New Monte Carlo Algorithms for Multi-Dimensional Integration with Hardware Acceleration

Andrea Pasquale

Università degli Studi di Milano - Corso di Laurea Magistrale in Fisica

13th July 2021

Monte Carlo integration in HEP

Monte Carlo Integration

Monte Carlo (MC) integration enables us to to solve complex multi-dimensional integrals

$$I = \int_V f(\mathbf{x}) d\mathbf{x} \tag{1}$$

by simply sampling the function f in N random points:

$$I \approx I_{\mathrm{MC}} = V \frac{1}{N} \sum_{\mathbf{x}_i \in V} f(\mathbf{x}_i) = V \langle f \rangle \tag{2}$$

The variance of I_{MC} can be computed using the previously sampled points as

$$\sigma_I^2 \approx \sigma_{\mathrm{MC}}^2 = \frac{1}{N-1} \left[V^2 \langle f^2 \rangle - I_{\mathrm{MC}}^2 \right] \Rightarrow \sigma_{\mathrm{MC}} \sim \frac{1}{\sqrt{N}} \tag{3}$$

Advantage of MC: the error depends only on $N \nRightarrow$ quadrature integration

Reducing the variance

Aim of MC algorithms: reduce the variance in the integral estimate through

- stratified sampling techniques ⇒ divide the integration region
- importance sampling techniques ⇒ sample using non-uniform distribution

$$\int_0^1 d^2x \sum_{i=1}^{3} e^{-50|\mathbf{x}-\mathbf{r}_i|} \quad \mathbf{r}_1 = (0.23, 0.23)\ , \quad \mathbf{r}_2 = (0.39, 0.39)\ , \quad \mathbf{r}_3 = (0.74, 0.74) \tag{4}$$

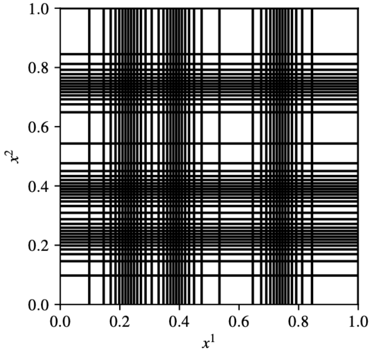


Figure: `VEGAS`: importance sampling grid

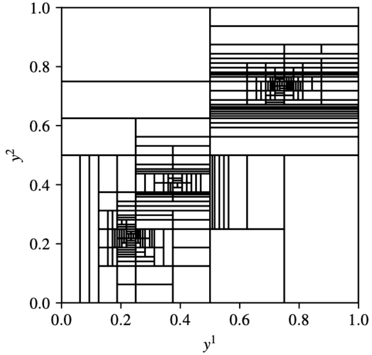


Figure: `MISER`: partition of the integration volume

The Standard Model

In the Standard Model we can predict the value of an observable as a series of terms

$$d\sigma = d\sigma^{\mathrm{LO}} + d\sigma^{\mathrm{NLO}} + d\sigma^{\mathrm{NNLO}} \dots \; . \tag{5}$$

every term is computed as

$$d\sigma \sim \underbrace{|\mathcal{M}(p_1, \dots, p_n)|^2}_{\substack{\text{scattering} \\ \text{amplitude}}} \times \underbrace{d\Phi_n(p_1, \dots, p_n)}_{\substack{\text{phase-space} \\ \text{density}}} \tag{6}$$

High-dimensional integrals arise from:

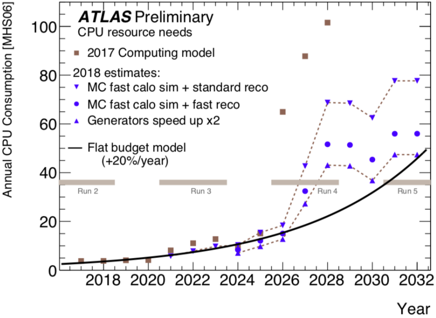
- $d\Phi_n(p_1, \dots, p_n) \; : \; D_{\mathrm{integral}} = 3 \times n - 4$
- $\mathcal{M}^{\text{L- loops}}(p_1, \dots, p_n) \; : \; D_{\mathrm{integral}} = 4 \times l$

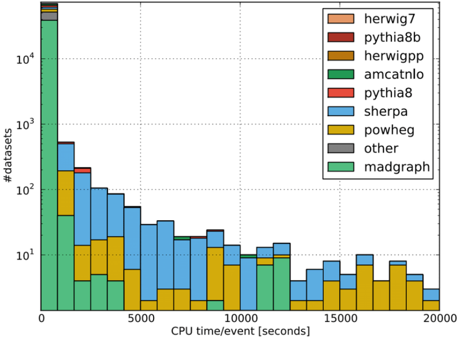
Higher order terms involve more looops and more particles resulting in complicated integrals to evaluate. Futhermore, they are usually peaked in small region of the integration volume near kinematics divergences which are difficult to sample.

Cost of Monte Carlo Integration

Problem: MC integration is computationally expensive for high accuracy requirements!

- High CPU resources
- Long computational times





The current integration algorithms will not be able to produce theoretical predictions that match the precision of the experimental data in the next years.

Solutions and aim of the thesis

Question:

- How can we obtain high-accuracy predictions at acceptable CPU costs and computational times?

Solution:

1. Develop new algorithms for multi-dimensional integration.
2. Look at new computer architecture: GPUs or multi-threading CPUs.

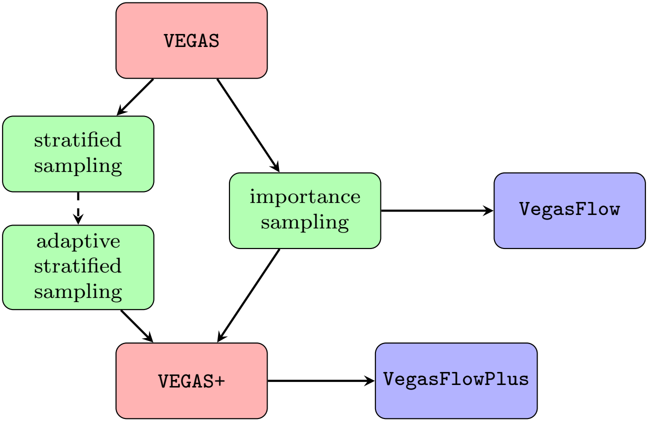
Outline of the thesis:

- Study of a new MC integrator effective with HEP integrands
- Implementation using hardware acceleration to lower the CPU usage
- Benchmark the performance of the new algorithm

Algorithms and implementation

Algorithms

We start from **VEGAS**, an algorithm for adaptive multi-dimensional MC integration implemented by Lepage in 1977.



We focus our analysis on **VEGAS+**, a new algorithm which employs a novel adaptive stratified sampling technique.

New features of VEGAS+

Adaptive stratified sampling of VEGAS+	**Stratified sampling of VEGAS**
Each hypercube h is sampled with a different number of points n_h which are adjusted iteratively. The integral and the variance are now computed as	Each hypercube h is sampled with the same number of points n_{ev}. The integral and the variance are computed as
$I = \frac{V}{N_{\text{st}}^D} \sum_h \frac{1}{n_h} \sum_{\mathbf{x} \in h} f(\mathbf{x}) = \sum_h I_h$	$I = \frac{V}{N_{\text{st}}^D} \sum_h \left(\frac{1}{n_{\text{ev}}} \sum_{\mathbf{x} \in h} f(\mathbf{x}) \right) = \sum_h I_h$
$\sigma_I^2 = \sum_h \sigma_h^2$	$\sigma_I^2 = \sum_h \sigma_h^2$

Samples redistribution algorithm

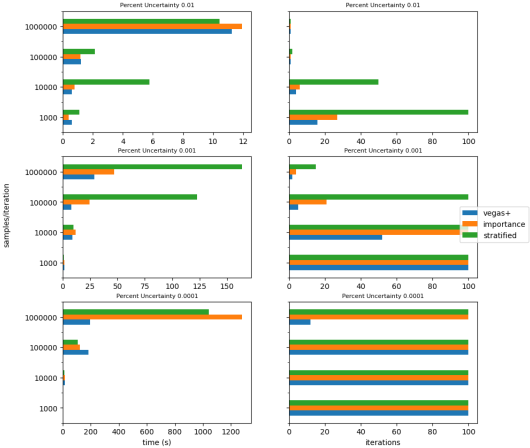
1. Choose number of stratifications $N_{\text{st}} = \lfloor (N_{\text{ev}}/4)^{1/D} \rfloor$
2. Accumulate the variance in each hypercube:

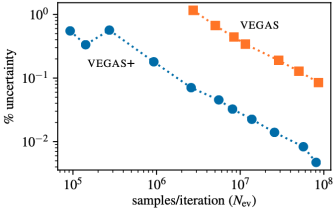
$$\sigma_h^2 \approx \frac{V_h^2}{n_h} \sum_{\mathbf{x} \in V_h} f^2(\mathbf{x}) - \left(\frac{V_h}{n_h} \sum_{\mathbf{x} \in V_h} f(\mathbf{x}) \right)^2$$

3. Replace the variance with d_h : $d_h \equiv \sigma_h^\beta$ with $\beta \geq 0$
4. Recalculate the number of samples for each hypercube for the next iteration

$$n_h = \max\left(2, d_h / \sum_{h'} d_{h'}\right)$$

Why do we choose to implement the **VEGAS+** algorithm?





$$\int_0^1 d^8x \sum_{i=1}^{3} e^{-50|\mathbf{x}-\mathbf{r}_i|}$$

$$\mathbf{r}_1 = (0.23, \ldots, 0.23)\ ,$$
$$\mathbf{r}_2 = (0.39, \ldots, 0.39)\ ,$$
$$\mathbf{r}_3 = (0.74, \ldots, 0.74)\ .$$

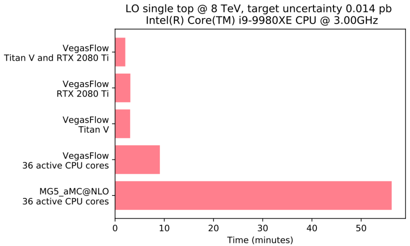
For the DY LO partonic level cross setion **VEGAS+** converge within the limit of 100 iterations when aiming at 0.0001% percent uncertainty.

VEGAS+ can overcome the poor performance of **VEGAS** with non-separable integrands. The redistribution of samples helps the importance sampling algorithm finding the peaks correctly.

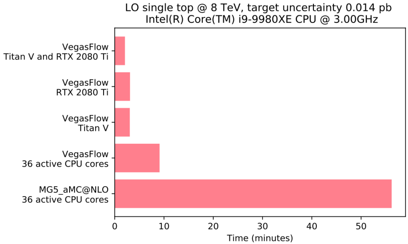
VegasFlow

`VegasFlow`: implementation of the Vegas importance sampling using hardware acceleration. This is possible thanks to the `TensorFlow` library which enable us to distribute python code to hardware acceleration devices.

Better computational times for physical integrand!



Advantage of graph implementation!



Our aim is to implement the `VEGAS+` algorithm within the `VegasFlow` library, empowering the algorithm by enabling to run the integration in GPUs.

Implementation of `VegasFlowPlus`

Details of the implementation

- Class derived from the `VegasFlow` integrator (same importance sampling algorithm)
- Adding stratified sampling : `generate_samples_in_hypercubes` + other modifications
- New feature of `VEGAS+`: `redistribute_samples`

Problems during the implementation:

- Number of events not constant ⇒ require `input_signature`
- Memory problem caused by `tf.repeat` ⇒ limit on number of hypercubes

Benchmark results

Benchmark results

Integration setup:

- warm-up of 5 iterations with 1M samples with grid refinement ($\alpha = 1.5$)
- 1M samples for each iteration after the warm-up

Integrator Name	Class	warm-up method	integration method
Importance Sampling	`VegasFlow`	importance	importance
Classic VEGAS	`VegasFlowPlus`	importance + stratified	importance + stratified
VEGAS/VEGAS+ hybrid	`VegasFlowPlus`	importance + adaptive	importance + stratified
VEGAS+	`VegasFlowPlus`	importance + adaptive	importance + adaptive

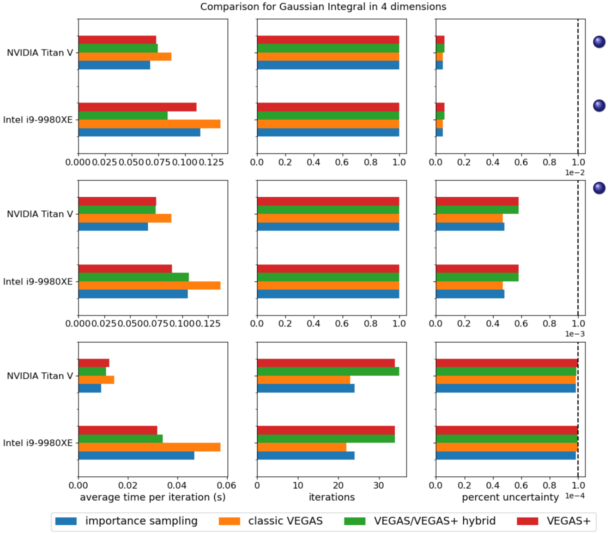
Hardware setup:

- professional-grade CPU: Intel(R) Core(TM) i9-9980XE - 36 threads
- professional-grade GPU: NVIDIA Titan V

We would like to answer the following questions:

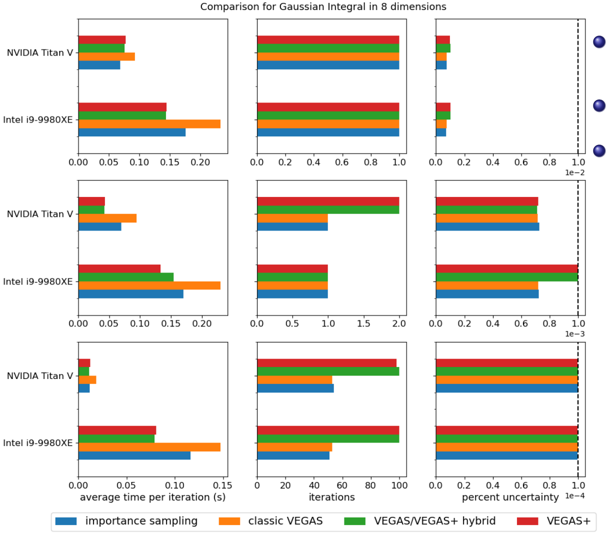
- Can `VegasFlowPlus` perform better than `VegasFlow`?
- Can `VegasFlowPlus` benefit from hardware acceleration?
- Can we provide the user with a *recipe* describing which integrators works best depending on the integral to compute?

Gaussian Integral Benchmark - Dimension 4



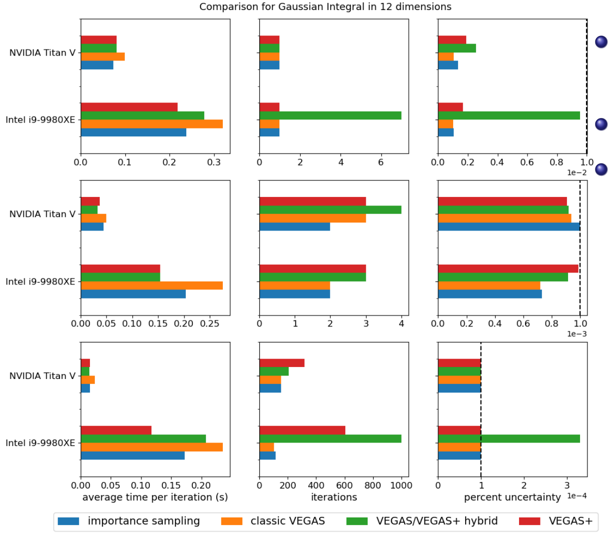
- classic VEGAS is the most accurate integrator followed by the importance sampling
- the VEGAS+ integrators are not effective since we are dealing with an integrand with a non-diagonal sharp peak
- benefits when running on GPU: up to 3x improvement

Gaussian Integral Benchmark - Dimension 8



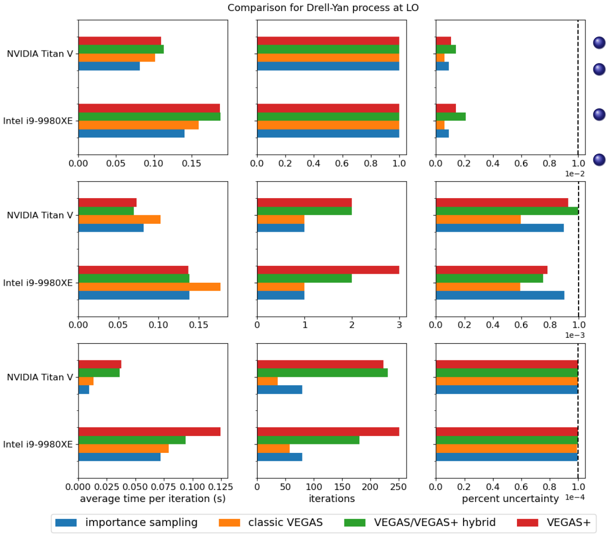
- trend similar to the previous case with worst perfomance of VEGAS+ integrators
- VEGAS+ algorithm fastest when running on CPU
- significant improvements in the computational times thanks to the graph implementation and the larger number of iterations

Gaussian Integral Benchmark - Dimension 12



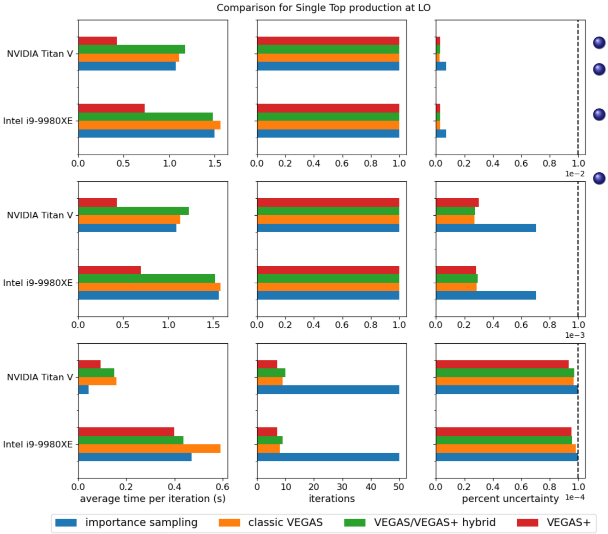
- classic VEGAS and importance sampling reach the accuracy required using the same number of iterations
- worst performance of the VEGAS+ algorithm
- speed-up factors: importance sampling 11.5, classic VEGAS 10.1, VEGAS/VEGAS+ hybrid 14.8 and VEGAS+ 7.8.

Drell-Yan at LO - partonic level



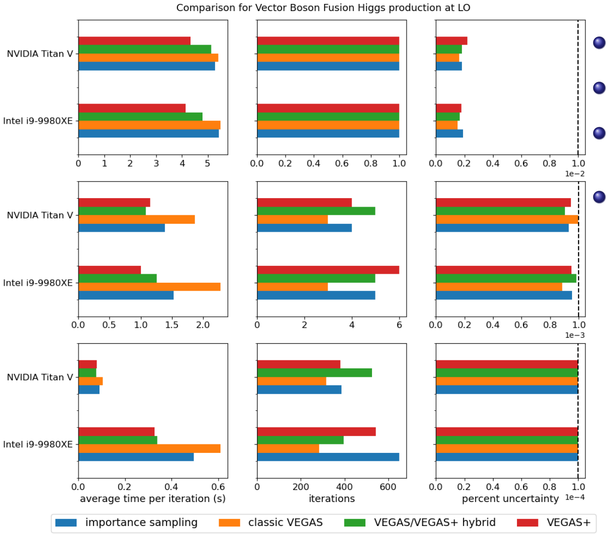
- physical integral with dimension 3
- classic VEGAS is the best performing integrator
- importance sampling is the fastest integrator both on CPU and GPU
- the worst performance of VEGAS+ suggests a that the integral has a sharp peak easily found by the VEGAS grid

Single Top Production at LO - partonic level



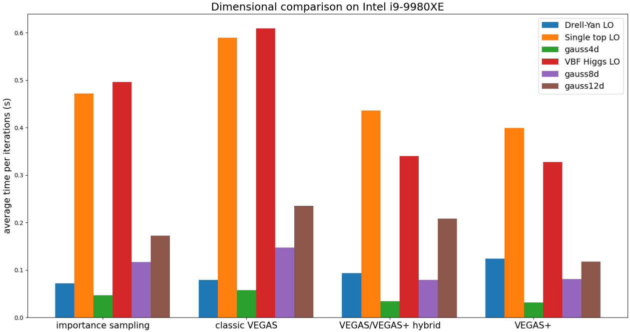
- physical integral of dimension 3
- the importance sampling is by far the less efficient integrator
- the adaptive stratified sampling of VEGAS+ is particulary effective for this integrand
- on GPU the importance sampling is still the fastest despite the worst performances

Vector Boson Fusion Higgs production at LO



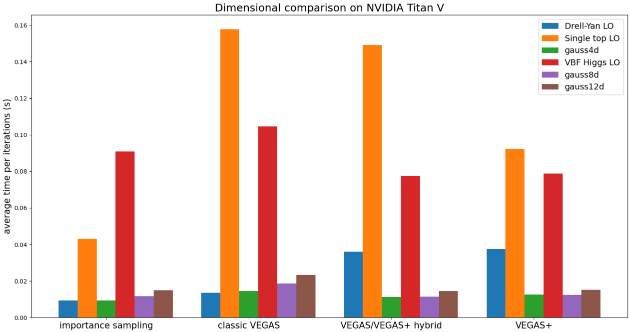
- physical integral of dimension 6 with convolution with PDFs
- classic VEGAS is the most efficient integrator overall
- significant benefits from GPU run, with speed-up factors between 4 and 6
- new algorithms more efficient than the importance sampling

Average time per iteration CPU



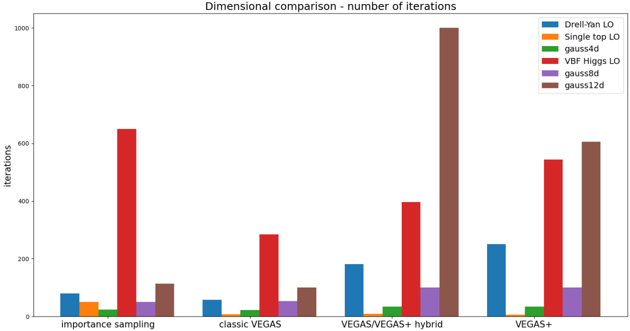
- VEGAS+ algorithms with shorter times when dealing with complex physical integrands such as Higgs or Top
- VEGAS+ is the fastest when dealing with a 12-dim Gauss distribution

Average time per iteration GPU



- fastest integrator importance sampling
- for VBF Higgs the VEGAS+ algorithms are faster than the importance sampling

Number of iterations



- classic VEGAS is the most efficient algorithm
- great performance of the new algorithms for the single Top production
- adaptive not effective with Gaussian due to the single sharp peak in high dimensions

Conclusions

In this thesis we have considered the problem of evaluating high-dimensional integrals in the context of HEP and the relative computational costs.
We focus on implementing the `VEGAS+` algorithm and we empower it by taking advantage of hardware acceleration.
The results of the benchmark show that:

- the implementation benefits from highly parallel scenarios ⇒ speed-up factors up to 10
- on CPU VEGAS+ is the fastest integrator
- new integrators more accurate when dealing with HEP integrands (Higgs or Single Top)
- classic VEGAS, a variation of VEGAS+, is the most efficient integrator

Future developments:

- implement new MC integration algorithms in `VegasFlow`
- machine-learning techniques for importance sampling

Back-up slides

Reducing the variance

Importance sampling

$$I = \underbrace{\int_V f(\mathbf{x})d\mathbf{x}}_{\substack{\text{integral of } f \text{ with}\\ \text{uniform sampling}}} = \int_V \frac{f(\mathbf{x})}{p(\mathbf{x})}p(\mathbf{x})d\mathbf{x}$$

$$= \underbrace{\int_V \frac{f(\mathbf{x})}{p(\mathbf{x})}dP(\mathbf{x})}_{\substack{\text{integral of } f/p\\ \text{with sampling } dP}}$$

Therefore, we can estimate the integral as

$$I_{\mathrm{MC}} = V\langle f/p\rangle_P$$

$$\sigma_I^2 \approx \sigma_{\mathrm{MC}}^2 = \frac{V^2}{N-1}\underbrace{\left[\langle (f/p)^2\rangle_P - \langle f/p\rangle_P^2\right]}_{=0 \text{ for } f=p}$$

Aim: find a function p that resemble the shape of f through adaptive recursive techniques.

Disadvantage:

Stratified Sampling

If we divide the integration volume in two subvolumes a and b, another estimator for the mean value of the function f is

$$\langle f\rangle' \equiv \frac{1}{2}(\langle f\rangle_a + \langle f\rangle_b)$$

with variance

$$\mathrm{Var}(\langle f\rangle') = \frac{1}{2N}\left[\mathrm{Var}_a(f) + \mathrm{Var}_b(f)\right]$$

While the variance of f is

$$\mathrm{Var}(f) = \underbrace{\frac{1}{2}[\mathrm{Var}_a(f) + \mathrm{Var}_b(f)]}_{\propto \mathrm{Var}(\langle f\rangle')} + \underbrace{\frac{1}{4}(\langle\!\langle f\rangle\!\rangle_a - \langle\!\langle f\rangle\!\rangle_b)^2}_{\geq 0} \ .$$

Aim: divide the integration domain in several subvolumes to reduce the variance

Disadvantage: we need at leat two points in each subvolume to compute the variance.

Theoretical prediction and Standard Model

In a generic Quantum Field Theory we can predict the value of an observable, such as the differential cross section, in the following way

$$d\sigma = \frac{1}{4E_A E_B |v_A - v_B|} \underbrace{d\Pi_n}_{\substack{\text{phase-space}\\ \text{density}}} \times \underbrace{|\mathcal{M}(k_A, k_B \to p_1, ..., p_n)|^2}_{\text{invariant scattering amplitude}} \tag{7}$$

The integration over the phase-space is of the form

$$\int d\Pi_n = \left(\prod_{i=1}^{n} \int \frac{d^3 p_i}{(2\pi)^3} \frac{1}{2E_i}\right) (2\pi)^4 \delta^{(4)}\Big(k_A + k_B - \sum_{i=1}^{n} p_i\Big) \ , \tag{8}$$

which corresponds to a $3n - 4$ dimensional integral.
The matrix element is computed using Feynman diagrams by combining the real emissions and the loop corrections to avoid IR and UV divergences.

$$\mathcal{M} = \begin{cases} \mathcal{M}^{\text{tree}} + \mathcal{M}^{\text{1-loop}} + \mathcal{M}^{\text{2-loops}} + \dots & \text{quantum corrections} \\ \mathcal{M}^{\text{tree}} + \mathcal{M}^{\text{1-leg}} + \mathcal{M}^{\text{2-legs}} + \dots & \text{real emissions} \end{cases} \tag{9}$$

The final expression will be of the form

$$d\sigma = d\sigma^{\text{LO}} + d\sigma^{\text{NLO}} + d\sigma^{\text{NNLO}} \dots \ . \tag{10}$$

By aiming at higher precisions we will encounter several complex multi-dimensional integrals:

- adding loop to a diagram $\Rightarrow D_{\text{loop}} = D_{\text{diagram}} + 4$
- adding external leg to a diagram $\Rightarrow D_{\text{leg}} = D_{\text{diagram}} + 3$
- more complex diagrams $\Rightarrow$ more difficult integral evaluation
- in QCD we also need to compute the convolution with the Parton Density Functions (PDFs) according to the QCD factorization theorem

$$d\sigma = \sum_{a,b} \int_0^1 dx_a dx_b \sum_F \int d\Phi_F \underbrace{f_{a/h_1}(x_a, \mu_F) f_{b/h_2}(x_b, \mu_F)}_{\text{parton density functions}} \underbrace{d\hat{\sigma}_{ab\to F}}_{\substack{\text{partonic}\\ \text{cross}\\ \text{section}}} \ . \tag{11}$$

The squared matrix element $|\mathcal{M}^2|$ is difficult to sample since it is particularly peaked in a small region of the integration domain, usually near kinematics divergences.
These regions become even smaller for high-dimensional integrals:

$$\frac{V_{\text{hypersphere}}}{V_{\text{hypercube}}} = \frac{1}{2^D} \frac{\pi^{\frac{D}{2}}}{\Gamma\left(\frac{D}{2}+1\right)} \approx \left(\frac{\sqrt{\pi}}{2}\right)^D \xrightarrow{D\to\infty} 0 \ , \tag{12}$$

VEGAS

VEGAS is an algorithm for adaptive multi-dimensional MC integration implemented by Lepage in 1977. It is the main drive for QCD fixed-order calculations programs such as MCFM, NNLOJET, MG5 and Sherpa.

Importance Sampling
The sampling distribution used is *separable*

$$p \propto g(x_1, x_2, x_3, \dots, x_n) = g_1(x_1) g_2(x_2) g_3(x_3) \dots g_n(x_n) \; .$$

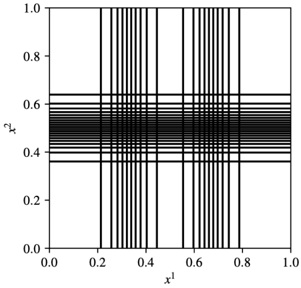
The algorithm divides the integration domain in subintervals with probability

$$g_i(x) = \frac{1}{N \Delta x_i}$$

after each iteration the intervals Δx_i are adjusted iteratively using the quantity

$$d_i \equiv \frac{1}{n_i} \sum_{x_j \in [x_i - \Delta x_i, x_i]} f^2(\mathbf{x}) \approx \Delta x_i \int_{x_i - \Delta x_i}^{x_i} d\mathbf{x} f^2(\mathbf{x})$$

$$\int_0^1 d^4x \left(e^{-100(\mathbf{x}-\mathbf{r}_1)^2} + e^{-100(\mathbf{x}-\mathbf{r}_2)^2} \right)$$



$$\mathbf{r}_1 = (0.33, 0.5, 0.5, 0.5)$$

$$\mathbf{r}_2 = (0.67, 0.5, 0.5, 0.5)$$

Stratified sampling

- Each axis is divided into a fixed number of stratifications $N_{\rm st} = \lfloor (N_{\rm ev}/2)^{1/D} \rfloor$ resulting in $N_{\rm st}$ hypercubes.
- Every hypercube is sampled with $n_{\rm ev}$ points : $n_{\rm ev} = \lfloor (N_{\rm ev}/N_{\rm st}^D) \rfloor \geq 2$
- The integral and the variance are computed as

$$I = \frac{V}{N_{\rm st}^D} \sum_h \left(\frac{1}{n_{\rm ev}} \sum_{\mathbf{x} \in h} f(\mathbf{x}) \right) = \sum_h I_h \quad , \quad \sigma_I^2 = \sum_h \sigma_h^2$$

Limitations of VEGAS

- not all integrands are separable
- stratified sampling uneffective for high-dimensional integrals

VEGAS+ algorithm

Problem: `VEGAS` (importance + stratified sampling) struggles with non-separable integrals.
Solution: `VEGAS+` (importance + *adaptive* stratified sampling).

Adaptive stratified sampling
Each hypercube h is sampled with a different number of points $n_h \neq n_{\rm ev}$ which are adjusted iteratively. The integral and the variance are now computed as

$$I = \frac{V}{N_{\rm st}^D} \sum_h \frac{1}{n_h} \sum_{\mathbf{x} \in h} f(\mathbf{x}) = \sum_h I_h \quad , \quad \sigma_I^2 = \sum_h \frac{\sigma_h^2}{n_h}$$

`VEGAS+` algorithm:

1. Choose number of stratifications $N_{\rm st} = \lfloor (N_{\rm ev}/4)^{1/D} \rfloor$
2. Accumulate the variance in each hypercube:

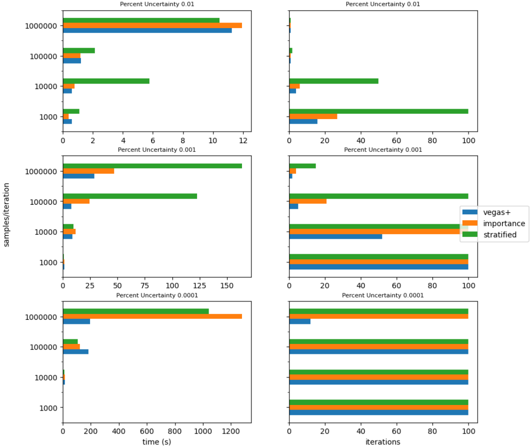
$$\sigma_h^2 \approx \frac{V_h^2}{n_h} \sum_{\mathbf{x} \in V_h} f^2(\mathbf{x}) - \left(\frac{V_h}{n_h} \sum_{\mathbf{x} \in V_h} f(\mathbf{x}) \right)^2$$

3. Replace the variance with d_h : $d_h \equiv \sigma_h^\beta$ with $\beta \geq 0$
4. Recalculate the number of samples for each hypercube for the next iteration

$$n_h = \max\left(2, d_h / \sum_{h'} d_{h'}\right)$$

A new implementation VegasFlowPlus

Novel implementation of the **VEGAS+** algorithm within **VegasFlow**: **VegasFlowPlus**.



Motivation: Several tests showed that **VEGAS+** can outperform the importance sampling of **VEGAS**, especially for physical integrands.

For the DY LO partonic level cross setion **VEGAS+** converge within the limit of 100 iterations when aiming at 0.0001% percent uncertainty.

These tests were performed with the single CPU implementation of **VEGAS** and **VEGAS+** currently available at `https://github.com/gplepage/vegas`

DY at LO

The first example that we consider is the Drell-Yan (DY) process, which consists in an inclusive production of high-mass lepton pairs in hadron-hadron collision:

$$h_A + h_B \to (\gamma^* + l\bar{l}) + X \ ,$$

where X is an undetected final state.