The YaJEM toolchain
— some background information

Thorsten Renk, 2014

1 Overview

Assuming the validity of factorized QCD in medium and for a general final state, computing the yield Y of any high P_T observable schematically amounts to solving an integral which can be written as

$$Y = \int d(\text{configurations}) d(\text{cuts}) f_A(x_1, Q^2) f_A(x_2, Q^2) \hat{\sigma}(\hat{s}, \hat{t}, \hat{u}) \\ D_{med}(P_{1_i}, \mu^2 | (\text{geometry}_1) D_{med}(P_{2_i}, \mu^2 | (\text{geometry}_2) \tag{1}$$

with $f_A(x_i, Q^2)$ the nuclear parton distributions probed at a momentum fraction x_i ad a scale Q^2, $\hat{\sigma}(\hat{s}, \hat{t}, \hat{u})$ the hard perturbatively calculable cross section in terms of partonic Mandelstam variables and $D_{med}(P_{i_j}, \mu^2 | \text{geometry}_i)$ the generalized medium-modified fragmentation function (MMFF) corresponding to an $n-$hadron distribution from a parton i with momenta P_{i_j}, given the particular geometry of the background QCD medium and its density evolution through which the parton shower evolves.

The medium geometry may here be specified using a series of transport coefficients and their spacetime dependence $\hat{T}(\zeta, \dots)$ where $\zeta = (\tau, x, y, \eta)$ and other factors, such as the relative aligment of fluid dynamical flow and parton direction may modify the transport coefficient in general.

The integral $\int d(\text{configurations})$ extends across all kinematically possible initial states and the integral $\int d(\text{cuts})$ across the experimental cuts relevant for the particular observable (this may involve clustering to jets, or particular trigger cuts, detector specific PID,...). Everywhere, δ-functions ensuring the conservation of conserved quantities are implied.

In particular, the configuration integral has the following pieces:

- The nuclear geometry part $\int dx_0 dy_0 d\phi d\psi$, integrating over the location of the hard vertex (x_0, y_0) of a binary N-N collision in the transverse plane for given impact parameter and incoming nucleon configuration, also selecting the transverse propagation direction ϕ with respect to the bulk reaction plane and the degree of away side acoplanarity ψ of the away side parton.
- The initial state parton configuration part $\int dx_1 dx_2 \sum_{a,b}$ integrating over all possible fractional momenta x and summing over parton types a, b
- The time evolution integral $d\tau$ integrating the medium along the spacetime path of partons propagating through it from the selected vertex

Similarly, the final state integral can schematically be cast into the form $\sum_{n=1}^{\infty} \prod_{i=0}^{n} \int d^3 P_i$ and accounts for the possibility of producing an n-hadron final state with momenta P_1 to P_n.

Computing an observable assuming the validity of factorized QCD for the MMFF thus boils down to 1) computing the hard process and the MMFF 2) solving all the configuration averaging and cut integrals.

2 Event generators as MC integration

Event generators address the problem by Monte Carlo (MC)-simulating a complete event from beginning to end, i.e. they generate a distribution of nucleons

in the incoming nuclei, randomly pick a hard vertex direction and acoplanarity, then select the hard scale and parton types, propagate the so generated partons through the medium while probabilistically building parton showers, hadronize and finally analyze the resulting distributions whether they fulfill a trigger cut or contribute to an observable yield. In the following, this will be referred to as 'forward event generation'.

While this is the most intuitive way to solve the integral above, it is, when seen from a numerical mathematics perspective, also one of the worst ways, as it technically corresponds to doing the multi-dimensional integral

$$\int dx_0 dy_0 d\phi d\psi \int dx_1 dx_2 \sum_{a,b} d\tau \sum_{n=1}^{\infty} \prod_{i=0}^{n} \int d^3 P_i$$

using a MC integration algorithm in one go. The numerical disadvantage of such a procedure is for instance apparent from the fact that until the full final hadron distribution has been created and analyzed, there is no criterion to discard an event which is not going to contribute to an observable early without investing more effort.

3 General properties of numerical integrators

More generally, a MC integrator is usually not the optimal tool to do a multi-dimensional integral. For instance, if the integrand is strongly peaked in some part of the integration region (as for instance parton momenta peak at low p_T), MC integators inevitably oversample this part and need to run a large number of events to get a reliable estimate of other regions in the integrand. Compensating this effect is the reason why e.g. PYTHIA allows to specify a minimal hard scale, so that by sliding the hard scale, equal statistics in all regions of phase space can be enforced by hand. Different integration algorithms, such as adaptive subdivision, do not suffer from this problem the same way and give a straightforward computation of the integral without manual intervention.

Integrals of the form $\int dxdyf(x)$ take the same effort as $\int dxdyf(x,y)$ for a MC integrator which has no way of realizing that one dimension is a trivial integral. This is not true for an adaptive subdivision routine which does get wise to the fact after a while (but the best solution is of course to do the trivial integral analytically before the numerical integration).

Finally, in dealing with an integral of the type $\int dxdyf(x)g(y)$, it is numerically orders of magnitude faster to factorize into $\int dxf(x) \cdot \int dyg(y)$ than to try to do it directly.

Many of the integrals indicated above however are of the latter type. For instance, the whole initial state parton configuration integral can be done independently of the geometry. This indicates that discarding forward event generation, a much faster solution to the required integral can be obtained. This idea, to use the numerical integration technique which is optimally adapted to the problem in question is at the core of the YaJEM toolchain. YaJEM is hence **not** an event generator, but a tool do one particular numerical integration as part of a larger numerical computation problem.

4 The toolchain

The YaJEM toolchain as described below is a series of steps to compute hard observables which is agressively optimized for performance (rather than ease of use). This toolchain is the way of using YaJEM that is most extensively tested and practically all published YaJEM results are based on a variant of it.

The toolchain consists of various pre-computation stages which tabularize certain quantities, which are in the end brought together by an event loop stage.

4.1 The hard QCD event

Currently YaJEM is merged with a leading order (LO) pQCD computation to give the yield of high P_T probes. Using an NLO computation for the parton spectra is conceptually not straightforward because the numerical generation of a final state shower resums part of the NLO dynamics, and hence double-counting becomes an issue. If matching with an NLO computation is desired, a scheme analoguously to what POWHEG does for p-p collisions needs to be implemented.

For numerical purposes, the parton p_T spectrum is most conveniently pre-computed for given $\sqrt{s}$ in the desired rapidity range using an adaptive subdivision integrator. This can be done, as the hard event is scale-separated from any medium dynamics. For a specified trigger P_T, the parton p_T spectrum can then be loaded above the trigger energy only, integrated up to a chosen limit, normalized and stored in integral version such that a single random number produces an event p_T scale from a lookup table. The partonic composition at given p_T can be stored similarly such that a second number gives parton type (quark or gluon) from another lookup table, and two more random numbers can be used to fetch away side parton type (conditional on near side parton type) and, if so desired, an acoplanarity (random p_T imbalance of the event mimicking NLO effects).

At the end of this stage, YaJEM has used four random numbers to create a back-to-back partonic event in set rapidity bins with the parton ID known. This incurs almost no numerical cost.

4.2 The medium averaging

Due to a scaling law identified in [1] which has been extensively tested since, despite taking nominally a full profile of a transport coefficient probed along the parton path $\hat{T}(\zeta)$ as input, observables are, for the set of path that can realistically occur in a hydrodynamical medium, to a very good approximation sensitive to $\Delta Q^2 = \int d\zeta \hat{T}(\zeta)$ and $L = \int d\zeta$ only. In all tests, this appears to be a general (i.e. **observable independent**) scaling law. Note that ΔQ^2 is in essence the value of `YAPARS(2)` whereas L enters indirectly via the value of `PARJ(82)`.

In addition, there is often an **observable-dependent** scaling law which allows to leave the lower virtuality evolution limit `PARJ(82)` unchanged if instead the replacement $\Delta Q^2 \to \Delta Q^2_{eff} = \Delta Q^2 + f(L)$ is performed where $f(L)$ is an observable-specific function (leading hadron distributions for fixed parent parton energy for instance show strong L dependence, jet energy distributions do not).

These two scaling laws allow the following observable-dependent treatment of the medium:

- If only a single branch of the back-to-back event is observed (such as for hadron or R_{AA} or jet structure analyses), the whole medium averaging can be precomputed by generating vertices (or taking binary collision points from an EbyE hydrodynamics), generating a random propagation direction (across all angles or within certain orientations with the reaction plane or event plane), compute ΔQ^2 and L along this ray and store the resulting pair $(\Delta Q^2, L)$ in a table. A central event has typically $O(1000)$ binary vertices, a decent angular resolution can be achieved with about 20 different orientations, hence the complete information of a hydrodynamical event as seen by YaJEM can be reduced to a table of about 20.000 numbers. If the second scaling law can be utilized, the table can be reduced to 10.000 numbers of ΔQ^2_{eff}. In this case, the whole medium averaging inside a computation reduces to reading the next pair $(\Delta Q^2, L)$ (or next number ΔQ^2_{eff}) from the table every time the event loop is executed. This is a procedure with negligible numerical cost.

- If both branches of the back-to-back event are relevant, the presence of an acoplanarity prevents complete factorization (which is however restored at sufficiently high P_T when partons become increasingly back-to-back). In this case, the computation of ΔQ^2 and L needs to be inside the event loop (rather than simply read from a pre-computed table) and has considerable numerical cost — which means this should only be done as needed.

At the end of this stage, YaJEM generates the two parameters ΔQ^2 and L for each of the two back-to-back partons which determine how the generalized fragmentation function will be distorted by the medium.

4.3 The final state

Currently, most experimental observable definitions are so complex that their accurate computation requires MC tools. For instance, analytically evaluating the energy falling into a charged anti-k_T jet with $R = 0.3$ which needs to contain a track of at least 4 GeV is close to impossible. Using MC instead, such observables are readily computed.

However, once known, the distributions (in the above example the probability distribution to recover within the cuts a certain energy $P(E)$) are, for given parton type, smooth functions of the initial parton energy and the medium characterizing parameters $(\Delta Q^2, L)$ and can be tabulated and interpolated well.

One key idea of the YaJEM toolchain is to medium-average these observable-specific distributions over the medium rather than event records. This works equally well, as there is no interaction across different events.

Effectively, this step requires to pre-compute observable specific tables of (in the above example) $P(E|E_{part}, \Delta Q^2, L)$ (or $P(E|E_{part}, \Delta Q^2_{eff})$) for quarks and gluons separately as a function of the initial parton energy E_{part} and the medium-specific parameters. Since the distributions are smooth and interpolate well, for a typical LHC computation across a larger P_T range $O(60)$ tables are needed, and the generation of these tables is the only place in the toolchain where the code YaJEM is used. Pre-tabulating tables as a function of E_{part} has

the obvious advantage that the statistical accuracy is user-controlled and easily made even across the whole phase space. In addition, it offers an even more significant advantage which is discussed below.

4.4 Bringing it together

The event loop part of the toolchain starts, as discussed above, with generating a back-to-back parton event. In the case of triggered correlation observables, it would then naively proceed to compute the medium parameters for both legs, and then compute observables from an interpolated $P(E|E_{part}, \Delta Q^2_{eff})$. This however is very uneconomical, as a) the medium averaging procedure is numerically much more expensive than using a lookup table and b) most partonic events do not trigger even without a medium.

For a hadron trigger condition, typially $O(1000)$ partonic events need to be generated per triggered event in vacuum (and $O(4000)$ in medium). This implies that only for 0.1% of events it is actually necessary to generate the full event including the expensive medium modification, while 99.9 % of the time the work is done in vain — if we could only just know how the event would behave in the absence of a medium.

But in fact, averaging over $P(E)$ rather than event records, we do — all that needs to be done is test against $P(E|E_{part}, 0)$ before the medium computation and only if the event passes the test, compute the medium parameters and test against the full $P(E|E_part, \Delta Q^2_{eff})$. This would not work on the level of averaging event records and corresponds to a vast performance boost, as the most expensive part of the computation is now only done when needed.

The full sequence executed in the event loop part of a toolchain for a triggered correlation observable is hence:

- Generate a partonic back-to-back events from lookup tables (almost zero numerical cost)
- Check whether this event would fulfill the trigger condition if there were no medium using final state lookup tables (low numerical cost), if not discard and loop over if not
- Compute the medium parameters for the given partonic event for the near-side (trigger) leg (high numerical cost)
- Check whether this event fulfills the trigger condition using interpolated final state lookup tables (low numerical cost), if not discard and loop over
- Compute the medium effect for the away side leg (high numerical cost)
- Compute the observable using interpolated lookup tables based on parton parameters and medium parameters (low numerical cost)
- Loop to the next event

5 Performance comparisons

While the toolchain described above is counter-intuitive and complicated, the performance gains are substantial. The first back-to-back hadron correlation

computation published in [2] was done in a forward event generation scheme, taking 30.000 CPU hours on the NERSC supercomputing facility. Using the YaJEM toolchain, the same computation can be done in 20 minutes on a single CPU with the same statistics if the lookup tables are precomputed — a factor of almost 100.000 in performance gain. Even pre-computing the lookup tables is a matter of less than 20 CPU hours — forward event generation schemes are just staggeringly wasteful.

Triggered 2+1 coincidences where the trigger requires hard hadrons on near and away side are alomst impossible to obtain in forward event generation — here the expected performance gain of the YaJEM toolchain is a factor or 10.000.000.

The performance gain is somewhat less in the case of jet observables. For comparison here, the MARTINI computation of the dijet imbalance took (according to S. Jeon) about a month of supercomputing time. Using the YaJEM toolchain, including computing the final state lookup tables with decent statistics, the computation can be done in a day, the performance advantage in this case is hence still about a factor 1000.

Perhaps these numbers make it clear why forward event generation, despite being intuitive and easy, is from a numerical point of view a very bad idea and why the YaJEM toolchain is different.

References

[1] T. Renk, Phys. Rev. C **78** (2008) 034908.

[2] T. Renk and K. Eskola, Phys. Rev. C **75** (2007) 054910