Ntuples and compact matrix element representations

Daniel Maître
Institute for Particle Physics Phenomenology, Physics Department, South Road, Durham DH6 5FE, UK
E-mail: daniel.maitre@durham.ac.uk

Abstract. In this contribution I will discuss the practicalities of storing events from a NNLO calculation on disk with the view of "replaying" the simulation for a different analysis and under different conditions, such as a different PDF fit or a different scale setting. I also present a way to store a compact representation of the matrix elements for low multiplicity processes.

1. Ntuples

Higher order predictions are CPU intensive and are needed to a high level of statistical precision. Figure 1 depicts the typical prediction workflow. With the high resource cost involved in generating these predictions it makes sense to try to amortise their cost by saving the computationally expensive parts in files in order to be able to reuse the stored values for calculations that differ only in the cheaper parts of the calculation.

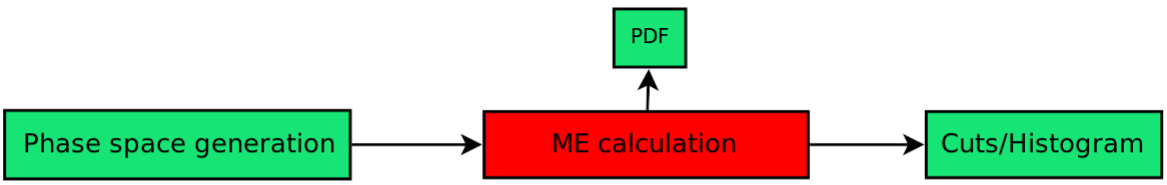

Figure 1. Standard prediction workflow: phase-space configurations are generated, the matrix elements are computed, they are combined with the value from the parton distribution function and binned in histograms. The ME calculation is the most costly operation and is highlighted in red.

Figure 2 shows the workflow when the generator is replaced by a process reading the event files. The advantage of the event file method include the fact that event files are easier to validate, share and handle than the full program that created it. The main disadvantage is the size of the files that are necessary to guarantee enough statistics for a large enough set of analysis. The size of event files have been investigated in [1, 2] for NNLO three jets production in e^+e^- collisions and in [3] for two-jet production at hadron colliders. The mechanics of how to implement nTuples for NNLO generators have been described at a previous ACAT [4].

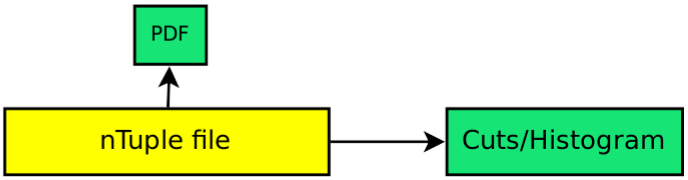

Figure 2. Ntuple prediction workflow: the event file is read to reproduce the weights and kinematic information stored.

2. Compact matrix element representation

The previous section described the trade-off between generality of the event file and its size. In their current form the NLO [5] or NNLO event files [3] emulate the integral

$$\sigma = \int d\phi_n \frac{d\sigma}{d\phi} C(\phi) \ , \tag{1}$$

where C is a set of cuts and ϕ is the phase-space information, including the initial state information. Because the event files encapsulate both initial and final state information they are more or less tied to a hadron collider energy: if the energy is changed substantially the mix of initial state flavours and initial momenta in the file will not be appropriate for a reliable prediction at the new energy. In this section we present a method to try to encode the information about the matrix elements of the higher order calculation in a way that is independent of the details of the hadronic initial state. This means one would need to perform the integration over the initial state momentum fraction explicitly.

At leading order the matrix elements are positive definite and can be interpreted as a probability density. This is not the case for higher order predictions where negative contributions have to be added. But as far as infrared safe observables are concerned there should only be a positive contribution after all contributions are summed. The goal in this contribution is to build a basis of phasespace functions from which one can build any infrared safe observable. In the following we only consider a $1 \to n$ process such as e^+e^- annihilation or Higgs decay.

We introduce as set of orthonormal functions of phasespace, the exact form of the parametrisation is not relevant to this discussion, only that we map phasespace coordinates to either the $[0, 1]$ or $[-1, 1]$ interval. We can also arrange for the Jacobian to be flat, such that the integration of any phasespace function can be written as:

$$\int d\phi f(\phi) = \int dx_1 \cdots dx_k f(\phi(x_1, ..., x_k)) \tag{2}$$

We introduce orthogonal polynomial functions Φ in one dimension

$$\int_{x_{min}}^{x_{max}} \Phi_i(x)\Phi_j(x)\, dx = \delta_{ij} \tag{3}$$

We can build basis functions for the phasespace by multiplying one-dimensional orthogonal functions for each dimension:

$$\Phi_{i_1,...,i_k}(x_1, ..., x_k) = \Phi^{(1)}_{i_1}(x_1)...\Phi^{(k)}_{i_k}(x_k) \ . \tag{4}$$

So we can now write any (reasonable) function of phase-space as

$$f(\phi) \approx \sum_{i_1,\ldots,i_k} c_{i_1,\ldots,i_k} \Phi_{i_1,\ldots,i_k}(\phi) \tag{5}$$

The number of indices corresponds to the number of free parameters (including azimuthal symmetry) as listed in Table 1. The number of parameters grows quickly as a function of the multiplicity and is not sustainable for large multiplicities where it would be intractable to calculate enough coefficients to obtain a good description of the function.

Table 1. Number of coordinates for the basis functions as a function of the multiplicity.

Multiplicity	number of coordinates
2	1
3	4
4	7
n	$3n-5$

The coefficients $c_{i_1,\ldots,i_k}$ in Eq. (5) are determined through integration over phasespace

$$c_{i_1,\ldots,i_k} = \int d\phi \, \Phi_{i_1,\ldots,i_k} f(\phi) \; . \tag{6}$$

Using this representation for the matrix we can write the total cross section given a set of cuts $C(\phi)$ as

$$\begin{aligned} \sigma &= \int d\phi_n \frac{d\sigma}{d\phi} C(\phi) \\ &= \sum_{i_1,\ldots,i_k} c_{i_1,\ldots,i_k} \int d\phi_n \Phi_{i_1,\ldots,i_k}(\phi) C(\phi) \\ &= \sum_{i_1,\ldots,i_k} c_{i_1,\ldots,i_k} d_{i_1,\ldots,i_k} \; . \end{aligned} \tag{7}$$

We see that we can separate the phase-space integration for matrix elements and for the cuts. In general we expect the the matrix element integration to be costly and the cuts integral is anticipated to be significantly easier. In Eq. (7) we decomposed the phasespace integral into a product of coefficients, each determined by a separate phasespace integration.

As an example we consider 3-jets production in e^+e^- collisions. For this example the centre of mass energy is fixed to 1 and we require 3 jets using $R = 0.1$ with a transverse momentum of 0.25 GeV.

First we consider the distributions $d\sigma/dx_i$. We can histogram in x_i while performing the phasespace integral (6) with f being the matrix element squared. The result is shown in Figure 3. We can then see if we can reconstruct the distribution from the coefficients $c_{i_1,\ldots,i_k}$. The result is shown in Figure 4. For this reconstruction we used a basis with 50 elements. We can see that the histograms are reconstructed quite well.

We now want to show that we can compute observables using our decomposition Eq. (7). We consider the leading jet transverse momentum histogram in the left panel of Figure 5. We

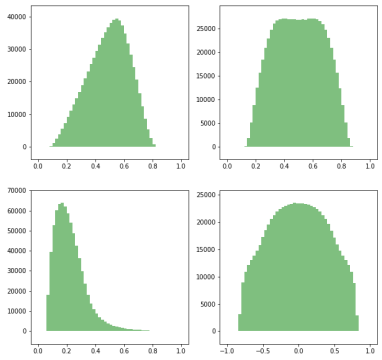

Figure 3. Marginal distributions for each coordinate.

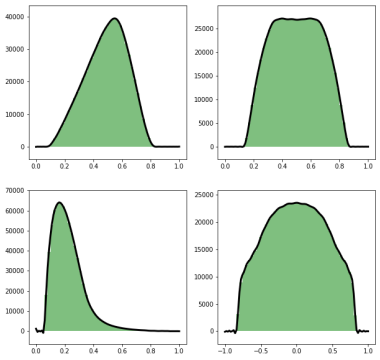

Figure 4. Reconstructed marginal distributions with 50 basis functions.

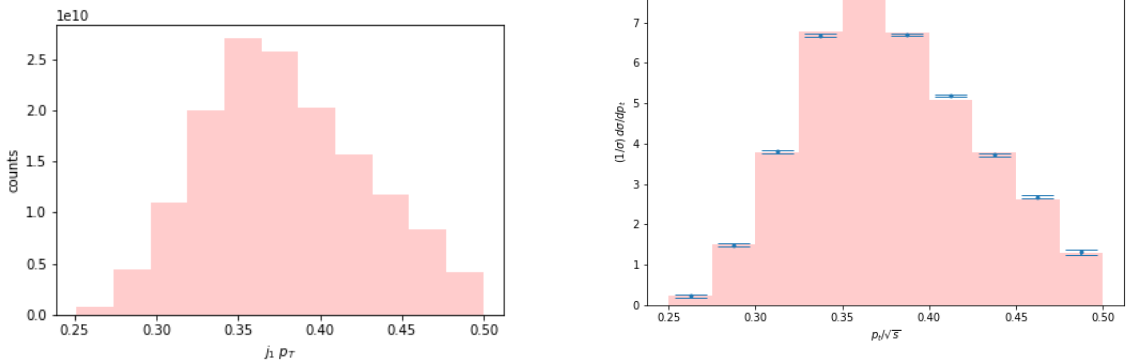

Figure 5. Left: Distribution of the first jet transverse momentum. Right: Distribution of the first jet transverse momentum reconstructed using Eq. (7).

compute the phase space integral Eq. (5) for each bin with f given by a cut function that is one on the support of the bin and zero otherwise.

Combining the result of the bins phasespace integral and the matrix element squared phasespace integral according to Eq. (7) we obtain the results shown on the right panel of Figure 5. The error bars in this figure are an estimate of the errors due to limited Monte-Carlo statistics in the determination of the matrix element and histogram bin coefficients and from the finite number of elements in the basis.

3. Generating unweighted samples

One advantage of having an approximation for the probability density is that we can draw samples from it using Gibbs sampling [6]. The algorithm is formulated as a Markov chain with the following steps:

- Start at a point X^i

$$X^i = (x_1^{(i)}, ...x_k^{(i)}) \tag{8}$$

- For each coordinate successively generate the next values for each coordinate x_j according to the conditional probability

$$x_j^{(i+1)} \simeq P(x_j|x_1^{(i+1)}, .., x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, ..., x_k^{(i)}) \tag{9}$$

- Once all coordinates are updated we have a new point to add to our sample.

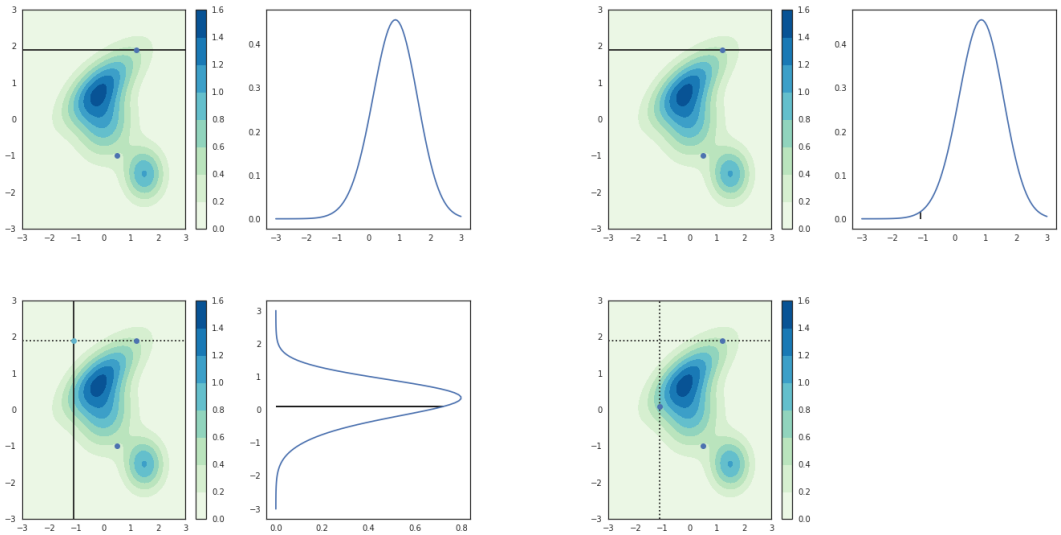

Figure 6. Illustration of the Gibbs sampling.

Figure 6 illustrates this process. We start at the point around $(1.8, 1.9)$. We build the conditional probability $P(x_1|x_2^{(i)})$ shown on the top right panel. We draw from that distribution the next value for x_1. We then calculate the conditional probability $P(x_2|x_1^{(i+1)})$, shown on the bottom right of Figure 6. We draw the next value of x_2 from that distribution.

Under normal circumstances it would be intractable to calculate the conditional probabilities in Eq. (9). The advantage of representing the matrix element in terms of basis function is that the conditional probability itself can be written in terms of the basis functions and its coefficients can be calculated through simple summation of the known coefficients of the full probability density:

$$P(x_j|x_1, ..., x_{j-1}, x_{j+1}, ..., x_k) = \sum_j d_j \Phi_j(x_j) \ , \tag{10}$$

with

$$d_j = \sum c_{i_1,...,i_{j-1},j,i_{j+1},...,i_k} \Phi_{i_1}(x_1) \cdots \Phi_{i_{j-1}}(x_{j-1}) \tag{11}$$

$$\times \Phi_{i_{j+1}}(x_{j+1}) \cdots \Phi_{i_k}(x_k) \ . \tag{12}$$

Once we have the conditional probability we need to to draw from it to obtain the next location. This is done by computing the cumulative distribution of the conditional probability and inverting it. The cumulative probability is easily obtained since integration in the space of the orthogonal polynomials is trivial: the integral of a polynomial is a polynomial so integration is a simple matrix multiplication.

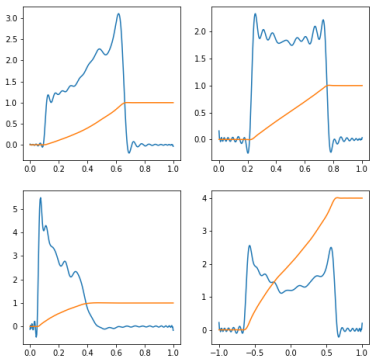

Figure 7. Sample conditional probabilities and cumulative distributions.

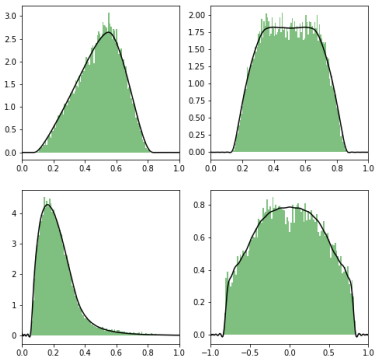

Figure 8. Unweighted sample constructed through Gibbs sampling.

Figure 7 shows examples of conditional probabilities and corresponding cumulative distributions for all coordinates at a sample point. The fluctuations visible in this figure are due to the limited statistics in the determination of the basis coefficient for the matrix elements and to the fact that the basis expansion has been truncated.

Figure 8 shows the same histograms as in Figure 3 for a sample of 20,000 events generated through the method outlined above.

4. Conclusion

In this contribution we discussed two ways of representing difficult matrix element calculations. The first is using event files and the second is to expand the matrix elements in a basis of orthogonal phasespace functions. The method of using orthogonal phasespace function is in the first stages of its development and much has to be done to prove its usability, including extending it to NLO or NNLO examples, using hadronic initial stages and fixing the numerical issues arising from the expansion truncation. One advantage of the method is that unweighted events can be generated using only linear algebra and would therefore be suitable for use on GPU accelerators.

References

[1] Maitre D, Heinrich G and Johnson M 2016 *PoS* **LL2016** 016 (*Preprint* `1607.06259`)
[2] Andersen J R *et al.* 2016 *9th Les Houches Workshop on Physics at TeV Colliders (PhysTeV 2015) Les Houches, France, June 1-19, 2015* (*Preprint* `1605.04692`) URL `http://lss.fnal.gov/archive/2016/conf/fermilab-conf-16-175-ppd-t.pdf`
[3] 2018 *Les Houches 2017: Physics at TeV Colliders Standard Model Working Group Report* (*Preprint* `1803.07977`) URL `http://lss.fnal.gov/archive/2018/conf/fermilab-conf-18-122-cd-t.pdf`
[4] Maître D 2018 *J. Phys. Conf. Ser.* **1085** 052017
[5] Bern Z, Dixon L, Febres Cordero F, Höche S, Ita H *et al.* 2014 *Comput.Phys.Commun.* **185** 1443–1460 (*Preprint* `1310.7439`)
[6] Geman S and Geman D 1984 *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-6** 721–741 ISSN 0162-8828