Fermilab U.S. DEPARTMENT OF ENERGY Office of Science

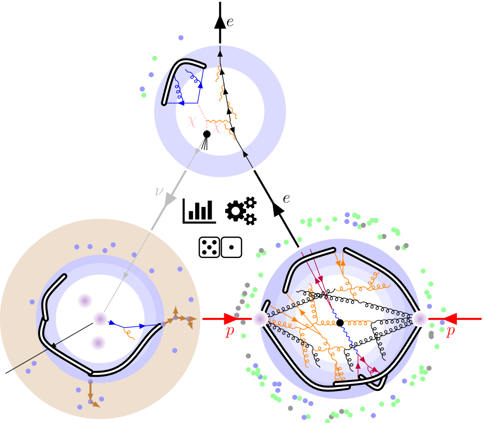
Monte Carlo for Theory and Event Generation in HEP

Joshua Isaacson

Argonne Mini-Workshop on Monte Carlo Methods

18 May 2023

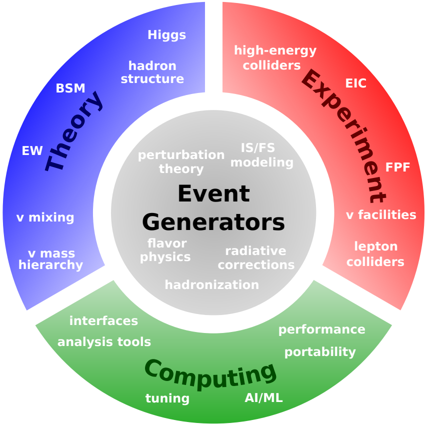
Introduction



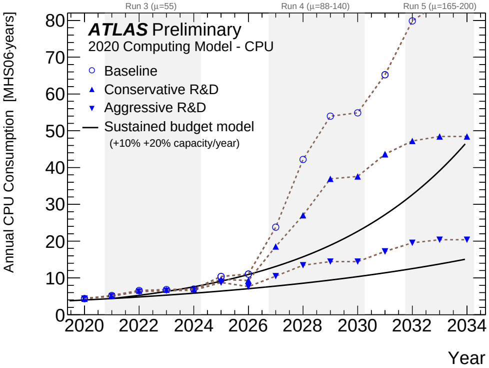
- The success of HEP experiments critically relies on advancements in physics modelling and computational techniques, driven by a close dialogue between large experimental collaborations and small teams of event generator authors.
- Development, validation, and long-term support of event generators requires a vibrant research program at the interface of theory, experiment, and computing

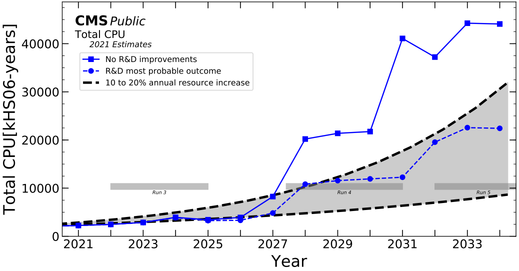
Why do we need generators?

- Precision understanding of Standard Model
- Ability to model BSM processes
- Essential role in planning and design of future experiments
- Connects the theory and experimental community
- Modelling non-perturbative effects



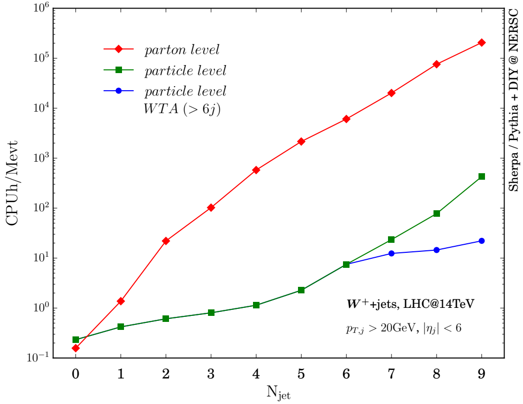
Computing Bottlenecks

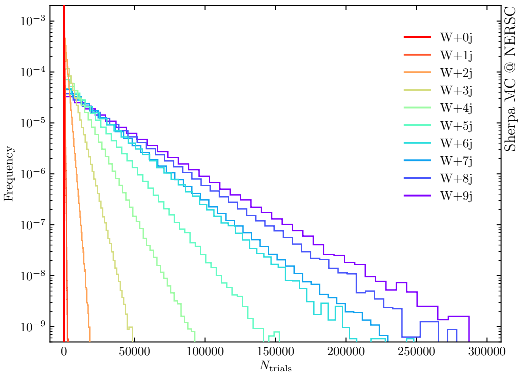




- LHC requires large number of Monte Carlo events
- Due to CPU costs, MC statistics will become significant uncertainty

Motivation





[S. Höche, S. Prestel, H. Schulz, 1905.05120]

- Time to generate an event dominated by hard process not shower
- Large computational cost for unweighting at high multiplicity

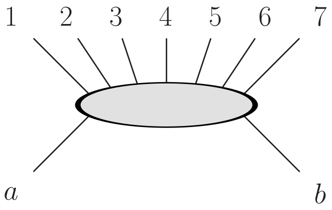
General Problem

Differential Cross Section

$$d\sigma = dx_a dx_b d\Phi_n(a, b; 1, \ldots, n)|M|^2$$

Phase Space

$$d\Phi_n(a, b; 1, \ldots, n) = \left[\prod_{i=1}^{n} \frac{d^3\vec{p}_i}{(2\pi)^3\, 2E_i}\right] (2\pi)^4 \delta^{(4)}\left(p_a + p_b - \sum_{i=1}^{n} p_i\right).$$



Importance Sampling

No Importance Sampling

$$\int_0^1 f(x)dx \xrightarrow{MC} \frac{1}{N}\sum_i f(x_i) \quad \text{iid } \mathcal{U}(0,1)$$

Importance Sampling

$$\int_0^1 \frac{f(x)}{q(x)}q(x)dx \xrightarrow{MC} \frac{1}{N}\sum_i \frac{f(x_i)}{q(x_i)} \quad \text{iid } q(x)$$

Importance Sampling

No Importance Sampling

$$\int_0^1 f(x)dx \xrightarrow{MC} \frac{1}{N}\sum_i f(x_i) \quad \text{iid } \mathcal{U}(0,1)$$

Importance Sampling

$$\int_0^1 \frac{f(x)}{q(x)}q(x)dx \xrightarrow{MC} \frac{1}{N}\sum_i \frac{f(x_i)}{q(x_i)} \quad \text{iid } q(x)$$

Goal: Choose a function $q(x)$ such that $\frac{f(x)}{q(x)} \approx 1$.

- Best is $q(x) = f(x)$, requires analytic inverse of CDF
- Acceptable to get close enough by fitting $f(x)$ to some assumed form

VEGAS

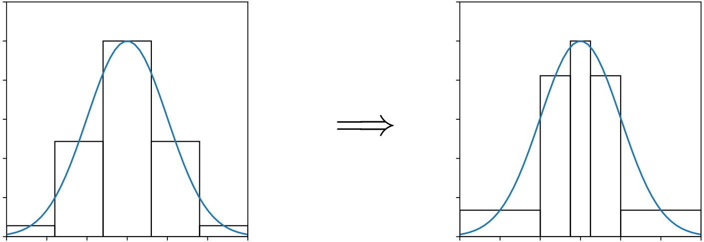
The VEGAS algorithm [P. Lepage 1980]

- Assumes integrand factorizes; i.e. $f(\vec{x}) = f_0(x_0)\cdots f_n(x_n)$

VEGAS

The VEGAS algorithm [P. Lepage 1980]

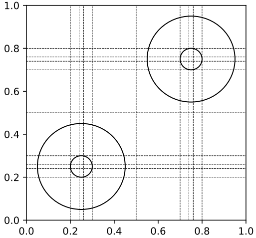
- Assumes integrand factorizes; i.e. $f(\vec{x}) = f_0(x_0) \cdots f_n(x_n)$
- Adapts bin edges such that area of each bin is the same



VEGAS

The VEGAS algorithm [P. Lepage 1980]

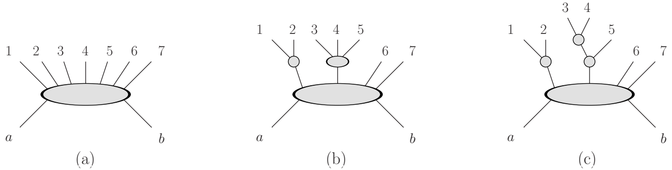
- Assumes integrand factorizes; i.e. $f(\vec{x}) = f_0(x_0) \cdots f_n(x_n)$
- Adapts bin edges such that area of each bin is the same
- Issues with features aligned along diagonals



Multichannel

Phase Space Factorization

$$\mathrm{d}\Phi_n(a,b;1,\ldots,n) = \mathrm{d}\Phi_{n-m+1}(a,b;\pi,m+1,\ldots,n)\,\frac{\mathrm{d}s_\pi}{2\pi}\,\mathrm{d}\Phi_m(\pi;1,\ldots,m)\,,$$

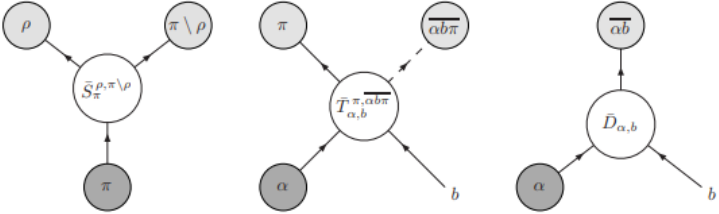


Multichanneling

Many different choices to factorize the phase space. Rewrite integral as:

$$\int f(x)dx = \sum_i \alpha_i \int f(x_i)dx_i\,, \qquad \sum_i \alpha_i = 1$$

Sherpa Approach



- Recursively apply t-channel and s-channel until all pieces consistent of the basic building blocks
- Number of channels grows at the same rate as number of diagrams (i.e. factorially)

Chili Approach

t-channel Component

$$\mathrm{d}x_a \mathrm{d}x_b \, \mathrm{d}\Phi_n(a,b;1,\ldots,n) = \frac{2\pi}{s} \left[\prod_{i=1}^{n-1} \frac{1}{16\pi^2} \, \mathrm{d}p_{i,\perp}^2 \, \mathrm{d}y_i \, \frac{\mathrm{d}\phi_\mathrm{i}}{2\pi} \right] \mathrm{d}y_n \; .$$

s-channel Component

$$\mathrm{d}\Phi_2(\{1,2\};1,2) = \frac{1}{16\pi^2} \frac{\sqrt{(p_1 p_2)^2 - p_1^2 p_2^2}}{(p_1+p_2)^2} \, \mathrm{d}\cos\theta_1^{\{1,2\}} \, \mathrm{d}\phi_1^{\{1,2\}} \; .$$

Note: Chili provides a method to limit the number of s-channel components. Reducing the scaling of the number of channels from factorial to polynomial.

Overview of Machine Learning

[https://xkcd.com/1838/]

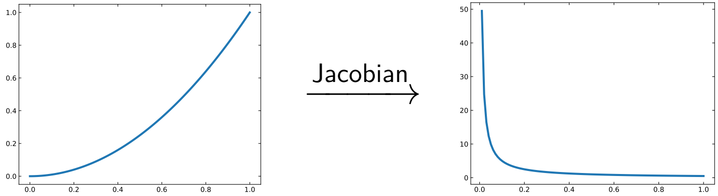
Machine Learning:

- A complex function with inputs and outputs
- Train parameters by minimizing a "loss function"
- Tools like TensorFlow, PyTorch, Keras, etc. make more like black box

Jacobian Cost

Use GAN or Dense network as tranformation:

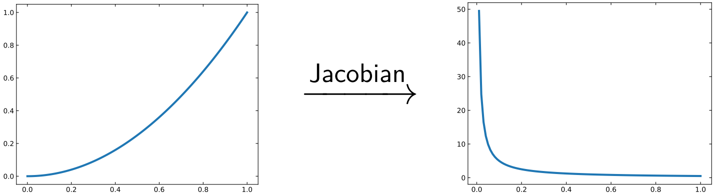
- Network contains n_{dim} nodes in input and output layers mapping from x to $q(x)$
- Requires Jacobian from transformation of variables: $q(y) = q(y(x)) = |\frac{\partial y}{\partial x}|^{-1}$



Jacobian Cost

Use GAN or Dense network as tranformation:

- Network contains n_{dim} nodes in input and output layers mapping from x to $q(x)$
- Requires Jacobian from transformation of variables: $q(y) = q(y(x)) = |\frac{\partial y}{\partial x}|^{-1}$



- Jacobian takes $\mathcal{O}\left(n^3\right)$ time to calculate, prohibitive at large n

Normalizing Flows

Goal: Develop a network architecture with analytic Jacobian.

Normalizing Flows

Goal: Develop a network architecture with analytic Jacobian.

Requirements:

- Bijective

Normalizing Flows

Goal: Develop a network architecture with analytic Jacobian.

Requirements:

- Bijective
- Continuous

Normalizing Flows

Goal: Develop a network architecture with analytic Jacobian.

Requirements:

- Bijective
- Continuous
- Flexible

Normalizing Flows

Goal: Develop a network architecture with analytic Jacobian.

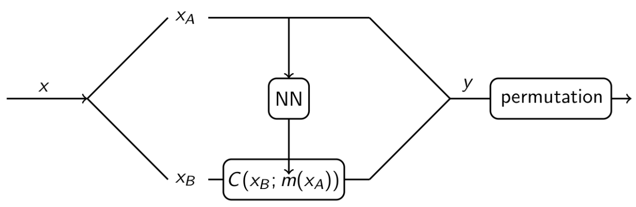
Requirements:

- Bijective
- Continuous
- Flexible

Answer: Normalizing Flows!

- First introduced in "Nonlinear Independent Component Estimation" (NICE) [1410.8516]
- More complex transformations using splines in [1808.03856] and [1906.04032]
- Easy to implement using TensorFlow-Probability [https://www.tensorflow.org/probability]

Normalizing Flows: Basic Building Block



Forward Transform:

$$y_A = x_A$$
$$y_{B,i} = C(x_{B,i}; m(x_A))$$

Inverse Transform:

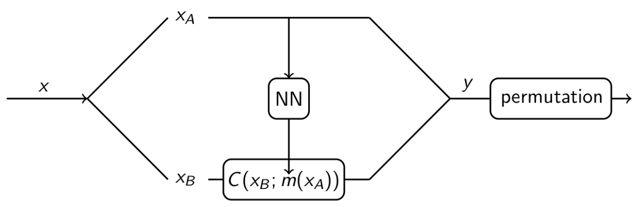
$$x_A = y_A$$
$$x_{B,i} = C^{-1}(y_{B,i}; m(y_A))$$

The C function: numerically cheap, easily invertible, and separable.

Jacobian:

$$\left|\frac{\partial y}{\partial x}\right| = \begin{vmatrix} 1 & \frac{\partial C}{\partial x_A} \\ 0 & \frac{\partial C}{\partial x_B} \end{vmatrix} = \frac{\partial C(x_B; m(x_A))}{\partial x_B}$$

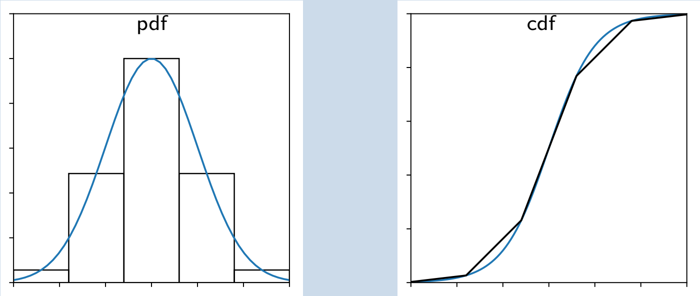
Normalizing Flows: Basic Building Block



Jacobian is $\mathcal{O}(\mathbf{n})$

Normalizing Flows: Piecewise CDF

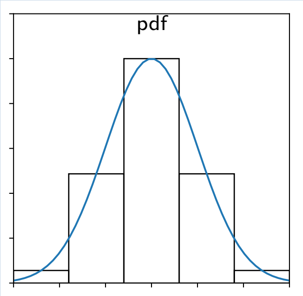
Piecewise Linear CDF: [Müller et al. 1808.03856]

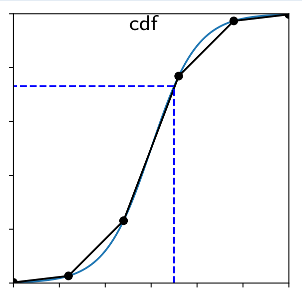


The NN predicts the pdf bin heights Q_i.

Normalizing Flows: Piecewise CDF

Piecewise Linear CDF: [Müller et al. 1808.03856]





$$C = \sum_{k=1}^{b-1} Q_k + \alpha Q_b$$

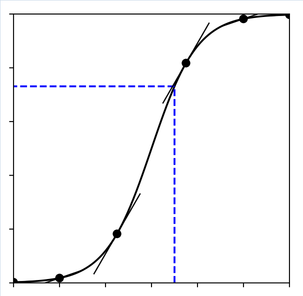
$$\alpha = \frac{x - (b-1)w}{w}$$

$$\left|\frac{\partial C}{\partial x_B}\right| = \prod_i \frac{Q_{b_i}}{w}$$

The NN predicts the pdf bin heights Q_i.

Normalizing Flows: Piecewise CDF

Rational Quadratic CDF: [Durkan et. al. 1906.04032]



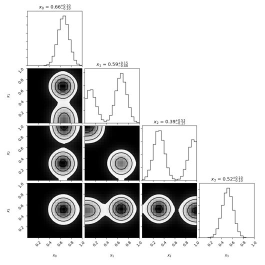
$$C = y^{(k)} + \frac{(y^{(k+1)} - y^{(k)})[s^{(k)}\alpha^2 + d^{(k)}\alpha(1-\alpha)]}{s^{(k)} + [d^{(k+1)} + d^{(k)} - 2s^{(k)}]\alpha(1-\alpha)}$$

$$\alpha = \frac{x - x^{(k)}}{w^{(k)}} \qquad s^{(k)} = \frac{y^{(k+1)} - y^{(k)}}{w^{(k)}}$$

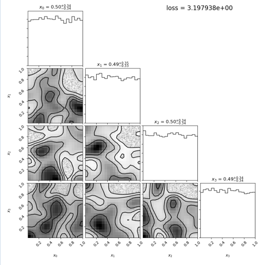
Predict widths ($w^{(k)}$), heights ($y^{(k)}$), and derivatives ($d^{(k)}$) of the knots of spline.

Test Functions: 4-d Camel

Target Distribution:



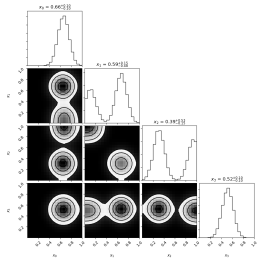
Before Training:



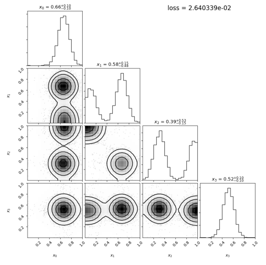
- $f_2(\vec{x}) = \frac{1}{2}(\alpha\sqrt{\pi})^{-n}\left(\exp\left\{-\frac{\sum_i (x_i - \frac{1}{3})^2}{\alpha^2}\right\} + \exp\left\{-\frac{\sum_i (x_i - \frac{2}{3})^2}{\alpha^2}\right\}\right)$

Test Functions: 4-d Camel

Target Distribution:



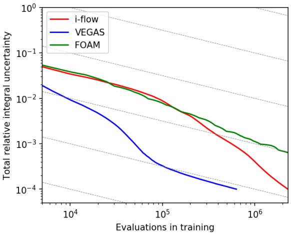
Final:

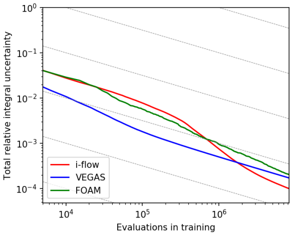


- $f_2(\vec{x}) = \frac{1}{2}(\alpha\sqrt{\pi})^{-n}\left(\exp\left\{-\frac{\sum_i (x_i - \frac{1}{3})^2}{\alpha^2}\right\} + \exp\left\{-\frac{\sum_i (x_i - \frac{2}{3})^2}{\alpha^2}\right\}\right)$

- Expected Result: 0.963657
- `i-flow`: 0.96365(10)
- `VEGAS`: 0.96345(10)

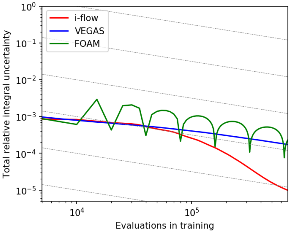
Other Integrands

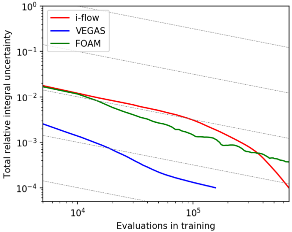




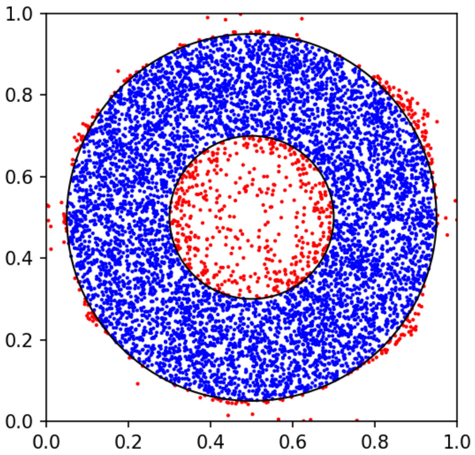
4-dimensional Gaussian
54-dimensional Polynomial

4-dimensional Camel
Scalar top loop





Other Integrands: Handling hard cuts



- `i-flow` trained with 5 million points
- Initially uniform on $[0,1]$
- 7500 points shown, with 6720 points inside and 780 outside (89.6% efficiency)

Other Integrands

Number of function evaluations to reach uncertainty of 10^{-4} (10^{-5} for polynomial)

	Dim	VEGAS	Foam	i-flow
Gaussian	2	**164,436**	6,259,812	2,310,000
	4	**631,874**	24,094,679	2,285,000
	8	**1,299,718**	> 50,000,000 †	3,095,000
	16	**2,772,216**	> 50,000,000 †	7,230,000
Camel	2	**421,475**	5,619,646	2,225,000
	4	24,139,889	21,821,075	**8,220,000**
	8	> 50,000,000 †	> 50,000,000	**19,460,000**
	16	993,294 †	> 50,000,000 †	32,145,000 †
Entangled circles	2	43,367,192	**17,499,823**	23,105,000
Annulus w. cuts	2	4,981,080 †	**11,219,498**	17,435,000
Scalar-top-loop	3	**152,957**	5,290,142	685,000
Polynomial	18	42,756,678	> 50,000,000	**585,000**
	54	> 50,000,000	> 21,505,000 *	**685,000**
	96	> 50,000,000 †	> 10,235,000 *	**1,145,000**

[C. Gao, JI, C. Krause, [arxiv:2001.05486, MLST]]

Comparison between Sherpa, Chili, and Chili + NF

Integration Methods:

- Sherpa uses recursive phase space, number of channels grows factorially
- Chili uses the new mapping with as many s-channels as possible included. This also grows factorially.
- Chili (basic) uses the minimum number of s-channels

Metrics:

- Compare the statistical uncertainty for a fixed number of events for optimizing and estimating the uncertainty
- Compare the unweighting efficiency. The unweighting efficiency is given by the average weight of all events over the bootstrapped median maximum weight using 100 samples to estimate the median. For details see [2001.10028] .

Sherpa vs Chili

Process	Sherpa		Chili		Chili (basic)	
	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts
W^++1j	0.5‰	7×10^{-2}	0.6‰	9×10^{-2}	0.6‰	9×10^{-2}
W^++2j	1.2‰	9×10^{-3}	1.1‰	2×10^{-2}	1.2‰	1×10^{-2}
W^++3j	2.0‰	1×10^{-3}	2.0‰	4×10^{-3}	2.9‰	2×10^{-3}
W^++4j	3.7‰	2×10^{-4}	4.9‰	7×10^{-4}	6.0‰	3×10^{-4}
W^++5j	7.2‰	4×10^{-5}	22‰	1×10^{-5}	26‰	1×10^{-5}

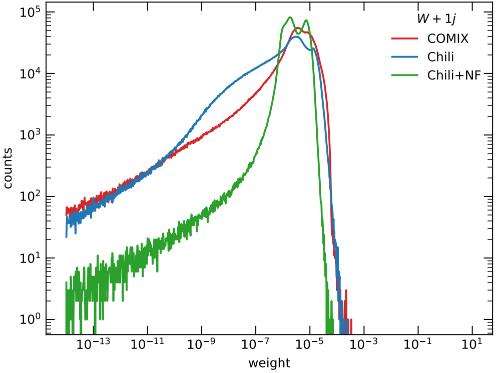
Process	Sherpa		Chili		Chili (basic)	
	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts
Z+1j	0.4‰	2×10^{-1}	0.5‰	1×10^{-1}	0.5‰	1×10^{-1}
Z+2j	0.8‰	2×10^{-2}	0.8‰	3×10^{-2}	1.0‰	2×10^{-2}
Z+3j	1.3‰	4×10^{-3}	1.6‰	7×10^{-3}	2.5‰	4×10^{-3}
Z+4j	2.2‰	8×10^{-4}	3.6‰	1×10^{-3}	5.0‰	6×10^{-4}
Z+5j	3.7‰	1×10^{-4}	11‰	1×10^{-4}	13‰	2×10^{-4}

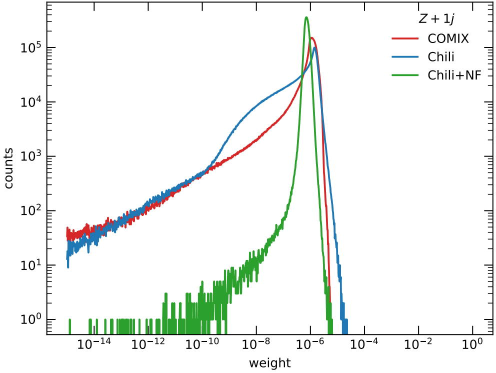
Chili vs Chili + NF

Process (color sum)	CHILI		CHILI (Basic)+NF	
	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts
W^++1j	0.4‰	2×10^{-1}	0.2‰	4×10^{-1}
W^++2j	0.7‰	4×10^{-2}	0.7‰	5×10^{-2}

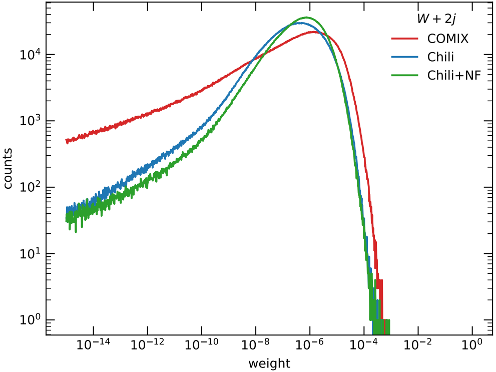
Process (color sum)	CHILI		CHILI (Basic)+NF	
	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts
Z+1j	0.4‰	2×10^{-1}	0.1‰	5×10^{-1}
Z+2j	0.6‰	5×10^{-2}	0.6‰	6×10^{-2}

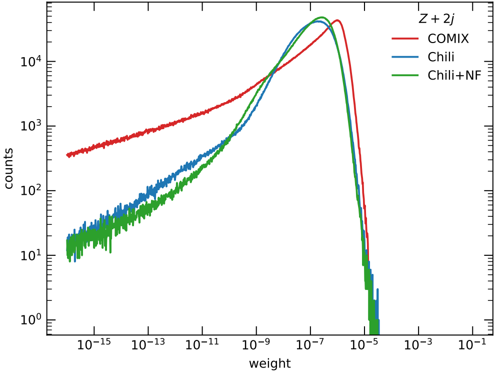
Weight distributions (1 jet)





Weight distributions (2 jet)





Conclusions

Traditional Integration

- Numerical integration and the need for Monte Carlo
- VEGAS algorithm and its deficiencies

Conclusions

Traditional Integration

- Numerical integration and the need for Monte Carlo
- VEGAS algorithm and its deficiencies

Chili

- Develop new mapping that controls number of s-channels
- Reduces scaling of number of channels
- No significant efficiency or accuracy penalty

Conclusions

Traditional Integration

- Numerical integration and the need for Monte Carlo
- VEGAS algorithm and its deficiencies

Chili

- Develop new mapping that controls number of s-channels
- Reduces scaling of number of channels
- No significant efficiency or accuracy penalty

Normalizing Flows

- Avoid computational difficulty of Jacobian
- Using splines to approximate CDF

Conclusions

Traditional Integration

- Numerical integration and the need for Monte Carlo
- VEGAS algorithm and its deficiencies

Chili

- Develop new mapping that controls number of s-channels
- Reduces scaling of number of channels
- No significant efficiency or accuracy penalty

Normalizing Flows

- Avoid computational difficulty of Jacobian
- Using splines to approximate CDF

Outlook

- Investigate more complex processes with normalizing flows
- Combine normalizing flows with multichanneling
- Implement GPU matrix element and phase space

Other Integrands: Test Functions

Gaussian

$$f_1(\vec{x}) = (\alpha\sqrt{\pi})^{-n} \exp\{-\frac{\sum_i (x_i - 0.5)^2}{\alpha^2}\}$$

Entangled Cricles

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

Annulus w. cuts

$$f_4(x_1, x_2) = \left\{ \begin{array}{cc} 1 & 0.2 < \sqrt{x_1^2 + x_2^2} < 0.45 \\ 0 & \text{else} \end{array} \right\}$$

Scalar Box

$f_5(t_1, t_2, t_3; s_{12}, s_{23}, s_1, s_2, s_3, s_4, m_1^2, m_2^2, m_3^2, m_4^2)$ with $s_{12} = -s_{23} = 130^2$ GeV^2, $s_1 = s_2 = s_3 = 0$ GeV^2, $s_4 = 125^2$ GeV^2, $m_1 = m_2 = m_3 = m_4 = 175$ GeV.

Polynomial

$$f_6(x_1, \ldots, x_n) = \sum_{i=1}^n -x_i^2 + x_i$$

Sherpa vs Chili

Process	Sherpa		Chili		Chili (basic)	
	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts
h+1j	0.4‰	2×10^{-1}	0.4‰	2×10^{-1}	0.4‰	2×10^{-1}
h+2j	0.8‰	2×10^{-2}	0.6‰	5×10^{-2}	0.6‰	5×10^{-2}
h+3j	1.4‰	3×10^{-3}	0.9‰	2×10^{-2}	0.9‰	2×10^{-2}
h+4j	2.4‰	6×10^{-4}	1.6‰	6×10^{-3}	1.7‰	7×10^{-3}
h+5j	4.5‰	1×10^{-4}	3.2‰	1×10^{-3}	3.6‰	1×10^{-3}

Process	Sherpa		Chili		Chili (basic)	
	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts
$t\bar{t}$+0j	0.6‰	1×10^{-1}	0.6‰	1×10^{-1}	0.6‰	1×10^{-1}
$t\bar{t}$+1j	0.9‰	2×10^{-2}	0.6‰	6×10^{-2}	0.9‰	3×10^{-2}
$t\bar{t}$+2j	1.4‰	4×10^{-3}	0.9‰	2×10^{-2}	1.4‰	1×10^{-2}
$t\bar{t}$+3j	2.6‰	7×10^{-4}	1.5‰	7×10^{-3}	2.9‰	2×10^{-3}
$t\bar{t}$+4j	4.0‰	1×10^{-4}	3.2‰	1×10^{-3}	3.5‰	8×10^{-4}

Sherpa vs Chili

Process	Sherpa		Chili		Chili (basic)	
	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts
γ+1j	0.4‰	2×10^{-1}	0.6‰	1×10^{-1}	0.6‰	1×10^{-1}
γ+2j	1.1‰	7×10^{-3}	2.2‰	3×10^{-3}	3.7‰	1×10^{-3}
γ+3j	2.4‰	5×10^{-4}	4.9‰	4×10^{-4}	10‰	1×10^{-4}
γ+4j	5.0‰	7×10^{-5}	20‰	3×10^{-5}	30‰	4×10^{-5}
γ+5j	9.3‰	2×10^{-5}	28‰	7×10^{-6}	36‰	2×10^{-6}

Process	Sherpa		Chili		Chili (basic)	
	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts
2jets	0.6‰	5×10^{-2}	0.4‰	1×10^{-1}	0.5‰	7×10^{-2}
3jets	1.2‰	5×10^{-3}	1.0‰	1×10^{-2}	1.8‰	7×10^{-3}
4jets	2.5‰	5×10^{-4}	2.0‰	3×10^{-3}	3.4‰	1×10^{-3}
5jets	4.7‰	9×10^{-5}	5.1‰	6×10^{-4}	8.1‰	2×10^{-4}
6jets	7.0‰	2×10^{-5}	15‰	5×10^{-5}	14‰	4×10^{-5}

Chili vs Chili + NF

Process (color sum)	Chili		Chili (Basic)+NF	
	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts
h+1j	0.2‰	5×10^{-1}	0.05‰	8×10^{-1}
h+2j	0.3‰	1×10^{-1}	0.3‰	2×10^{-1}

Process (color sum)	Chili		Chili (Basic)+NF	
	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts
$t\bar{t}$+0j	0.1‰	6×10^{-1}	0.05‰	7×10^{-1}
$t\bar{t}$+1j	0.2‰	3×10^{-1}	0.3‰	2×10^{-1}

Chili vs Chili + NF

Process (color sum)	CHILI		CHILI (Basic)+NF	
	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts
γ+1j	0.6‰	2×10^{-1}	0.1‰	5×10^{-1}
γ+2j	1.8‰	5×10^{-3}	1.4‰	9×10^{-3}

Process (color sum)	CHILI		CHILI (Basic)+NF	
	$\Delta\sigma/\sigma$ 6M pts	η 100 evts	$\Delta\sigma/\sigma$ 6M pts	η 100 evts
2jets	0.2‰	4×10^{-1}	0.08‰	6×10^{-1}
3jets	0.5‰	6×10^{-2}	0.7‰	3×10^{-2}