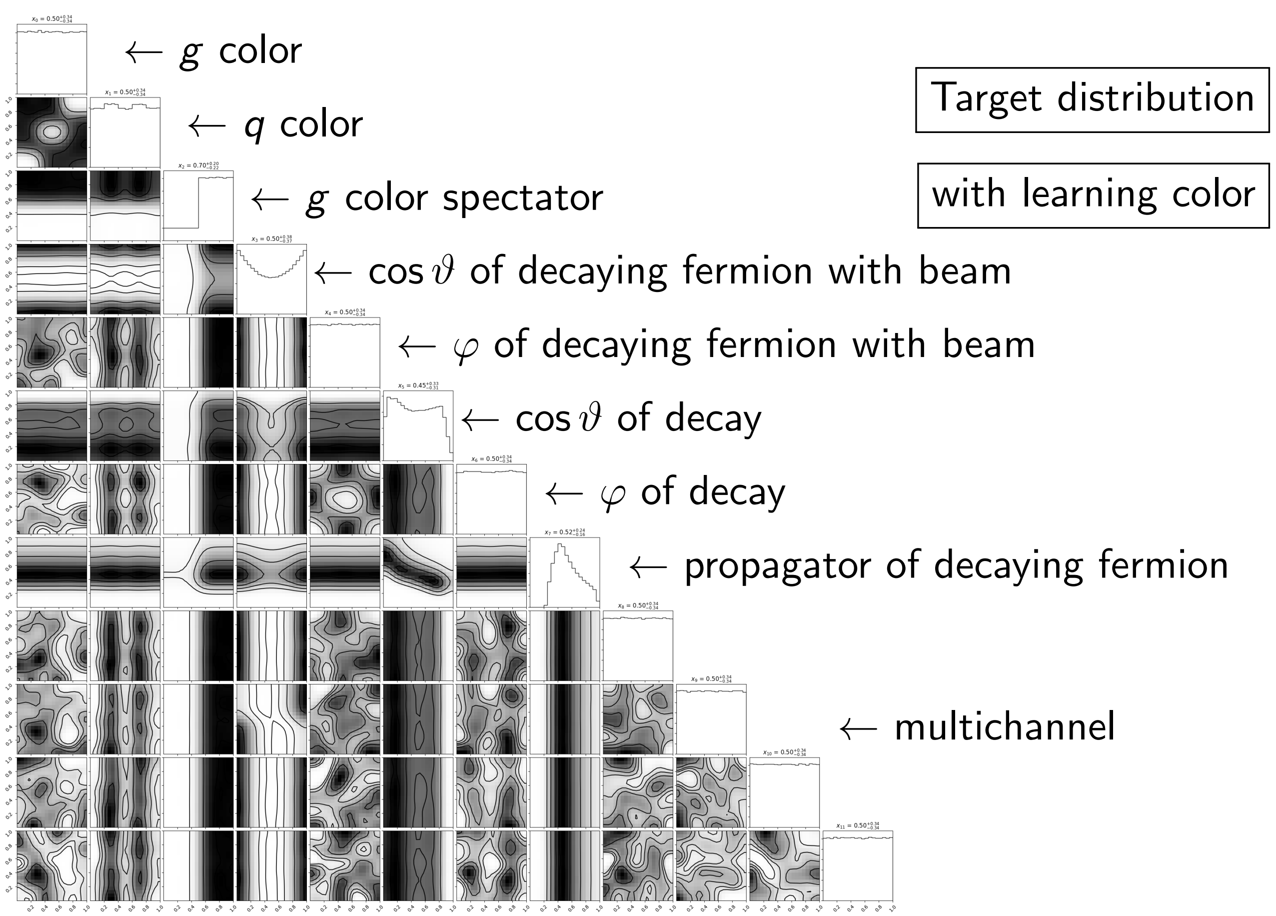


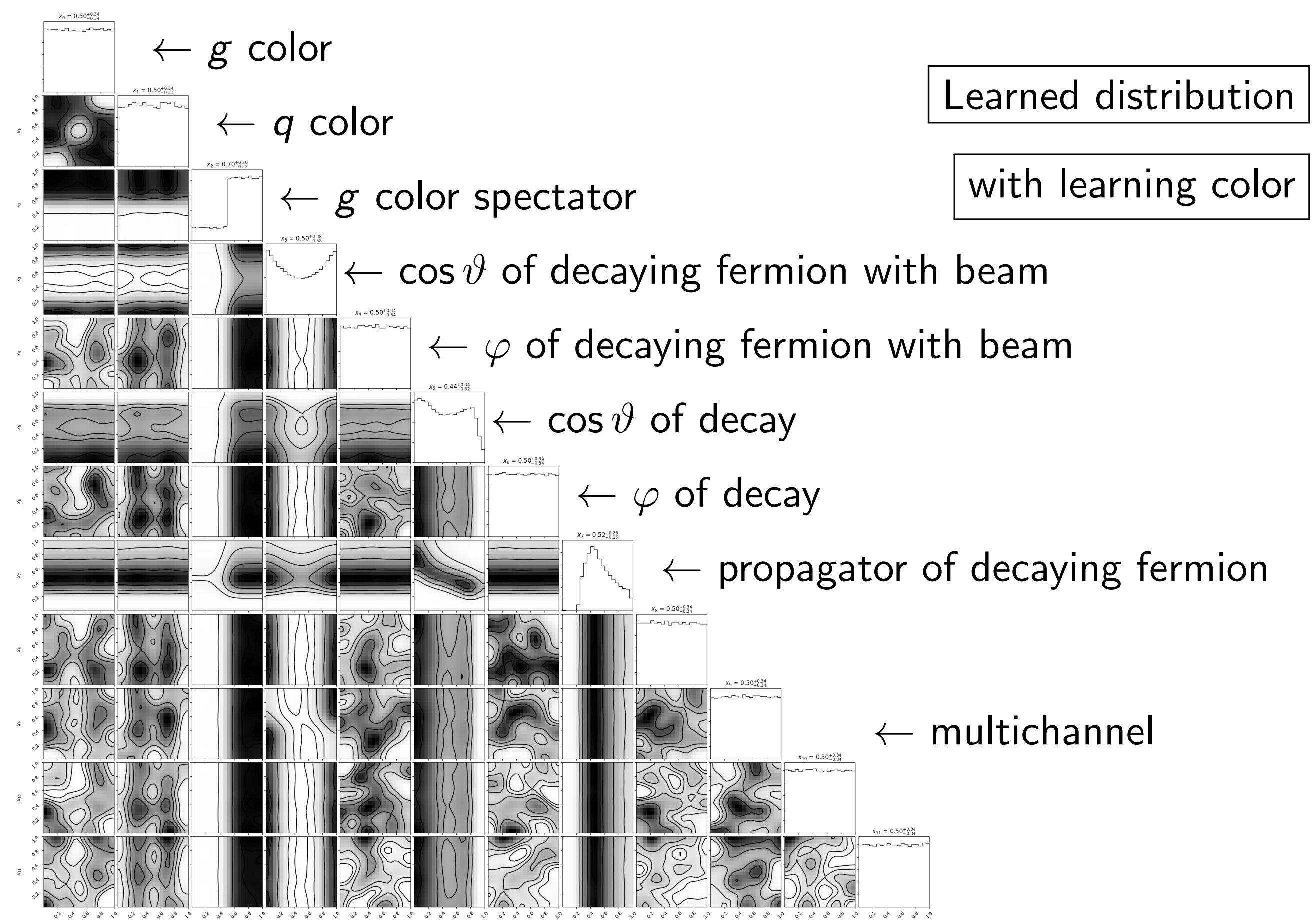
- Sherpa computes matrix element squared with color sampling

- recursive multi-channel algorithm maps the integration domain in **i-flow** (a unit hypercube) to physical variables: $n_{dim} = (\underbrace{3n_f - 4}_{\textbf{kinematics}}) + (\underbrace{n_f - 1}_{\textbf{multi-channel}}) + \underbrace{n_{ihadrons}}_{\textbf{proton pdf}}$

- integrating over final color configurations adds $2n_c - 1$ more variables

Example: $e^+e^- \to jjj$





$$\sigma_{NN} = 4887.1 \pm 4.6pb$$

$$\sigma_{Sherpa} = 4887.0 \pm 17.7pb$$

Example: $pp \to V$+jets

unweighting efficiency $\langle w\rangle/w_{\max}$		LO QCD $n=0$	$n=1$	$n=2$	$n=3$	$n=4$	NLO QCD (RS) $n=0$	$n=1$
$W^+ + n$ jets	Sherpa	$2.8 \cdot 10^{-1}$	$3.8 \cdot 10^{-2}$	$7.5 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$	$8.3 \cdot 10^{-4}$	$9.5 \cdot 10^{-2}$	$4.5 \cdot 10^{-3}$
	NN+NF	$6.1 \cdot 10^{-1}$	$1.2 \cdot 10^{-1}$	$1.0 \cdot 10^{-3}$	$1.8 \cdot 10^{-3}$	$8.9 \cdot 10^{-4}$	$1.6 \cdot 10^{-1}$	$4.1 \cdot 10^{-3}$
	Gain	2.2	3.3	1.4	1.2	1.1	1.6	0.91
$W^- + n$ jets	Sherpa	$2.9 \cdot 10^{-1}$	$4.0 \cdot 10^{-2}$	$7.7 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$	$9.7 \cdot 10^{-4}$	$1.0 \cdot 10^{-1}$	$4.5 \cdot 10^{-3}$
	NN+NF	$7.0 \cdot 10^{-1}$	$1.5 \cdot 10^{-1}$	$1.1 \cdot 10^{-2}$	$2.2 \cdot 10^{-3}$	$7.9 \cdot 10^{-4}$	$1.5 \cdot 10^{-1}$	$4.2 \cdot 10^{-3}$
	Gain	2.4	3.3	1.4	1.1	0.82	1.5	0.91
$Z + n$ jets	Sherpa	$3.1 \cdot 10^{-1}$	$3.6 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$	$4.7 \cdot 10^{-3}$		$1.2 \cdot 10^{-1}$	$5.3 \cdot 10^{-3}$
	NN+NF	$3.8 \cdot 10^{-1}$	$1.0 \cdot 10^{-1}$	$1.4 \cdot 10^{-2}$	$2.4 \cdot 10^{-3}$		$1.8 \cdot 10^{-3}$	$5.7 \cdot 10^{-3}$
	Gain	1.2	2.9	0.91	0.51		1.5	1.1

TABLE II: Unweighting efficiencies at the LHC at $\sqrt{s} = 14$ TeV using the NNPDF 3.0 NNLO PDF set and a correspondingly defined strong coupling. Jets are identified using the k_T clustering algorithm with $R = 0.4$, $p_{T,j} > 20$ GeV and $|\eta_j| < 6$. In the case of Z/γ^* production, we also apply the invariant mass cut $66 < m_{ll} < 116$GeV.

Why does it not work so well for $n \geq 2$ jets?

- Discrete variables like multi-channel or color can not be modeled well by a continuous distribution.

- After all, it is a MC technique, to get the corners right requires some luck or a very large number of samples to train, which then runs into memory problem.

Anomaly Detection w. i-flow

Nachman, Shih. [2001.04990]

- Pick an observable $\mathcal{O}$ reconstructed from data and define a signal region (SR), e.g. invariant mass of di-jets

- Learn two distributions on a set of other kinematic features $\{x_i\}$for SR and the side bands (SB) conditioned on $\mathcal{O}$: $P_{SR}(x_i \,|\, \mathcal{O} \in SR),\ P_{SB}(x_i \,|\, \mathcal{O} \notin SR)$

- Interpolate P_{SB} into SR and calculate the likelihood ratio $$R = \frac{P_{SR}(x_i \,|\, \mathcal{O} \in SR)}{P_{SB}(x_i \,|\, \mathcal{O} \in SR)}$$

- $R \approx 1$ for SM backgrounds but bigger than 1 for BSM events.

Conclusion

- as a MC Integrator, compared to VEGAS and FOAM, **i-flow** is the only one that performs consistently up to high dimensions $(D \gtrsim 8)$
- as a MC event generator, the unweighting efficiency exceeds that of traditional methods by a factor of 2 to 3 in simple processes (V+0,1jet)
- code available at https://gitlab.com/i-flow/i-flow

Back-up: more examples

	Dim	VEGAS	Foam	i-flow	Exp
Gaussian	2	0.99897(20)	0.99907(5)	0.99923(5)	0.999186
	4	0.99856(29)	0.99981(35)	0.99847(5)	0.998373
	8	0.99709(42)	0.99780(320)	0.99684(8)	0.996749
	16	0.99330(61)	0.72388(11428)	0.99327(23)	0.993509
Camel	2	0.98112(89)	0.98169(5)	0.98171(4)	0.98166
	4	0.96378(222)	0.96356(30)	0.96389(25)	0.963657
	8	0.87752(759)	0.93007(142)	0.92788(44)	0.928635
	16	0.43139(25)	0.96498(17337)	0.86153(104)	0.862363
Entangled circles	2	0.013675(44)	0.013685(3)	0.013692(8)	0.0136848
Ring w. cuts	2	0.51122(57)	0.51083(16)	0.51062(21)	0.510508
Scalar-top-loop	3	1.93687(32)e−10	1.93699(1)e−10	1.93696(1)e−10	1.936964e−10

$$\frac{(I_{code} - I_{true})}{\Delta I_{code}}:$$

	Dim	VEGAS	Foam	i-flow
Gaussian	2	−1.08	−2.32	0.88
	4	0.65	4.11	1.94
	8	0.81	0.33	1.14
	16	−0.34	−2.36	−1.04
Camel	2	−0.61	0.6	1.25
	4	0.06	−0.32	0.93
	8	−6.73	1.01	−1.72
	16	−1723.89	0.59	−0.8
Entangled circles	2	−0.23	0.07	0.87
Ring w. cuts	2	1.24	2.01	0.53
Scalar-top-loop	3	−0.29	2.6	−0.45

Back-up: Hyper-parameter Optimization for $W + 1$jet

Parameter	$c_{\min}$	$c_{\max}$	prior	c_{best}
learning rate	10^{-5}	10^{-2}	log	$4.55 \cdot 10^{-4}$
LR decay	0	1	lin	0.534
LR step size	2	10	lin	5
$N_{\text{nodes}}^{\max}$	2^5	2^9	lin	2^9

Parameter	$c_{\min}$	$c_{\max}$	prior	c_{best}
N_{samples}	1000	10000	log	4148
N_{epochs}	500	5000	log	3824
N_{bins}	10	100	log	16
N_{layers}	6	10	lin	8