Phase space Monte Carlo for DIS

Andreas van Hameren
Institute of Nuclear Physics
Polish Academy of Sciences
Kraków

presented at the

EICPL seminar, 21-06-2021

Supported by grant No. 2019/35/B/ST2/03531 of the Polish National Science Centre.

Outline

- Numerical integration
- Monte Carlo integration
- Random variable generation
- Phase space generation
- Phase space generation for DIS
- KaTie

Numerical integration

Let $\mathbf{M}$ be a space on which a Lebesgue measure is defined.
Let f be a Lebesgue integrable function.

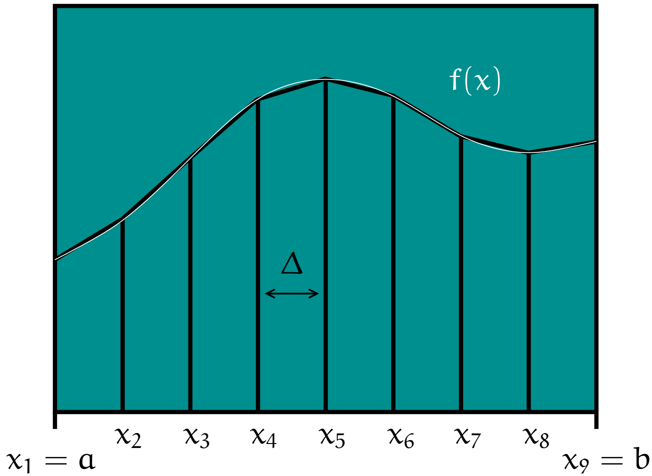
Numerical integration is the attempt to obtain the integral of f by the evaluation of $f(x)$ for several $x \in \mathbf{M}$. More specifically

$$\int_{\mathbf{M}} d^{\omega}x\, f(x) \stackrel{?}{=} \sum_{i=1}^{N} w(x_i) f(x_i)$$

- various methods exist for choosing the *integration points* x_i and the *weights* $w(x_i)$
- several aspects have to be considered to determine if a method is *good*
  - do you need precision or/and speed?
  - does the method give an error estimate?
  - does the method allow for increasing the precision (by increasing N)?
  - how does the cost of generating the x_i and evaluating the $w(x_i)$ compare to the cost of evaluating $f(x_i)$?
- example for $\mathbf{M} = [a, b]$: define $\Delta = (b - a)/(N - 1)$ and

$$x_i = a + \Delta(i - 1) \quad , \quad w(x_1) = w(x_N) = \frac{\Delta}{2} \quad , \quad w(x_i) = \Delta \quad \text{for} \quad 1 < i < N$$

Numerical integration for a 1-dim example

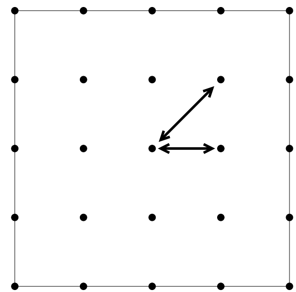


- example for $\mathbf{M} = [a, b]$: define $\Delta = (b - a)/(N - 1)$ and

$$x_i = a + \Delta(i - 1) \quad , \quad w(x_1) = w(x_N) = \frac{\Delta}{2} \quad , \quad w(x_i) = \Delta \quad \text{for} \quad 1 < i < N$$

Curse of dimenionality

- If there is no *a priori* knowledge about f, it seems most reasonable to take the points $\{x_i\}$ distributed over $\mathbf{M}$ as uniformly as possible.
- In the 1-dim case, this means the regular grid of the example before.
- Finding uniformly distributed point sets in more dimensions is a highly non-trivial problem.



For a hyperqubic grid in n dimensions

$$\sum_{i_1=1}^{N}\sum_{i_2=1}^{N}\cdots\sum_{i_n=1}^{N} w_{i_1,i_2,\ldots,i_n}\, f\big(a_1+\Delta_1(i_i-1), a_2+\Delta_2(i_2-1),\ldots,a_n+\Delta_n(i_n-1)\big)$$

you need $\mathcal{O}(N^n)$ points to reach a similar precision as the 1-dim case with $\mathcal{O}(N)$ points.

So if the integration error decreases as $N_{\text{eval}}^{-\alpha}$ in the 1-dim case, it decreases as $N_{\text{eval}}^{-\alpha/n}$ in the n-dim case.

So no matter how good your 1-dim method is, no matter how large α is, in high dimension you always lose.

Monte Carlo integration

Let g be a *probability density* on $\mathbf{M}$ such that if $f(x) \neq 0$ then $g(x) \neq 0$.
Let $\{x_i\}$ be a sequence of points in $\mathbf{M}$ independently drawn at random from g.
Then, for $N \to \infty$, the probability distribution of the random variable

$$X_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{g(x_i)}$$

becomes Gausian, with expectation value and variance

$$\mathsf{E}(X_N) = \int_{\mathbf{M}} d^\omega x\, f(x) \quad , \quad \mathsf{V}(X_N) = \frac{1}{N} \left[\int_{\mathbf{M}} d^\omega x\, \frac{f(x)^2}{g(x)} - \left(\int_{\mathbf{M}} d^\omega x\, f(x) \right)^2 \right]$$

Monte Carlo integration

Let g be a *probability density* on $\mathbf{M}$ such that if $f(x) \neq 0$ then $g(x) \neq 0$.
Let $\{x_i\}$ be a sequence of points in $\mathbf{M}$ independently drawn at random from g.
Then, for $N \to \infty$, the probability distribution of the random variable

$$X_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{g(x_i)}$$

becomes Gausian, with expectation value and variance

$$E(X_N) = \int_{\mathbf{M}} d^\omega x\, f(x) \quad , \quad V(X_N) = \frac{1}{N} \left[\int_{\mathbf{M}} d^\omega x\, \frac{f(x)^2}{g(x)} - \left(\int_{\mathbf{M}} d^\omega x\, f(x) \right)^2 \right]$$

$$\begin{aligned} E(X_N) &= \int_{\mathbf{M}} d^\omega x_1\, g(x_1) \int_{\mathbf{M}} d^\omega x_2\, g(x_2) \cdots \int_{\mathbf{M}} d^\omega x_N\, g(x_N) \left(\frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{g(x_i)} \right) \\ &= \frac{1}{N} \sum_{i=1}^{N} \int_{\mathbf{M}} d^\omega x_i\, f(x_i) \prod_{j \neq i}^{N} \int_{\mathbf{M}} d^\omega x_j\, g(x_j) = \frac{1}{N} N \int_{\mathbf{M}} d^\omega x\, f(x) \times 1 \end{aligned}$$

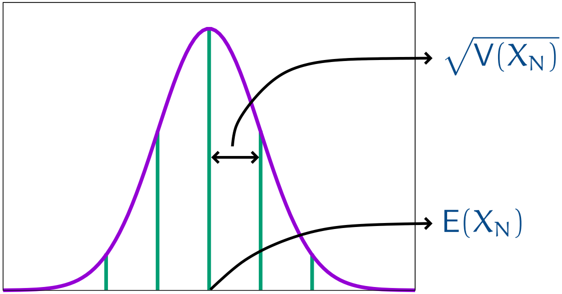
Monte Carlo integration

Let g be a *probability density* on $\mathbf{M}$ such that if $f(x) \neq 0$ then $g(x) \neq 0$.
Let $\{x_i\}$ be a sequence of points in $\mathbf{M}$ independently drawn at random from g.
Then, for $N \to \infty$, the probability distribution of the random variable

$$X_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{g(x_i)}$$

becomes Gausian, with expectation value and variance

$$E(X_N) = \int_{\mathbf{M}} d^\omega x \, f(x) \quad , \quad V(X_N) = \frac{1}{N} \left[\int_{\mathbf{M}} d^\omega x \, \frac{f(x)^2}{g(x)} - \left(\int_{\mathbf{M}} d^\omega x \, f(x) \right)^2 \right]$$



X_N is an estimate of the integral of f with error estimate $\sqrt{V(X_N)}$

Monte Carlo integration

Let g be a *probability density* on $\mathbf{M}$ such that if $f(x) \neq 0$ then $g(x) \neq 0$.
Let $\{x_i\}$ be a sequence of points in $\mathbf{M}$ independently drawn at random from g.
Then, for $N \to \infty$, the probability distribution of the random variable

$$X_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{g(x_i)}$$

becomes Gausian, with expectation value and variance

$$E(X_N) = \int_{\mathbf{M}} d^{\omega}x\, f(x) \quad , \quad V(X_N) = \frac{1}{N}\left[\int_{\mathbf{M}} d^{\omega}x\, \frac{f(x)^2}{g(x)} - \left(\int_{\mathbf{M}} d^{\omega}x\, f(x)\right)^2\right]$$

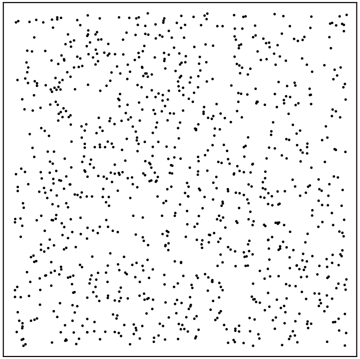
- X_N is an estimate of the integral of f with error estimate $\sqrt{V(X_N)}$
- $V(X_N)$ can be estimated itself with $\left[N^{-1} \sum_{i=1}^{N} f(x_i)^2/g(x_i)^2 - X_N^2\right]/(N-1)$
- the error decreases as $N^{-1/2}$, independently of $\mathbf{M}$
- *importance sampling*: convergence can be improved by choosing g such that it has the same shape as f. If you can construct $g(x) = f(x)/\int_{\mathbf{M}} d^{\omega}y\, f(y)$, then you actually solved the integration problem without the need of Monte Carlo.

Random number generation

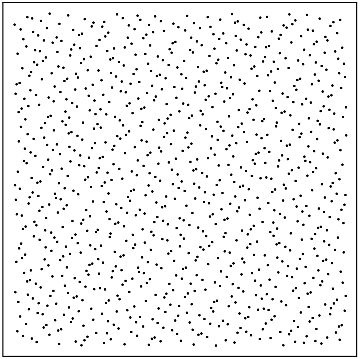
We will always assume there is a generator of uniformly distributed random numbers $\rho \in [0, 1]$ available.

In practice, this is a *pseudo random number* generator, based on a deterministic algorithm (which is actually usefull for reproducibility).

In higher dimension, so called *quasi random numbers* can be applied to achieve better uniformity:



(pseudo) random



quasi random

Error decreases as $(\log N)^p/N$, but accurate error estimation is much more difficult for Quasi Monte Carlo.

Random variable generation

Several methods exist to generate random variables with non-trivial distribution g on non-trivial space $\mathbf{M}$:

Inversion: find mapping $\varphi : [0,1]^\omega \to \mathbf{M}$ such that

$$|J_{\varphi^{-1}}(x)| = \int_0^1 d^\omega\rho\, \delta^\omega(x - \varphi(\rho)) = g(x)$$

- Is practically only possible for 1-dim cases, for which it can be formulated as solving

$$\int_{-\infty}^{x} dy\, g(y) = \rho \int_{-\infty}^{\infty} dy\, g(y) \quad \Rightarrow \quad \varphi : \rho \to x$$

Random variable generation

Several methods exist to generate random variables with non-trivial distribution g on non-trivial space $\mathbf{M}$:

Inversion: find mapping $\varphi : [0,1]^\omega \to \mathbf{M}$ such that

$$|J_{\varphi^{-1}}(x)| = \int_0^1 d^\omega\rho\, \delta^\omega(x - \varphi(\rho)) = g(x)$$

- Is practically only possible for 1-dim cases, for which it can be formulated as solving

$$\int_{-\infty}^{x} dy\, g(y) = \rho \int_{-\infty}^{\infty} dy\, g(y) \quad \Rightarrow \quad \varphi : \rho \to x$$

$$g(x) \stackrel{?}{\sim} \frac{\theta(1 < x < 2))}{x} \quad \Rightarrow \quad \int_1^x \frac{dy}{y} = \rho \int_1^2 \frac{dy}{y} \;\Leftrightarrow\; \log(x) = \rho \log(2) \;\Leftrightarrow\; x = 2^\rho$$

Random variable generation

Several methods exist to generate random variables with non-trivial distribution g on non-trivial space $\mathbf{M}$:

Inversion: find mapping $\varphi : [0,1]^\omega \to \mathbf{M}$ such that

$$|J_{\varphi^{-1}}(x)| = \int_0^1 d^\omega\rho\, \delta^\omega(x - \varphi(\rho)) = g(x)$$

- Is practically only possible for 1-dim cases, for which it can be formulated as solving

$$\int_{-\infty}^{x} dy\, g(y) = \rho \int_{-\infty}^{\infty} dy\, g(y) \quad \Rightarrow \quad \varphi : \rho \to x$$

Rejection: given an approximate $\tilde{g}$ and a number c such that $c\tilde{g}(x) > g(x)\; \forall\, x \in \mathbf{M}$

1. generate x from $\tilde{g}$ and $\rho \in [0,1]$
2. if $\rho\, c\, \tilde{g}(x) \leq g(x)$, then accept x, else reject x and try again

Random variable generation

Several methods exist to generate random variables with non-trivial distribution g on non-trivial space $\mathbf{M}$:

Inversion: find mapping $\varphi : [0,1]^\omega \to \mathbf{M}$ such that

$$|J_{\varphi^{-1}}(x)| = \int_0^1 d^\omega\rho\,\delta^\omega(x-\varphi(\rho)) = g(x)$$

- Is practically only possible for 1-dim cases, for which it can be formulated as solving

$$\int_{-\infty}^{x} dy\,g(y) = \rho \int_{-\infty}^{\infty} dy\,g(y) \quad \Rightarrow \quad \varphi : \rho \to x$$

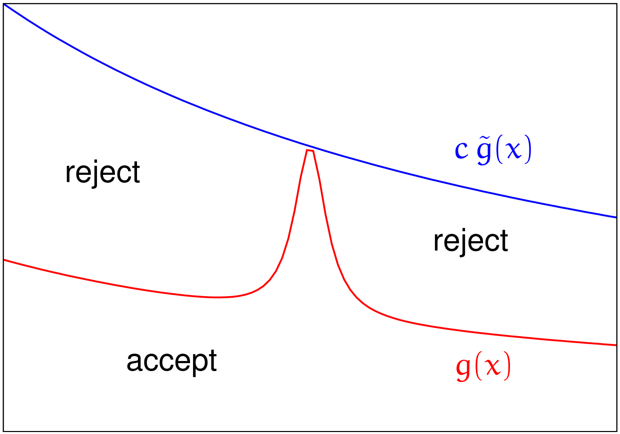
Rejection: given an approximate $\tilde{g}$ and a number c such that $c\tilde{g}(x) > g(x)\ \forall\, x \in \mathbf{M}$

1. generate x from $\tilde{g}$ and $\rho \in [0,1]$
2. if $\rho\, c\, \tilde{g}(x) \leq g(x)$, then accept x, else reject x and try again

$$f(x) = \int_{\mathbf{M}} d^\omega y\,\tilde{g}(y) \int_0^1 d\rho\,[\theta(\rho\, c\, \tilde{g}(y) \leq g(y))\,\delta(x-y) + \theta(\rho\, c\, \tilde{g}(y) > g(y))\,f(x))]$$

$$\Rightarrow f(x) = g(x)\left(\int_{\mathbf{M}} d^\omega y\,g(y)\right)^{-1}$$

Random variable generation



Rejection: given an approximate $\tilde{g}$ and a number c such that $c\tilde{g}(x) > g(x)\ \forall\, x \in \mathbf{M}$

1. generate x from $\tilde{g}$ and $\rho \in [0, 1]$
2. if $\rho\, c\, \tilde{g}(x) \leq g(x)$, then accept x, else reject x and try again

- The larger c, the less efficient the rejection, i.e. the more trails are needed.
- Interpreting $g(x)/\tilde{g}(x)$ as the *weight* of x, rejection produces a sequence of x-es with constant weight, i.e. they are *unweigthed*.

Random variable generation

Several methods exist to generate random variables with non-trivial distribution g on non-trivial space $\mathbf{M}$:

Inversion: find mapping $\varphi : [0,1]^\omega \to \mathbf{M}$ such that

$$|J_{\varphi^{-1}}(x)| = \int_0^1 d^\omega\rho\,\delta^\omega(x - \varphi(\rho)) = g(x)$$

- Is practically only possible for 1-dim cases, for which it can be formulated as solving

$$\int_{-\infty}^{x} dy\, g(y) = \rho \int_{-\infty}^{\infty} dy\, g(y) \quad \Rightarrow \quad \varphi : \rho \to x$$

Rejection: given an approximate $\tilde{g}$ and a number c such that $c\tilde{g}(x) > g(x)\ \forall\, x \in \mathbf{M}$

1. generate x from $\tilde{g}$ and $\rho \in [0,1]$
2. if $\rho\, c\, \tilde{g}(x) \leq g(x)$, then accept x, else reject x and try again

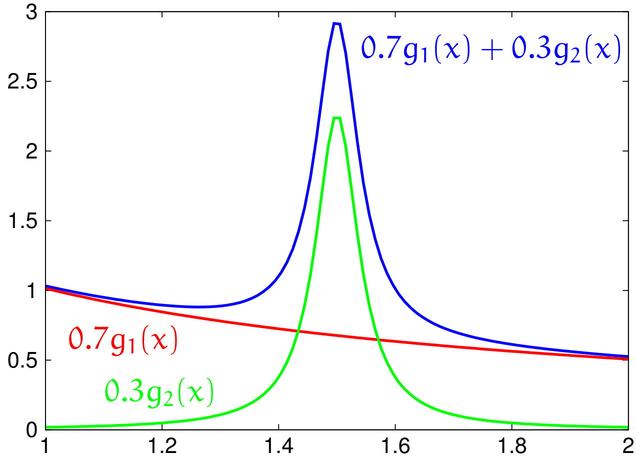
- The larger c, the less efficient the rejection, i.e. the more trails are needed.
- Interpreting $g(x)/\tilde{g}(x)$ as the *weight* of x, rejection produces a sequence of x-es with constant weight, i.e. they are *unweigthed*.

Many very efficient 1-dim algorithms are based on combining inversion and rejection.

Adaptive random variable generation

Multi-channel (mixture distribution): given n densities g_i and possitive weights w_i with $\sum_{i=1}^{n} w_i = 1$, we can define the density $g(x) = \sum_{i=1}^{n} w_i\, g_i(x)$. To generate x following g

1. generate $\rho \in [0, 1]$ and find i such that $\sum_{j=1}^{i-1} w_j < \rho \leq \sum_{j=1}^{i} w_j$
2. generate $x \in \mathbf{M}$ following g_i



Adaptive random variable generation

Multi-channel(mixture distribution): given n densities g_i and possitive weights w_i with $\sum_{i=1}^{n} w_i = 1$, we can define the density $g(x) = \sum_{i=1}^{n} w_i \, g_i(x)$. To generate x following g

1. generate $\rho \in [0, 1]$ and find i such that $\sum_{j=1}^{i-1} w_j < \rho \leq \sum_{j=1}^{i} w_j$
2. generate $x \in \mathbf{M}$ following g_i

Adaptive multi-channel for overlapping densities g_i (Kleiss, Pittau 1994):

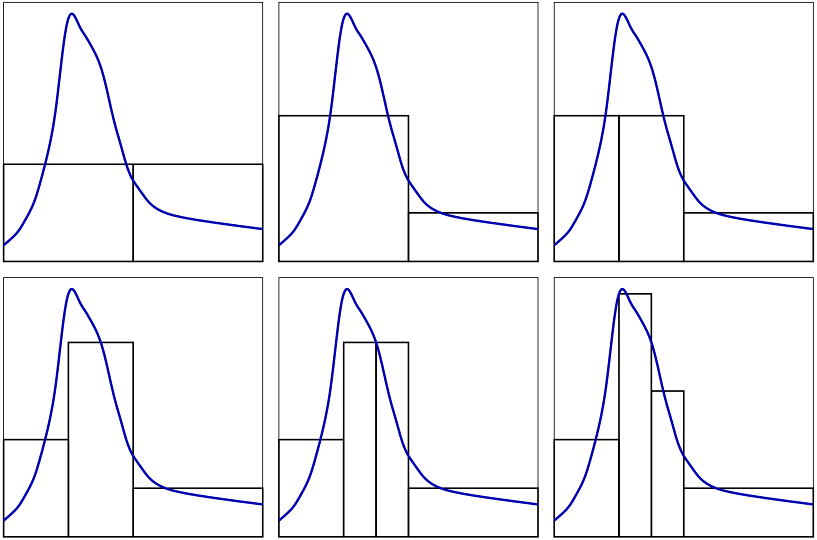
3. collect batches of N integration points x_j and update the weights following

$$w_i \leftarrow w_i \sqrt[p]{\frac{1}{N} \sum_{j=1}^{N} \frac{g_i(x_j)}{g(x_j)} \left(\frac{f(x_j)}{g(x_j)} \right)^p} \quad \text{and normalize} \quad \sum_{i=1}^{n} w_i = 1$$

- Originally formulated for $p = 2$ to achieve variance minimalization.
- Works for up to many channels ($\mathcal{O}(1000)$), but the evaluation of $g(x)$ becomes expensive.
- The channels are typically imagined to correspond to Feynman graphs, defining kinematical channels.

Adaptive random variable generation

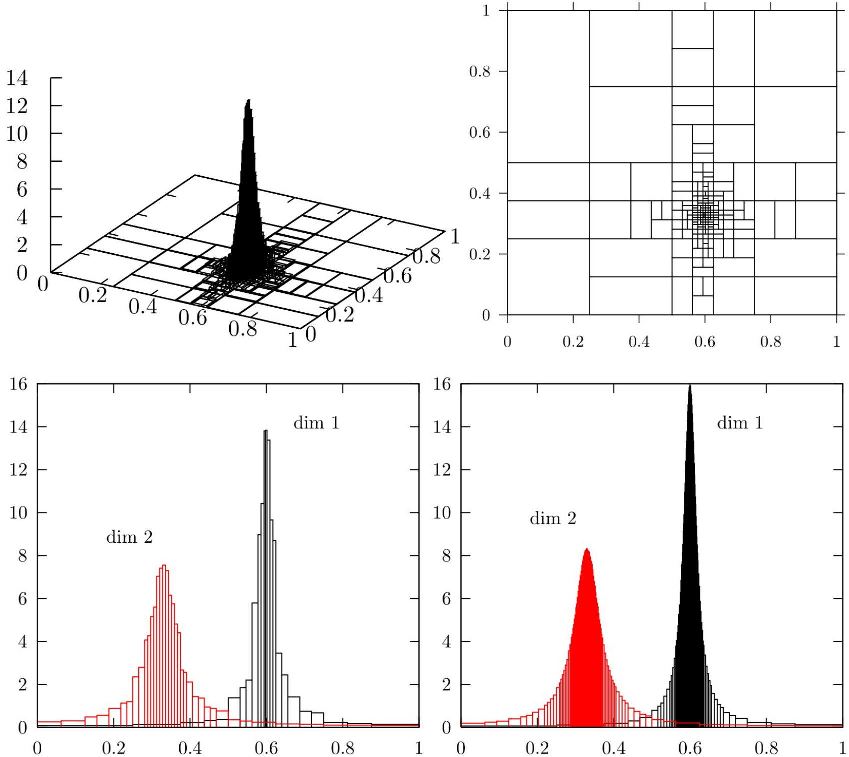
Besides the weights w_i in $g(x) = \sum_{i=1}^{n} w_i\, g_i(x)$, we can also try to adapt the densities g_i themselves. In particular when the densities g_i are uniform non-overlapping partition of the integration space, various strategies are thinkable.



Adaptive random variable generation

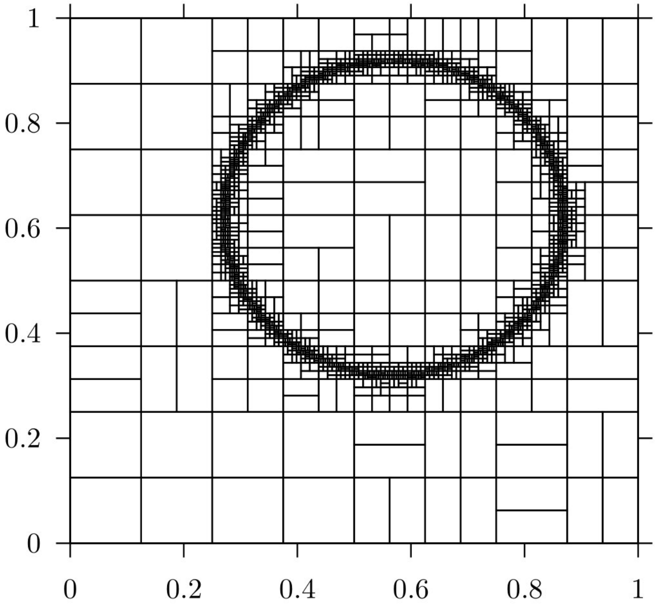
VEGAS approach

In case of multi-dimensional integration problems, it works best for factorizable integrands. Then, one-dimensional partitions are best used.



Adaptive random variable generation

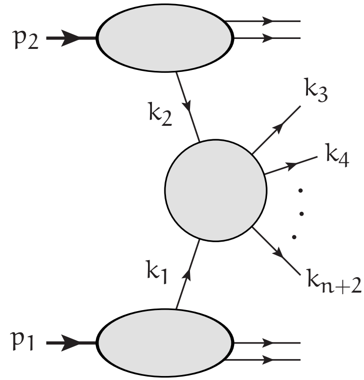
For non-factorizable integrands, truly multi-dimensional partitions can be applied (eg. FOAM Jadach 2000 and PARNI AvH 2009), but the curse of dimensionality returns.



k_T-dependent factorization

Hadron-scattering process Y with partonic processes y contributing to multi-jet final state

$$d\sigma_Y(p_1, p_2; k_3, \ldots, k_{2+n}) = \sum_{y \in Y} \int d^4k_1\, \mathcal{P}_{y_1}(k_1) \int d^4k_2\, \mathcal{P}_{y_2}(k_2)\, d\hat{\sigma}_y(k_1, k_2; k_3, \ldots, k_{2+n})$$



Collinear factorization:

$$\mathcal{P}_{y_i}(k_i) = \int \frac{dx_i}{x_i} f_{y_i}(x_i, \mu)\, \delta^4(k_i - x_i p_i)$$

k_T-dependent factorization factorization:

$$\mathcal{P}_{y_i}(k_i) = \int \frac{d^2\mathbf{k}_{iT}}{\pi} \int \frac{dx_i}{x_i} \mathcal{F}_{y_i}(x_i, |\mathbf{k}_{iT}|, \mu)\, \delta^4(k_i - x_i p_i - k_{iT})$$

Differential partonic cross section:

$$d\hat{\sigma}_y(k_1, k_2; k_3, \ldots, k_{2+n}) = d\Phi_Y(k_1, k_2; k_3, \ldots, k_{2+n})\, \Theta_Y(k_3, \ldots, k_{2+n}) \\ \times \mathrm{flux}(k_1, k_2) \times \mathcal{S}_y\, |\mathcal{M}_y(k_1, \ldots, k_{2+n})|^2$$

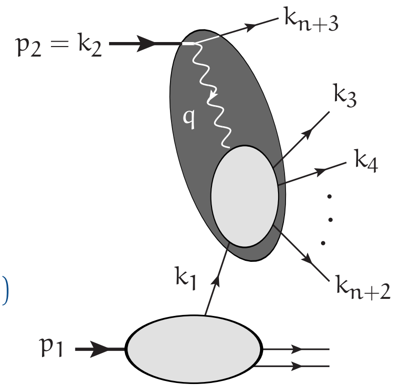
Parton-level phase space:

$$d\Phi_Y(k_1, k_2; k_3, \ldots, k_{3+n}) = \left(\prod_{i=3}^{n+3} d^4k_i \delta_+(k_i^2 - m_i^2) \right) \delta^4(k_1 + k_2 - k_3 - \cdots - k_{n+3})$$

k_T-dependent factorization

eh-scattering process Y with partonic processes y contributing to multi-jet final state

$$d\sigma_Y(p_1, p_2; k_3, \ldots, k_{3+n}) = \sum_{y \in Y} \int d^4k_1 \, \mathcal{P}_{y_1}(k_1) \qquad d\hat{\sigma}_y(k_1, k_2; k_3, \ldots, k_{3+n})$$



Collinear factorization:

$$\mathcal{P}_{y_i}(k_i) = \int \frac{dx_i}{x_i} f_{y_i}(x_i, \mu) \, \delta^4(k_i - x_i p_i)$$

k_T-dependent factorization factorization:

$$\mathcal{P}_{y_i}(k_i) = \int \frac{d^2\mathbf{k}_{iT}}{\pi} \int \frac{dx_i}{x_i} \mathcal{F}_{y_i}(x_i, |\mathbf{k}_{iT}|, \mu) \, \delta^4(k_i - x_i p_i - k_{iT})$$

Differential partonic cross section:

$$\begin{aligned} d\hat{\sigma}_y(k_1, k_2; k_3, \ldots, k_{3+n}) &= d\Phi_Y(k_1, k_2; k_3, \ldots, k_{3+n}) \, \Theta_Y(k_3, \ldots, k_{3+n}) \\ &\times \text{flux}(k_1, k_2) \times \mathcal{S}_y \, |\mathcal{M}_y(k_1, \ldots, k_{3+n})|^2 \end{aligned}$$

Parton-level phase space:

$$d\Phi_Y(k_1, k_2; k_3, \ldots, k_{3+n}) = \left(\prod_{i=3}^{n+3} d^4k_i \delta_+(k_i^2 - m_i^2) \right) \delta^4(k_1 + k_2 - k_3 - \cdots - k_{n+3})$$

Parton-level event generation

- choose partonic subprocess $y = y_1, y_2 \to y_3, \ldots y_{n+2}$ with probability $P(y)$
- generate initial-state variables x_1, x_2, k_{T1}, k_{T2} with probability $P(y; x_1, x_2, k_{T1}, k_{T2})$
- generate final-state momenta $k_3, \ldots, k_{n+2}$ with differential probability

$$dF(y, k_1, k_2; k_3 \ldots, k_{2+n}) = d\Phi_Y(k_1, k_2; k_3, \ldots, k_{2+n})\, P(y, k_1, k_2; k_3, \ldots, k_{2+n})$$

- assign weight $= 0$ to phase space point if it does not satisfy the inclusive cuts...
- ...else evaluate PDFs and matrix element and assign weight

$$\mathcal{W}_y(k_1, \ldots, k_{2+n}) = \frac{\mathcal{F}_{y_1}(x_1, k_{T1})\, \mathcal{F}_{y_2}(x_2, k_{T2})\, |\mathcal{M}_y(k_1, \ldots, k_{2+n})|^2\, \mathcal{S}_y\, \mathrm{flux}(k_1, k_2)}{P(y)\, P(y; x_1, x_2, k_{T1}, k_{T2})\, P(y, k_1, k_2; k_3, \ldots, k_{2+n})}$$

- choose/create probabilities P wisely/adaptively in order to let $\mathcal{W}_y(k_1, \ldots, k_{2+n})$ fluctuate as little as possible from event to event ...
- ...this requires an optimization stage for each subprocess y during which crude estimates of partonic cross sections are made
- there is a lot of engineering/parameters in P, but there is only QFT in $\mathcal{M}_y$

Phase Space

The differential volume of n-particle phase space is given by

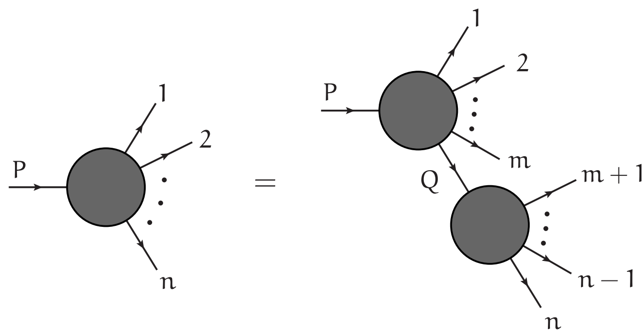
$$d\Phi_n(\,p_1, s_1\,,\,p_2, s_2\,\ldots p_n, s_n\,;\,P\,) = \\ d^4p_1\delta(p_1^2 - s_1)\,d^4p_2\delta(p_2^2 - s_2)\cdots d^4p_n\delta(p_n^2 - s_n)\,\delta^4(P - p_1 - p_2 - \cdots - p_n)$$

and satisfies the recursive relation

$$d\Phi_n(\,p_1, s_1\,,\,p_2, s_2\,\ldots p_n, s_n\,;\,P\,) = \\ dS\,d\Phi_{m+1}(\,p_1, s_1\,,\,p_2, s_2\,\ldots p_m, s_m\,,\,Q, S\,;\,P\,) \\ \times\,d\Phi_{n-m}(\,p_{m+1}, s_{m+1}\,,\,p_{m+2}, s_{m+2}\,\ldots p_n, s_n\,;\,Q\,)$$

with integration over S and Q.

So phase space can be completely decomposed into 2-particle phase spaces, and can be written in terms of invariants and angles.



2-particle phase space

We want to generate p_a, p_b in a 2-particle phase space $\Phi(\,p_a, s_a\,,\,p_b, s_b\,;\,P\,)$. This implies that P and also s_a, s_b are given (generated or squared external masses) and we can define

$$|\vec{q}| = \sqrt{\frac{\lambda(P^2, s_a, s_b)}{4P^2}} \quad \text{with} \quad \lambda(x,y,z) = x^2 + y^2 + z^2 - 2xy - 2yz - 2zx$$

Now, we can

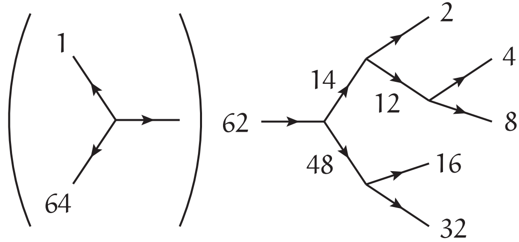
1. generate $\varphi \in [0, 2\pi]$ and $z \in [-1, 1]$
2. construct $q^0 = \sqrt{s_a + |\vec{q}|^2}$ and $\vec{q} = |\vec{q}|\left(\sqrt{1-z^2}\cos\varphi\,,\,\sqrt{1-z^2}\sin\varphi\,,\,z\right)$
3. q is p_a in the center-off-mass frame of P, and needs to be boosted:
$$p_a^\mu = \left(E, \vec{q} + V\vec{P}\,\right) \quad \text{with} \quad E = \frac{q\cdot P}{\sqrt{P^2}} \quad \text{and} \quad V = \frac{q^0 + E}{P^0 + \sqrt{P^2}}$$
4. and finally $p_b = P - p_a$

This construction gives a Jacobian $\dfrac{\sqrt{P^2}}{\pi|\vec{q}|} = \dfrac{2P^2}{\pi\sqrt{\lambda(P^2, s_a, s_b)}}$

n-particle phase space

To generate, for example, 5-particle phase space, choose a decomposition into 2-particle phase spaces.

External momenta labelled by a power of 2 and $p_i + p_j = p_{i+j}$ and $p_{2^{n+3}-1} = 0$, so for $n = 5$: $p_{127} = 0$ and $p_i = -p_{127-i}$



The density factor of the example graph is

$$g(\{p\}) = g_{48}(s_{48})\, g_{14}(s_{14})\, \frac{2s_{62}}{\pi\sqrt{\lambda(s_{64}, s_{48}, s_{14})}}\, g_{12}(s_{12})\, \frac{2s_{14}}{\pi\sqrt{\lambda(s_{14}, s_{12}, s_2)}} \times \frac{2s_{12}}{\pi\sqrt{\lambda(s_{12}, s_8, s_4)}}\, \frac{2s_{48}}{\pi\sqrt{\lambda(s_{48}, s_{32}, s_{16})}}$$

The virtual invariants (s_{48}, s_{14}, s_{12}) need to be generated, and one can use densities anticipating the behavior of the integrand

$$\text{e.g.} \quad g_{12}(s) \propto \frac{1}{(s - M_Z^2)^2 + \Gamma_Z^2 M_Z^2}$$

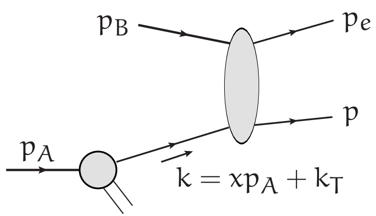
More graphs can be included via the multi-channel method. This way, the squared graphs in a squared amplitude can be matched, while interferences cannot.

Electron-proton scattering

Electron-proton scattering

$$\frac{d^2\sigma^{u-\mathrm{quark}}_{e^-p\to e^-X}}{dx_{Bj}\,dQ^2} = \int dx \int \frac{d^2k_T}{\pi}\,\mathcal{F}_u(x,|\vec{k}_T|,Q)\int d\Phi\big(p_B + k \to \{p_e, p\}\big)\,\frac{1}{2xs}\,|\overline{\mathcal{M}}(e^-u^* \to e^-u)|^2$$

$$\times\,\delta\big(Q^2 + (p_B - p_e)^2\big)\,\delta\left(x_{Bj} - \frac{Q^2}{2p_A\cdot(p_B - p_e)}\right)$$



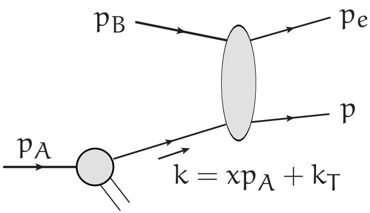
$$p_A^2 = p_B^2 = p_e^2 = p^2 = 0 \qquad s = 2p_A\cdot p_B \qquad y = \frac{Q^2}{x_{Bj}\,s}$$

collinear factorization: $\mathcal{F}_u(x,|\vec{k}_T|,Q) \to f_u(x,Q)\,\delta\big(|\vec{k}_T|^2\big)$

Electron-proton scattering

$$\frac{d^2\sigma^{u-\text{quark}}_{e^-p\to e^-X}}{dx_{Bj}\,dQ^2} = \int dx \int \frac{d^2k_T}{\pi}\,\mathcal{F}_u(x,|\vec{k}_T|,Q)\int d\Phi\big(p_B+k\to\{p_e,p\}\big)\,\frac{1}{2xs}\,|\overline{\mathcal{M}}(e^-u^*\to e^-u)|^2$$

$$\times\,\delta\big(Q^2+(p_B-p_e)^2\big)\,\delta\left(x_{Bj}-\frac{Q^2}{2p_A\cdot(p_B-p_e)}\right)$$



$$p_A^2 = p_B^2 = p_e^2 = p^2 = 0 \qquad s = 2p_A\cdot p_B \qquad y = \frac{Q^2}{x_{Bj}\,s}$$

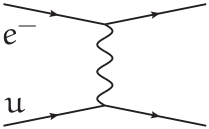
collinear factorization: $\mathcal{F}_u(x,|\vec{k}_T|,Q)\to f_u(x,Q)\,\delta\big(|\vec{k}_T|^2\big)$

$$\int d\Phi\big(p_B+k\to\{p_e,p\}\big)\,\delta\big(Q^2+(p_B-p_e)^2\big)\,\delta\left(x_{Bj}-\frac{Q^2}{2p_A\cdot(p_B-p_e)}\right)$$

$$=\left\{\text{collinear}:\frac{\delta(x-x_{Bj})}{8\pi\,x_{Bj}\,s}\;,\;\;k_T\text{—factorization}:\frac{1}{16\pi^2\,x_{Bj}^2\,s\sqrt{\Delta(x,k_T)}}\right\}$$

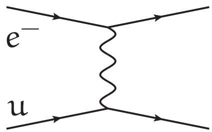
$$\Delta(x,|\vec{k}_T|) = -\prod_{i=1}^{4}\left(\frac{|\vec{k}_T|}{Q}\pm\sqrt{x/x_{Bj}-y}\pm\sqrt{1-y}\right)$$

Electron-proton scattering

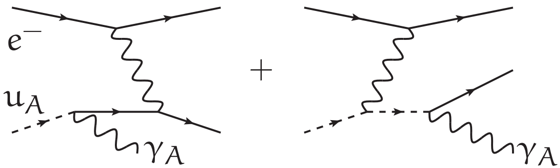


$$\frac{\left|\overline{\mathcal{M}}(e^- u \to e^- u)\right|^2}{(4\pi\alpha C_u)^2} = \frac{2xs}{Q^2}\frac{1+(1-\tilde{y})^2}{\tilde{y}} \quad , \quad \tilde{y} = \frac{x_{\mathrm{Bj}}}{x}y$$

Electron-proton scattering

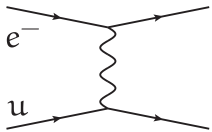


$$\frac{\left|\overline{\mathcal{M}}(e^-u \to e^-u)\right|^2}{(4\pi\alpha C_u)^2} = \frac{2xs}{Q^2}\frac{1+(1-\tilde{y})^2}{\tilde{y}} \quad , \quad \tilde{y} = \frac{x_{\mathrm{Bj}}}{x}\, y$$

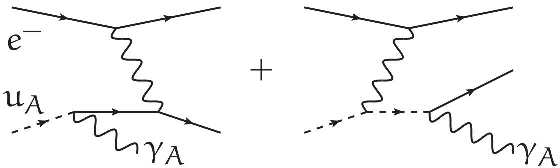


$$\frac{\left|\overline{\mathcal{M}}(e^-u^* \to e^-u)\right|^2}{(4\pi\alpha C_u)^2} = \frac{2xs}{Q^2}\frac{1+(1-y)^2}{y}$$

Electron-proton scattering



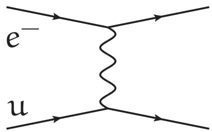
$$\frac{\left|\overline{\mathcal{M}}(e^- u \to e^- u)\right|^2}{(4\pi\alpha C_u)^2} = \frac{2xs}{Q^2}\,\frac{1+(1-\tilde{y})^2}{\tilde{y}} \quad , \quad \tilde{y} = \frac{x_{\mathrm{Bj}}}{x}\, y$$



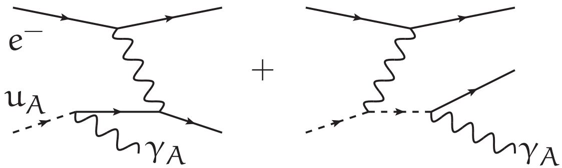
$$\frac{\left|\overline{\mathcal{M}}(e^- u^* \to e^- u)\right|^2}{(4\pi\alpha C_u)^2} = \frac{2xs}{Q^2}\,\frac{1+(1-y)^2}{y}$$

$$\frac{1}{2\pi\alpha^2}\,\frac{x_{\mathrm{Bj}}\, Q^4}{1+(1-y)^2}\,\frac{d^2\sigma^{u-\mathrm{quark}}_{e^-p\to e^-X}}{dx_{\mathrm{Bj}}\, dQ^2} = \frac{C_u^2}{\pi}\int_{Q^2/s}^{1} dx \int_0^\infty dk_T^2\, \mathcal{F}_u(x, k_T^2, Q)\, \frac{\theta\big(\Delta(x,k_T)\big)}{\sqrt{\Delta(x,k_T)}}$$

Electron-proton scattering



$$\frac{\left|\overline{\mathcal{M}}(e^- u \to e^- u)\right|^2}{(4\pi\alpha C_u)^2} = \frac{2xs}{Q^2}\,\frac{1+(1-\tilde{y})^2}{\tilde{y}} \quad , \quad \tilde{y} = \frac{x_{Bj}}{x}\,y$$



$$\frac{\left|\overline{\mathcal{M}}(e^- u^* \to e^- u)\right|^2}{(4\pi\alpha C_u)^2} = \frac{2xs}{Q^2}\,\frac{1+(1-y)^2}{y}$$

$$\frac{1}{2\pi\alpha^2}\,\frac{x_{Bj}\,Q^4}{1+(1-y)^2}\,\frac{d^2\sigma^{u-\text{quark}}_{e^-p\to e^-X}}{dx_{Bj}\,dQ^2} = \frac{C_u^2}{\pi}\int_{Q^2/s}^{1} dx \int_0^{\infty} dk_T^2\,\mathcal{F}_u(x,k_T^2,Q)\,\frac{\theta\big(\Delta(x,k_T)\big)}{\sqrt{\Delta(x,k_T)}}$$

$$\left[\kappa = 1+\frac{x}{x_{Bj}}-2y-2\cos(\pi\rho)\sqrt{(1-y)(\frac{x}{x_{Bj}}-y)}\right] \quad = C_u^2\,Q^2\int_{Q^2/s}^{1} dx \int_0^1 d\rho\,\mathcal{F}_u\left(x\,,\,Q^2\kappa(\rho,x)\,,\,Q\right)$$

$$\left[\xi = 1+\frac{k_T^2}{Q^2}-2\cos(\pi\rho)\sqrt{1-y}\,\frac{k_T}{Q}\right] \quad = C_u^2\,x_{Bj}\int_0^{Q^2\kappa_+(1)} dk_T^2 \int_0^1 d\rho\,\mathcal{F}_u\left(x_{Bj}\xi(\rho,k_T)\,,\,k_T^2\,,\,Q\right)$$

KaTie

https://bitbucket.org/hameren/katie

- parton level event generator, like Alpgen, Helac, MadGraph, etc.
- arbitrary hadron-hadron or hadron-lepton processes within the standard model (including effective Higgs-gluon coupling) with several final-state particles.
- 0, 1, or 2 off-shell intial states.
- produces (partially un)weighted event files, for example in the LHEF format.
- requires LHAPDF. TMD PDFs can be provided as files containing rectangular grids, or with TMDlib (Hautmann, Jung, Krämer, Mulders, Nocera, Rogers, Signori 2014).
- a calculation is steered by a single input file.
- employs an optimization stage in which the pre-samplers for all channels are optimized.
- during the generation stage several event files can be created in parallel.
- event files can be processed further by parton-shower program like CASCADE.
- (evaluation of) matrix elements separately available, including C++ interface.