2 Monte Carlo simulations

The Monte Carlo methods use a statistical approach to study phenomena. The idea is to solve problems that could be deterministic in nature through stochastic numerical methods. To do so, different methods of random sampling of statistical variables have been developed which go by the name of Monte Carlo.

Even if none of these methods is bound to computer simulations, the growth of computer power in the last years allowed the promotion of random sampling as an effective tool to study incredibly complex phenomena through numerical simulations in various natural fields.

In particular, in particle physics Monte Carlo methods are used throughout the entire simulation of a scattering process. The program used for event generations from matrix element is MadGraph[**madgraph**] which accounts for the events generations for a selected channel from a given model, e.g. Standard Model. This is accompanied by tools that simulate the hadronization and the parton showering like Sherpa[**sherpa**] and Pythia[**pythia**]. On the other hand, to fully simulate a real event it is necessary to simulate the interaction between particles and the detector materials, this is done mainly through Geant4[**geant**].

The first section of this chapter will describe how to use Monte Carlo methods to perform multi-dimensional integration, followed by a summary of the tool used to generate events in this thesis: MadGraph5.

2.1 Monte Carlo integration

The problem of performing multi-dimensional integrals in particle physics is tied up to the calculation of cross sections. Indeed, selected a process with n final state particles and a matrix element $|M|^2$ the cross section is given by:

$$\sigma = \frac{1}{2s} \int |M|^2 d\Phi(n) \tag{2.1}$$

where s is the center of mass energy and $\Phi(n)$ is the $3n$-dimensional phase space. In order to perform integrals of such kind consider a d-dimension integral over the space V of a function $f(x)$:

$$I = \int_\Omega d^d x f(\bar{x}) p(\bar{x}) \tag{2.2}$$

where:

$$\int_\Omega p(\bar{x}) d^d x = 1 \qquad p(\bar{x}) \geq 0 \quad \forall \bar{x} \in \Omega \tag{2.3}$$

An estimate of this integral can be obtained sampling N random d-dimensional vectors $\bar{x}_i$ from the probability distribution function $p(\bar{x})$ and then evaluate the function $f(\bar{x}_i)$.

The approximated formula for I is given by:

$$I_{MC} = \frac{1}{N} \sum_{i=1}^{N} f(\bar{x}_i) \quad \xrightarrow[N\to\infty]{} \quad I \tag{2.4}$$

which tends to I as $N \to \infty$.

The quantity I_{MC} must be accompanied by its error that can be shown to be for a Monte Carlo process with large N:

$$\sigma = \frac{\sigma_I}{\sqrt{N}} \qquad \sigma_I = \int_\Omega f^2(\bar{x}) p(\bar{x}) d^d x - I^2 \tag{2.5}$$

where σ_I is the variance of the integral.

If this error is compared with the ones obtained from numerical methods is clear that Monte Carlo integration becomes better with high-dimensional spaces. Using a numerical method the error depends linearly on the number of points used, in case of d dimension the number of points will be $N = n^d$ where n is the fixed spacing along the axis. For the Simpson and the trapezoidal methods the error is proportional to:

$$\epsilon_{Simp} \propto \frac{1}{n^4} \propto \frac{1}{N^{4/d}} \tag{2.6}$$

$$\epsilon_{trap} \propto \frac{1}{n^2} \propto \frac{1}{N^{2/d}} \tag{2.7}$$

while with the MC integration it is always $\epsilon_{MC} \propto \frac{1}{\sqrt{N}}$ implying that already with 6-8 dimensions the Monte Carlo method is more efficient.

Importance sampling

Besides the effectiveness of the Monte Carlo integration, various techniques allow to reduce the variance of the integral. One of the most used is called ”Importance Sampling”.

Given an integral as in Eq. 1.2, $p(\bar{x})$ could not be the best probability distribution to minimize the variance. In order to find it, introduce a new function $g(\bar{x})$ where $g(\bar{x}) \geq 0$ and $\int_\Omega g(\bar{x}) d^d x = 1$ like:

$$I = \int_\Omega \left[\frac{f(\bar{x}) p(\bar{x})}{g(\bar{x})} \right] g(\bar{x}) d^d x \tag{2.8}$$

to find the best $g(\bar{x})$ minimize the Lagrangian L which contains the constraint on $g(\bar{x})$ and the variance of the integral with respect to $g(\bar{x})$:

$$L\left\{g\right\} = \int_\Omega \left[\frac{f(\bar{x}) p(\bar{x})}{g(\bar{x})} \right]^2 g(\bar{x}) d^d x + \lambda \left(\int_\Omega d^d x g(\bar{x}) - 1 \right) \tag{2.9}$$

this gives as best probability function $g(\bar{x}) = \lambda |f(\bar{x}) p(\bar{x})|$, where λ is a Lagrange multiplier and can be found imposing $\int_\Omega g(\bar{x}) d^d x = 1$.

If $f(\bar{x}) \geq 0$, $\lambda = 1/I$ and an estimator for the integral is:

$$I_{RS} = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\bar{x}_i) p(\bar{x}_i)}{g(\bar{x}_i)} = \frac{1}{N} \sum_{i=1}^{N} I \frac{f(\bar{x}_i) p(\bar{x}_i)}{f(\bar{x}_i) p(\bar{x}_i)} = I \tag{2.10}$$

where the $\bar{x}_i$ are drawn from the probability distribution function $g(\bar{x})$ and it has zero variance if the integral value is known. Considering that it is the unknown variable instead, the variance can be reduced choosing a function close to the initial integrating function such as its Taylor expansion.

An example of the effects of the Importance Sampling technique on an exponential function is shown in Fig. 1.1.

Rejection method

The rejection method is a technique used to sample random variables from any probability distribution function. Given the target probability distribution function $f(x)$ the method works as follow:

- Find a second probability distribution function $g(x)$ integrable which is known how to sample from, and such that $g(x) \geq f(x)$ for all x in the interested range;
- Generate a number x_i distributed according to $g(x)$;
- Accept or reject the proposed number with probability $p_{acc} = \frac{f(x_i)}{g(x_i)}$.

To check that the sampling occur from the correct distribution, define the probability to generate a number between x and $x+dx$ such that $p_{gen} dx = g(x) dx$. The sampled distribution is the product between p_{gen} and p_{acc}:

$$p_{samp} dx = p_{gen} p_{acc} dx = g(x) \frac{f(x}{g(x)} dx = f(x) dx \tag{2.11}$$

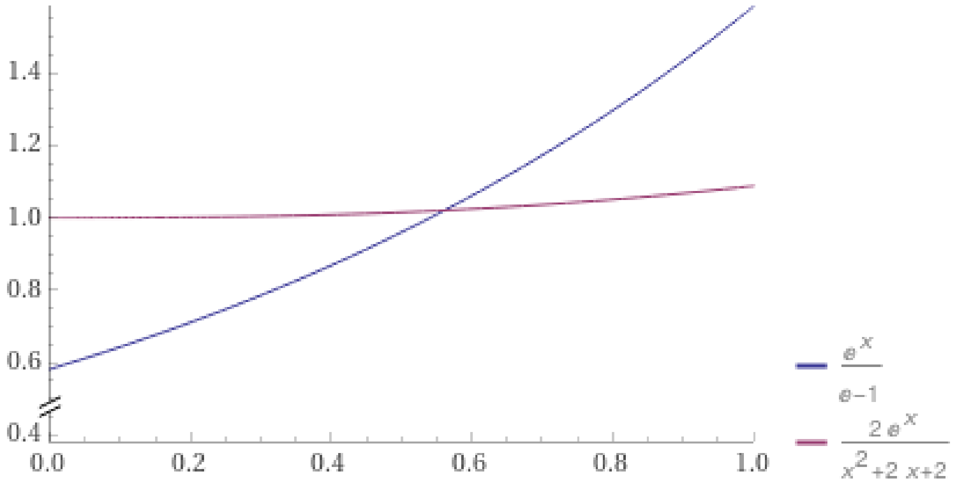

Figura 2.1: Effects of the importance sampling on $f(x) = \frac{e^x}{e-1}$ in the range $[0, 1]$. The initial integrating function (blue) has been divided by its Taylor expansion to the second order. The final integrating function (red) is much closer to a uniform distribution in the given range.

the sampled function is correctly $f(x)$. In practice, this is done by sampling a second random number y from a uniform distribution in the range $[0, 1]$ and accepting the initial variable if $y < \frac{f(x_i)}{g(x_i)}$. An example of the usage of this method is showed in Fig. 1.2.

From this outline of the methods, its advantages are clear:

- It is not needed the knowledge of the normalization of the probability distribution function nor its cumulative distribution function;
- The better $g(x)$ approximates to $f(x)$ the more efficient the method is because less proposals are rejected.

On the other hand, the more important drawback is that it can be inefficient because two random number must be generated and there is the possibility to completely lose them by rejecting the proposal.

2.2 MadGraph5

MadGraph5[**MG5**] is a tool for automatically generating matrix elements for High Energy Physics processes. It is an open source project written in Python able to generate matrix elements at tree level for any Lagrangian model implementable with FeynRules and perform computations in the Standard Model at Next to Leading Order.

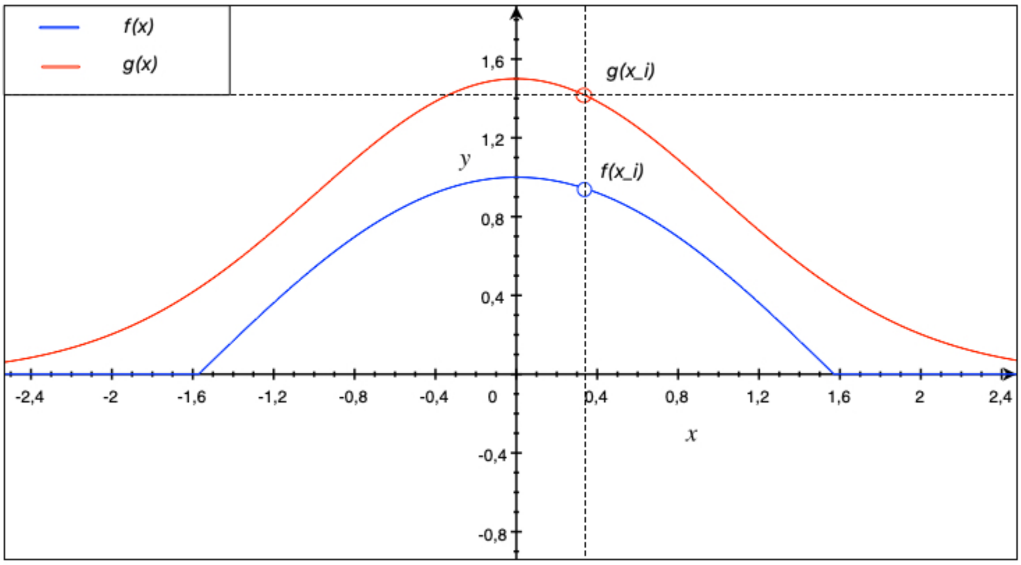

Figura 2.2: Rejection technique for the function $f(x)$ (blue) sampling random variables from the function $g(x)$ (red). A better approximation of $f(x)$ means that, after generating a number x_i, the ratio $\frac{f(x_i)}{g(x_i)} \approx 1$. Therefore, the probability to sample a number y from $[0, 1]$ uniformly such that $y < \frac{f(x_i)}{g(x_i)}$ is reduced, thus leading to a more efficient random sampling.

To generate the events, which can be defined as a possible collision occurring in a detector, three steps are necessary:

- Determine all the production channels finding the Feynman diagrams with the given initial and final particles;
- Calculate the full matrix element $|M|^2 = MM^*$ where M is the sum of the Feynman diagrams;
- Integrate over the phase space to obtain the cross section as shown in Eq. 1.1

The algorithm used by MadGraph5 checks for the topologies only allowed by the model optimizing the generation procedure instead of checking for all possible topologies. To this end, two dictionaries are defined: the first called `Vertices` which maps all combination of n particles to the possible n-point interactions and the second called `Currents` which maps, for all n-point interactions, $n-1$ particles to all combinations of resulting particles for the interactions.

Given these two maps, the flag `from_group` (FG) indicates if, according to the vertices, a particle results from the merging of particles. With these tools, the algorithm will follow the following step to generate the matrix elements:

- Consider all the particles as outgoing and set the flag FG `True` for external particles;
- If a vertex can combine all the external particles create the diagram if at least two particles have FG sets as `True`;
- Group the particles with at least one FG = `True` in accordance to the currents dictionary;
- Set FG = `True` for the combined particles and repeat from the first step with these new external particles;
- Finally, stop the algorithm when at most two particles remain.

An example of the algorithm for the process $q\bar{q} \to t\bar{t}g$ is presented in Tab. 1.1.

Once all the diagrams have been generated, the full squared matrix element is evaluated. This problem could be intensive and time consuming, indeed, considering a process with N diagrams, an analytical computation leads to the calculation of N^2 matrix terms. Additionally, the number of diagrams scales factorially with the number of particles, so given n particles the number of diagrams scales like $n!^2$. MadGraph5 uses various techniques to optimize this process, firstly the squared matrix element is calculated in terms of helicity wave functions. From the starting wave functions of the external legs, they are combined to obtain new wave functions

Initial particles	Groups	Replacements	Result
$q, \bar{q}, t, \bar{t}, g$	$(q, \bar{q}), t, \bar{t}, g$	$(g), t, \bar{t}, g$	Fail, Only one FG = `True`
	$q, \bar{q}, (t, \bar{t}), g$	$q, \bar{q}, (g), g$	Fail, Only one FG = `True`
	$q, \bar{q}, t, (\bar{t}, g)$	$q, \bar{q}, (\bar{t}), g$	Fail, Only one FG = `True`
	$(q, g)\bar{q}, t, \bar{t}$	$(q), \bar{q}, t, \bar{t}$	Fail, Only one FG = `True`
	$(q, \bar{q}), (t, \bar{t}), g$	$(g), (g), g$	Diagram 1, two FG = `True`
	$(q, \bar{q}), t, (\bar{t}, g)$	$(g), t, (\bar{t})$	Diagram 2, two FG = `True`
	$(q, \bar{q}), (t, g), \bar{t}$	$(g), (t), \bar{t}$	Diagram 3, two FG = `True`
	$(q, g), (t, \bar{t}), \bar{q}$	$(q), (g), \bar{q}$	Diagram 4, two FG = `True`
	$q, (\bar{q}, g), (t, \bar{t})$	$q, (\bar{q}), (g)$	Diagram 5, two FG = `True`

Tabella 2.1: Summary of the algorithm used to generate matrix element for the process $q\bar{q} \to t\bar{t}g$, where q denotes the possible quarks in a proton-proton scattering; referring to the Standard Model the possible interaction vertices are $(q, \bar{q}, g)$, $(t, \bar{t}, g)$ and (g, g, g), as presented in Appendix A.1.

corresponding to the propagators in the diagram by successive helicity wave function calls.

Furthermore, wave functions that lead to the same sub-diagram can be reutilized between diagrams, effectively recycling all the already seen sub-diagrams and reducing the scaling complexity of the problem.

With these implementations the number of terms to calculate in the squared matrix element is reduced to N and the scaling of the number of terms with the number of particles approximately to $(n-1)!2^{n/2}$.

To perform the final phase space integration, the aspect of the integrating function needs to be considered. Generally, the phase space presents sharp peaks in different regions and to do an efficient integration the positions of these peaks must be found and properly mapped.

The approach used by MadGraph5 is called "Single-Diagram-Enhanced multi channel integration". The idea behind this method is to decompose the initial integrating function f in a sum of components such that:

$$f = \sum_{i=1}^{n} f_i \qquad \text{with} \qquad f_i \geq 0 \forall i \tag{2.12}$$

Each of these components, referred as channels, should contains an easy to map peak function to get a much simpler integral for the single term.

Considering the particular problem the decomposition is the one that contains the single diagram squared matrix element:

$$f_i = \frac{|M_i|^2}{\sum_j |M_j|^2} |M_{tot}|^2 \tag{2.13}$$

where $|M_i|^2$ is the squared matrix element of a single diagram and $M_{tot} = \sum_i M_i$ is the full matrix element. Doing so, the initial complex integral is reduced in n independent integrations where the structure is known from the propagators:

$$\int |M_{tot}|^2 = \int \frac{\sum_i |M_i|^2}{\sum_j |M_j|^2|} |M_{tot}|^2 = \sum_i \int \frac{|M_i|^2}{\sum_j |M_j|^2} |M_{tot}|^2 \tag{2.14}$$

To further improve the results of the Monte Carlo integration for each of these channel is used Importance Sampling. This imply, as shown in Sec. 2.1, the definition of n new functions g_i each one similar to the f_i. Then, the final phase space integration is:

$$I = \int d\Phi f(\Phi) = \sum_{i=1}^{n} \int d\Phi g_i(\Phi) \frac{f_i(\Phi)}{g_i(\Phi)} = \sum_{i=1}^{n} I_i \tag{2.15}$$

given the basis of Eq. 1.13, the structure of the g_i is easily derived from the propagator structure of the corresponding diagram. The advantages of the splitting into n integration channels can be summarized in these points:

- More efficient single numerical integration;
- Errors are added in quadrature, so the procedure does not worsen the final result;
- The weight functions correspond to the single squared matrix diagram, so they can be calculated during the evaluation of $|M_{tot}|^2$;
- The procedure is parallel in nature, each channel can be calculated on a different node of a cluster and then combined to get the final result.

The events generated by MadGraph are *unweighted* events, this means that each event has the same weight and the number of events is proportional to a selected density of the phase space. The obtained unweighted distribution is the one that is expected to have in nature from a real experiment. On the other hand, a *weighted* sample has the same number of events for different probabilities in the phase space, resulting in an inevitable different weight for each event.

However, as described above, the integrator generates points in the phase space distributed accordingly an adaptive function and not its the exact measure, so the events have different Monte Carlo weights. In addition, the channel splitting introduces a second weight related to the probability of occurrence of the single channel. The unweighting procedures consist of using the rejection method to generate events for the different channels and keeping only the ones that have the same final weight that is defined as the product of the two weights.