Universiteit Antwerpen
Faculteit Wetenschappen
Departement Fysica
Academiejaar 2010-2011

Determination of PDF4MC parton densities from HERA data using the Pythia Monte Carlo generator

Proefschrift ingediend met het oog op het behalen van de graad
Master in de Fysica

Author: Sara Alderweireldt
Promotor: Dr. H. Jung
Copromotor: Prof. Dr. P. Van Mechelen

27th May 2011

Preface

In the first place I'd like to thank my promotor and my copromotor, Hannes Jung and Pierre Van Mechelen, for providing this subject and making it possible for me to work on it in Antwerp. To both of them I'm also grateful, next to the expected guidance, for the many problem solving suggestions and tips on good references, as well as their willingness to adapt to my unusual schedule of theory courses and my tendency for taking a while to get going on things. On a different note I send a big thank you to M., for not only listening whenever I wanted to vent, but also understanding what I meant even if I could not get it said, and for believing in me at times when I did not do so myself. A similar feeling goes out to S., who's been there for me since long before I started this work and put up with me at times when I did not have much attention for anything but myself and my own work. Next I have to thank my family, for always being there and not complaining if my work kept my focus away from them and at times put me into a grumpy mood. Finally I'd like to thank B. and D., for providing moments of enjoyment in busy times, entirely unbeknownst to themselves, and my computer, for happily calculating at my command for many, many hours. Each of these people had their own uniquely different contribution, and I would have had a much harder time completing this work if I'd had to do without any of them.

Again, many thanks!

Sara Alderweireldt
27th May 2011

Abstract

The content of this work on parton density functions is dual. After a general theoretical introduction (Part I), the first main part (Part II) focuses on obtaining integrated and unintegrated parton density functions and parton luminosity functions from several different sources and comparing them. We work with results from Monte Carlo event generator Pythia 8.1, the LHAPDF interface, the Kimber-Martin-Ryskin approach and a CCFM dataset. In the second main part (Part III), the focus moves to Deep Inelastic Scattering events with the HERA combined results as reference data. Because of limitations in Pythia 8.1 this requires the use of Pythia 6.4. The idea is to choose a parton density function parametrization and to optimize it for Pythia by tuning to reference data, a method sometimes called Pdf4mc. We prepare the reference data manually and write a Rivet analysis to extract similar information from Monte Carlo generated data. Furthermore we study the PDF parametrization formats understood by the LHAPDF interface. Finally we look at generated data for one specific PDFset, taking a first tiny step in a tuning procedure by looking at the effect of varying one parameter and tuning it with the Professor package.

Contents

Part I - Introduction

The first part of this thesis aims to provide a background on the theory and tools we used. We start with some general particle physics theory (see section 1), take a look at particle accelerators (2), discuss simulation with event generators (3) and end with a description of the software tools (4).

1 Physics

In the following section we attempt to give a structured overview of the theoretical background needed to understand the work done for this thesis. We do not aim to give a full representation of any of the subjects, this would lead much too far, but instead focus on the elements which are really relevant for us. For more detailed descriptions and further references we refer the reader to [1], [4], [6] and [8].

1.1 Quantum Chromodynamics

Up to a certain level, the fundamental aspects of nature are described by the standard model. It incorporates quantum electroweak theory, describing the electromagnetic- and the weak fundamental interaction, as well as quantum chromodynamics, describing the strong fundamental interaction. Even though it is a very important general theory unifying many smaller components and capable of many valid explanations, it cannot be considered as a unified field theory, a theory of everything. For example gravitational interaction, quite obvious in daily life, and dark matter and dark energy are not included in the standard model. Next to these incompletions, there are also multiple unexplained elements such as the hierarchy problem or neutrino oscillations, arising within the theory. All this, however, does not mean that the standard model cannot be used as a valid general theory. One should only take care to stay within its limits.

In this work we will be dealing with the strong interaction and therefore consider quantum chromodynamics (QCD) in particular. The strong force is responsible for interactions between quarks and gluons, also called partons. When observed, these partons are always contained in colour neutral hadrons; either baryons (three quarks) or mesons (two quarks). Colour charge is one of the fundamentals of the theory when looked at from the mathematical side of things, considering colour gauge group $SU(3)_c$. However, it is unnecessary for the purpose of this thesis to fully illustrate the mathematical framework of the theory. Instead we wil concentrate on some general properties and more specifically the structure of the proton, which is affected by the strong interaction and is one of the things studied in the past with the Hadron-Electron Ring Accelerator at DESY, Hamburg and today with the Large Hadron Collider at CERN, Geneva.

1.1.1 Confinement and asymptotic freedom

Looked at from the physics side of things, two fundamental points of QCD are confinement and asymptotic freedom. The former is needed to explain why we always see hadrons and never individual partons propagating through the macroscopic world. The latter allows the perturbative treatment of QCD at high energies, explaining the running strong coupling constant α_s.

In quantum electrodynamics (QED), the coupling constant is $\alpha \approx \frac{1}{137}$, also known as the

fine structure constant. Thanks to its smallness, higher orders in a perturbative treatment quickly become unimportant. This is the basis of Feynman diagrams and calculations. Calculated in the same way though, the strong coupling constant α_s would be bigger than one, creating a big problem in the light of perturbative calculations. Higher orders, or more complex Feynman diagrams, would become increasingly more important, rendering the whole technique entirely pointless.

Luckily, it turns out that α_s is not constant, but dependent on the separation of interacting particles. At large distances it is large, while at short distances it is small. This is what is called asymptotic freedom. At short distances, say within their hadron, partons are free and can move around. Looking at small energy scales (large distance scales), this effect would go unnoticed. Only when going further, looking at higher energy scales and performing (deep) inelastic scattering experiments, the 'free' partons become visible. The decreasing of α_s towards smaller distances means that at high energy scales, a perturbative treatment is perfectly acceptable.

For an explanation of the asymptotic freedom we have to go a bit further. Considering QED, there is the effect of charge screening. This is entirely equivalent to the macroscopic charge screening in electrodynamics. There is screening, until at a distance closer than the nearest neighbour, the effect falls away. The QED equivalent of the nearest neighbour is the electron Compton wavelength λ_c. This means that for the (Q)ED case, the effective charge, and thus the coupling constant α, increases towards smaller distances. Now considering QCD, there is this same effect, but the possibility of gluon self-interaction and thus gluon-loops adds a second effect working in the opposite direction. Mathematically, playing with the number of quarks and gluons in the theory, the net effect could go either way. For the standard model situation, the gluons win and the effect for QCD is opposite to the effect for QED. The strong coupling constant α_s decreases towards smaller distances.

Lastly we should consider confinement. Again the gluons are responsible for the effect. It is the gluon that carries the strong force, like the photon carries the electromagnetic force. The difference is that the photon is charge neutral, while the gluon is colour charged. When two electrons, a QED situation, separate, the electric field between them disappears. Nothing remains to keep them together. In a QCD situation though, when two quarks separate, there is still the colour field of the gluons to take into account. A 'string' of colour charge still links them together, even as they separate. They are confined. However, continuing to move apart, at some point a new quark-antiquark pair will be created inbetween and there will be two sets of quarks linked by a colour string. This is the basis of hadronization or fragmentation, the fact that in particle detectors bunches of many colour neutral hadrons are observed and never colour charged particles on their own. Confinement remains a rather mystical element of QCD. That the effect concerns large distance scales, the scales where perturbative treatment is problematic, does not help clear things up.

1.2 Proton structure

1.2.1 Scattering experiments

Scattering experiments are an important tool in the study of the structure of matter. An important factor thereby is the energy scale at which the matter is looked at. As in ordinary microscopy, the detail seen can never be greater than the size of the probe. Looking at for example the proton at low enough energy, it might look pointlike, but we know it is not. It consists of partons, which only become visible if the energy scale is increased and thus

the probe size becomes smaller. Originally, people looked at elastic scattering. Particles with known properties, e.g. their momentum, are set to collide and from studying the cross sections, information can be obtained about their structure and the interaction. But elastic scattering can only go so far. Higher energies allow for the studied proton to break up, forcing us to deal with inelastic scattering. Inelastic structure functions are introduced for the proton. All this renders the description of the experiment, including cross sections (1.2.2) and structure functions (1.2.3), a lot more complicated.

To study the proton, one can for example perform Deep Inelastic Scattering (DIS, figure 1) experiments where electron-proton collisions are realized and the electron probes the proton to a degree where quarks have become 'visible'. The electron then scatters off one of those quarks, which is a pointlike particle. We learn that the proton consists of three valence quarks and many sea quarks, more of which are resolved with increasing energy scale Q^2. This idea is contained in the Quark Parton Model (QPM, 1.2.4). Also introduced at this point are the parton density functions (PDFs, 1.2.6). In short they describe the distribution of partons carrying a fraction x of the proton momentum. It is clear that at higher energy scale Q^2, the observed x-values will be smaller, since the available proton momentum has to be distributed over more resolved partons. Going further, with the introduction of QCD, gluons needed to be added to the picture. They fit in the proton together with the sea quarks. Following QCD also means that parton radiation has to be included, which will lead to scaling violations (1.2.5). When looking at the proton structure function for fixed x but variable energy scale Q^2, behaviour will vary. This is visible in figure 2.

Due to radiative effects, acceleration of charged particles is limited. For the continued movement towards higher energy scales Q^2 this means at some point electrons are no longer an option. Luckily, the limit for protons lies higher, still allowing improvement. Electron-proton collisions were studied at HERA between 1992 and 2007 (2.1). Further steps are now taken in the study of the proton structure with the proton-proton scattering experiments at LHC which started in 2009 (2.2).

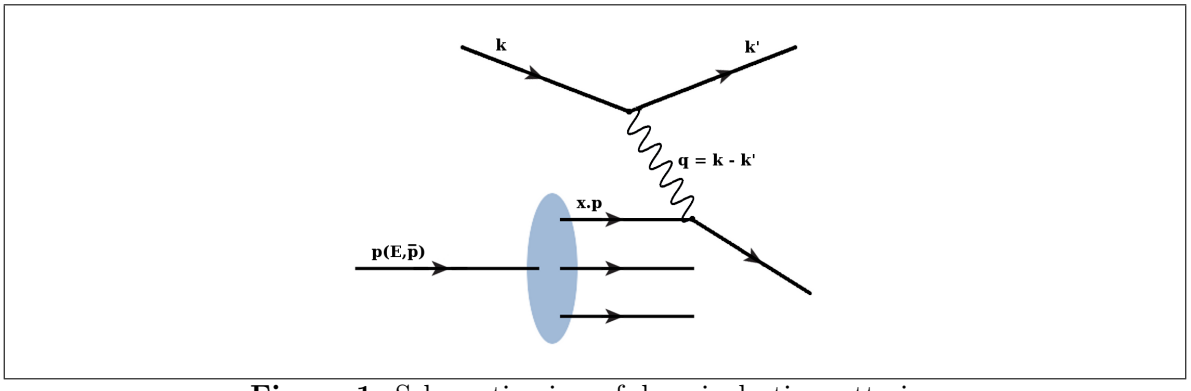


Figure 1: Schematic view of deep inelastic scattering

1.2.2 Cross sections

Classically, if one looks at scattering of a pointlike object on a solid target, the likelihood of interaction happening is proportional to the ratio of the area of the target (the cross section) to the total area. However, when dealing with beams as a target, the target is not solid, we call it soft. The question is not 'will it be a hit or a miss', but rather 'how great is the interaction at some distance'. Also, not only the target is important, but also the projectile.

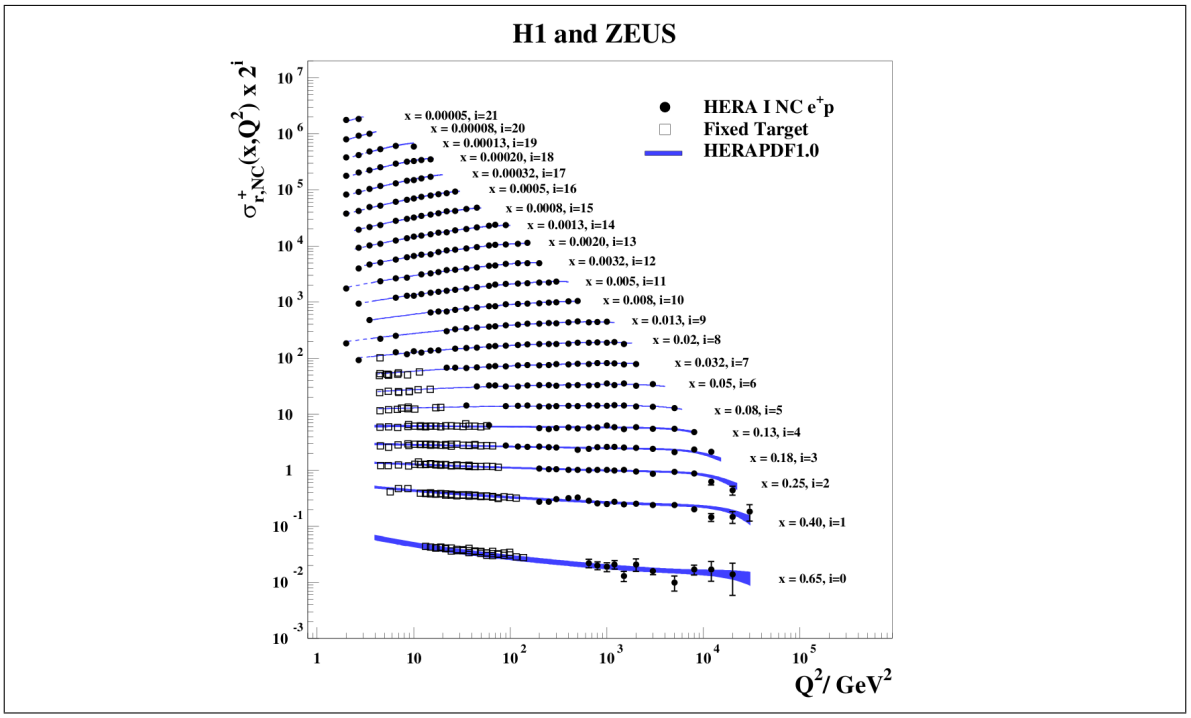


Figure 2: HERA combined NC e^+p reduced cross section and fixed-target data as a function of Q^2 with the HERAPDF1.0 fit superimposed, reproduced from [7]

Particles with different properties (massive/massless, charged/neutral, ...) may interact differently. Finally there is inelastic scattering to deal with, where the outgoing particles differ from the ingoing. As mentioned before, at high enough energy ingoing particles may break up before the actual scattering, or new particles may be created. It no longer suffices to look at the changes in the ingoing particles, all side products and processes need to be considered too. Luckily, each subprocess will have its own 'exclusive' cross section σ_i which can be studied. Together all these exclusive cross section will form the total or 'inclusive' cross section:

$$\sigma_{tot} = \sum_{i=1}^{n} \sigma_i$$

We will not go into detail about how different types of cross sections can be calculated, for more details and references we refer to [5], [6] and [8].

Another important term we have to discuss when talking about scattering experiments with particle accelerators is luminosity L. Often used is the integrated luminosity, the integral of the instantaneous luminosity over time:

$$\mathcal{L} = \int L dt$$

Luminosity can be seen as a flux of particles per unit area per unit time. If we consider a beam of particles, the luminosity would be the number of particles dN passing through an area $d\sigma$ per unit time, or when scattering off a potential for example, the number of particles dN scattered into a solid angle $d\Omega$ per unit time:

$$\mathcal{L} = \frac{dN}{d\sigma} \qquad \text{or} \qquad \mathcal{L} = \frac{1}{D(\theta)} \frac{dN}{d\Omega}$$

From that we can define the differential cross section, the number of particles scattered into a solid angle, divided by the luminosity:

$$\frac{d\sigma}{d\Omega} = \frac{1}{L}\frac{dN}{d\Omega}$$

Finally we consider the event rate as seen experimentally:

$$\frac{dN}{dt} = L \cdot \sigma \qquad \text{or} \qquad N = \mathcal{L} \cdot \sigma$$

1.2.3 Structure functions

When studying deep inelastic scattering one needs structure functions, functions parameterizing the structure of the target as seen by the probe. We consider electron-proton scattering. The scattering is mediated by the exchange of a virtual photon. There is a momentum transfer of $q^\mu = k^\mu - k'^\mu$, where the k's are the incoming- and outgoing lepton four-momentum. We can also define the first of several DIS standard variables: the momentum transfer or scale Q^2, which will determine the amount of proton structure resolved by the photon.

$$Q^2 = -q^2$$

In the proton rest frame it can be stated that, with proton four-momentum $p^\mu(E_p;\ \vec{p} = 0)$ and photon four-momentum $q^\mu(E_q = E_e - E'_e;\ \vec{q} = \vec{k} - \vec{k'})$:

$$\begin{aligned} M^2 &= p^2 \\ \nu &= p \cdot q = E_p \cdot E_q - \vec{p} \cdot \vec{q} = M \cdot (E_e - E'_e) \end{aligned}$$

There are only two independent variables, but we can also define dimensionless variants:

$$\begin{aligned} x &= \frac{Q^2}{2\ p \cdot q} = \frac{Q^2}{2\nu} = \frac{Q^2}{2M \cdot (E_e - E'_e)} \\ y &= \frac{p \cdot q}{p \cdot k} = \frac{Q^2}{2x\ p \cdot k} = \frac{E_e - E'_e}{E_e} \end{aligned}$$

Finally we note the invariant mass of the outgoing system W:

$$W^2 = (p + q)^2$$

The labels deep and inelastic attach some conditions to these variables. Deep means an energy scale much larger than the proton mass: $Q^2 \gg M^2$, while inelastic means invariant mass much larger than the proton mass: $W^2 \gg M^2$.

Next we look for observables linked to the process. These would be the structure functions mentioned previously, function of variables x and Q^2:

$$F_i(x, Q^2)$$

The structure functions are defined in function of the lepton cross section. It goes beyond the scope of this document to provide details about the derivation of this link between structure functions and cross sections, including the study of vertices and matching tensor equations. We show the following equation to give a general idea of what kind of functions we are dealing with.

$$\frac{d^2\sigma_{ep}}{dxdQ^2} = \frac{4\pi\alpha^2}{Q^4}\left[\left(1 + (1-y)^2\right) F_1 + \frac{(1-y)}{x}\left(F_2 - 2xF_1\right)\right] \tag{1}$$

1.2.4 The quark parton model

The quark parton model (QPM) draws a now known to be simplified picture of DIS. In the same infinite momentum frame as mentioned previously, time dilation ensures that, in comparison with the very fast moving proton, the interaction rate of the quarks will be very small. They are essentially free during the time range in which the interaction of the virtual photon with one of the quarks happens. This 'freedom' means that the total interaction, which is inelastic, can be approached as an incoherent sum of pointlike elastic interactions, the scattering from the free quarks. Each electron-quark scattering has a defined cross section, and each quark has a probability of being present in the proton and carrying momentum fraction ξ (PDFs, 1.2.6). Combining these properties we can write down the incoherent sum to find the cross section of the electron-proton interaction:

$$\frac{d^2\sigma_{ep}}{dxdQ^2} = \sum_q \int_0^1 d\xi \; f_q(\xi) \left(\frac{d^2\hat{\sigma}_{eq}}{dxdQ^2} \right)$$

The eq-cross section can be calculated using matrix elements. We will not go into details about this derivation, but we show the result (looking similar to the previously given expression for the ep-cross section ((1), page 5)):

$$\frac{d^2\hat{\sigma}_{eq}}{dxdQ^2} = \frac{4\pi\alpha^2}{Q^4} \left(1 + (1-y)^2\right) \cdot \frac{1}{2} e_q^2 \delta(x - \xi)$$

If we continue the comparison with the ep-cross section, which we wrote down with proton structure functions $F_i(x)$ in equation (1), we can extract the quark structure functions $\hat{F}_i(x)$ from the expression for the eq-cross section:

$$\hat{F}_2(\xi) = 2x \; \hat{F}_1(\xi) = x e_q^2 \delta(x - \xi)$$

The proton structure will not be a delta-function like that, but a distribution in x. The quarks in the proton don't always carry the same momentum fraction. Here enters again the PDF we mentioned before: the probability distribution $f_q(\xi)$ for a quark to carry momentum fraction ξ inside the proton. In precisely the same way the ep-cross section follows from the eq-cross section, the proton structure function can be found by weighting the quark structure function with the quark PDF:

$$F_2(x) = 2x \; F_1(x) = \sum_q \int_0^1 d\xi \; f_q(\xi) \;\; x \; e_q^2 \delta(x - \xi) = \sum_q e_q^2 \cdot x \; f_q(x)$$

Finally we need to show where the gluon comes in and this 'simple' QPM needs to be improved. It is a rather straightforward step from the sum in the proton structure function $F_2(x)$, which (with $f_q = q = q_{val} + \bar{q}$) can also be written as:

$$F_2(x) = x \cdot \left[\frac{4}{9}u + \frac{1}{3}d + \frac{1}{3}s + \ldots + \frac{4}{9}\bar{u} + \frac{1}{3}\bar{d} + \frac{1}{3}\bar{s} + \ldots \right]$$

to some interesting tricks where one considers the known proton quantum numbers. One looks at sum rules. Summing over partons in certain combinations has to result in correct values for matching properties of the proton. Examples are the charge (+1), and the

strangeness (0). Both can be retrieved by looking at flavour sum rules for the u-, d- and s-quark:

$$\int_0^1 (u(x) - \bar{u}(x))\, dx = \int_0^1 u_{val}(x)dx = 2$$
$$\int_0^1 \left(d(x) - \bar{d}(x)\right) dx = \int_0^1 d_{val}(x)dx = 1$$
$$\int_0^1 (s(x) - \bar{s}(x))\, dx = \int_0^1 s_{val}(x)dx = 0$$

The third one directly shows the strangeness to be zero: no strange valence quarks. The first and the second one show the known uud composition of the proton, which leads to the proton charge:

$$2 \cdot e_u + 1 \cdot e_d = 2 \cdot \left(\frac{2}{3}\right) + 1 \cdot \left(-\frac{1}{3}\right) = 1$$

This does not yet provide any indication of a gluon though. It is only by looking at another sumrule, experimentally, considering proton momentum, that it becomes clear something is missing if only quarks are considered. Extracting from experiments the momentum sum over all quarks in the proton (with $q_{sea} = q + \bar{q}$):

$$\int_0^1 x \cdot (u_{sea} + d_{sea} + s_{sea})\, dx \approx 0.5$$

indicates that something other than the valence- and sea-quarks is carrying nearly half of the proton momentum. This would be the gluons. While in charge sums the presence of gluons went unnoticed (as they are neutral), looking at the momentum balance showed us another element was necessary.

In chapter nine of *Quarks and Leptons* [8], a nice schematic overview of the possible views of proton structure is printed, including the repercussions for the proton structure function. We reproduce it in figure 3.

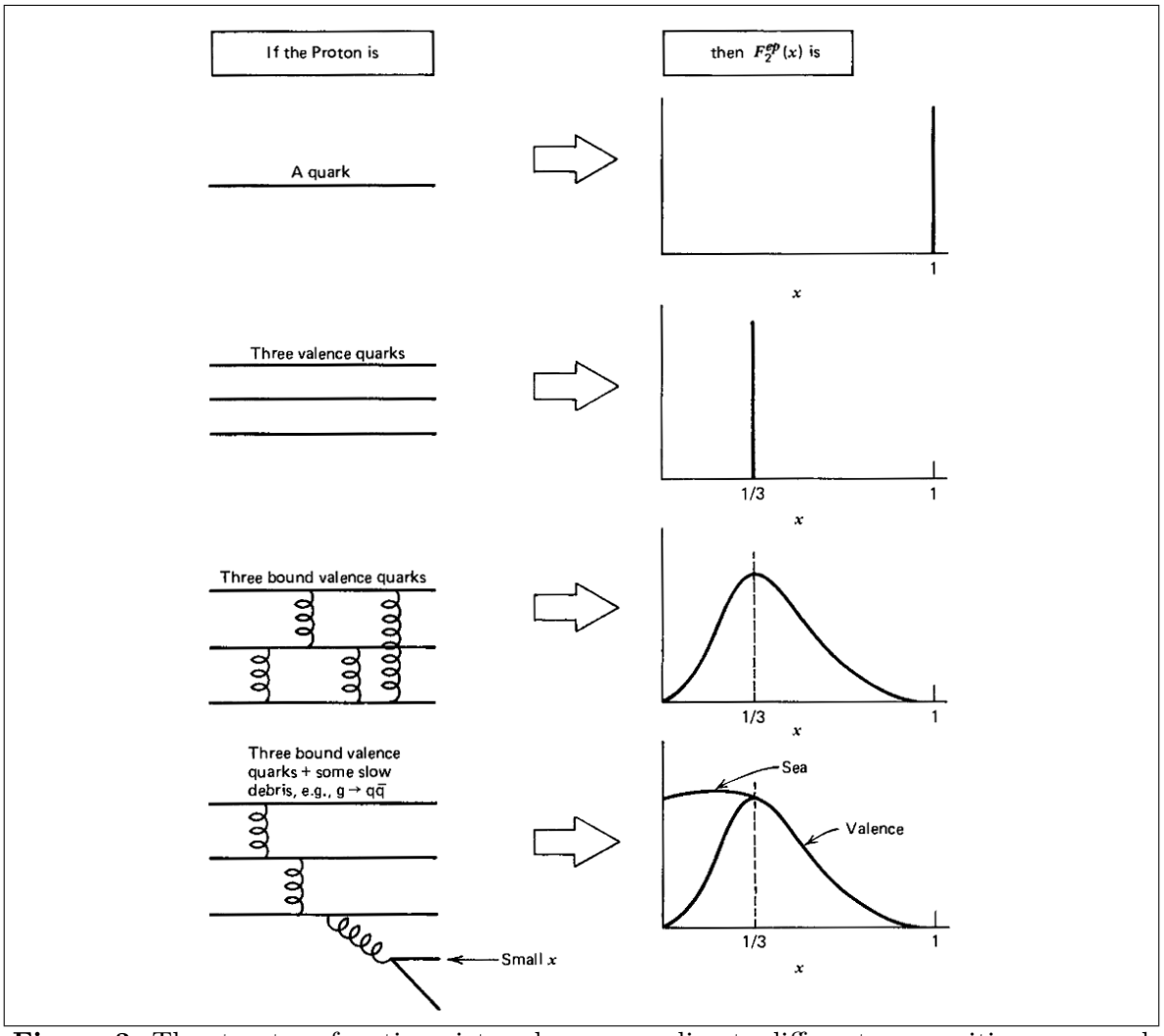


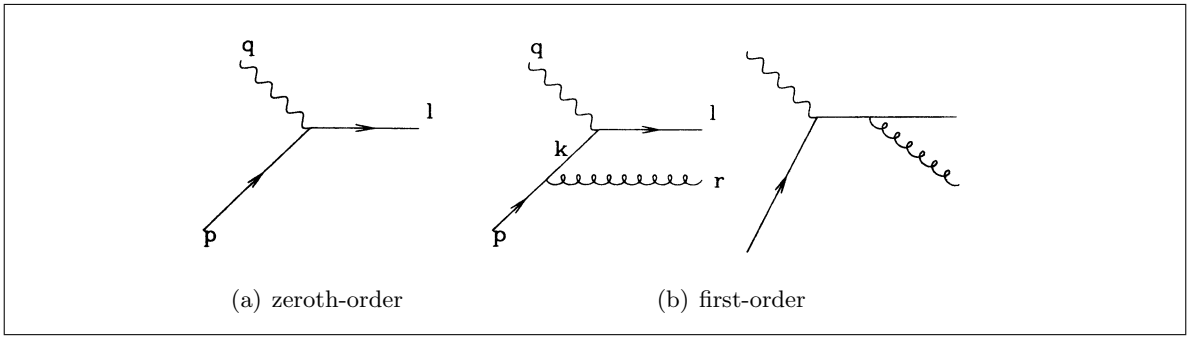
Figure 3: The structure function pictured corresponding to different compositions assumed for the proton (reproduced from [8])

1.2.5 Scale violations and factorization

We started the section about the QPM on the note that it was incomplete and we mentioned before that parton radiation is one of the elements of QCD that needs to be added to the model to improve it. In the context of the QPM we discussed the proton structure function independent of scale Q^2, while experiments indicate scaling violation in that same scale Q^2 (remember figure 2). QCD theory includes this scaling violation. The partons considered in the QPM can for example radiate a gluon, thus acquiring a transverse momentum (where before they had none) and changing the whole picture. We have to study possible QCD contributions if we want to describe the proton structure correctly.

In the QPM we had $\gamma^* q \rightarrow q$, the scattering of a virtual photon off a quark. This is the zeroth order diagram, to which we want to make a first order correction. We show the diagrams in figure 4. The matching quark structure function at zeroth order was:

$$\hat{F}_2(\xi) = x e_q^2 \delta(x - \xi)$$



(a) zeroth-order (b) first-order

Figure 4: Zeroth- and first-order Feynman diagrams for $\gamma^* q \rightarrow q$, reproduced from [4]

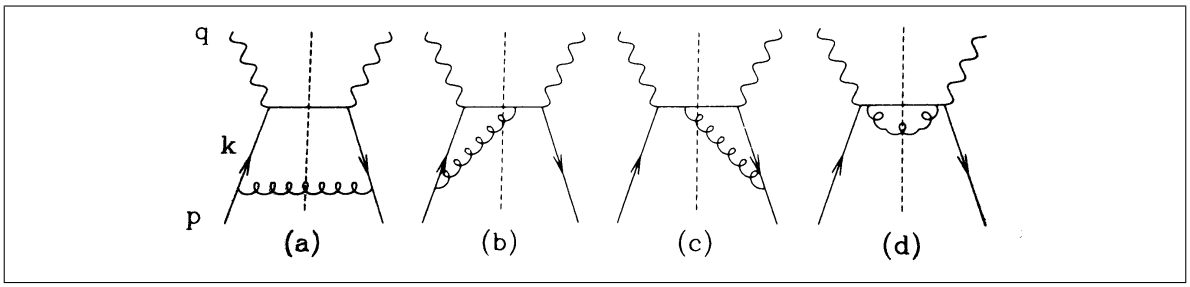


Figure 5: First order squared Feynman diagrams for realistic gluon radiation, reproduced from [4]

For the first order corrections, we need to consider the squared diagrams (figure 5), which represent realistic gluon radiation (both real and virtual). We skip the primary derivation for the corrections to the quark structure function and look only at the final steps. One first gets to:

$$\hat{F}_2(x,\nu) = e_q^2 \frac{\alpha_s}{2\pi}\, x\mathcal{P}(x) \int_{\kappa^2}^{2\nu} \frac{d|k^2|}{|k^2|} = e_q^2 \frac{\alpha_s}{2\pi}\, x\mathcal{P}(x)\, \ln\left(\frac{2\nu}{\kappa^2}\right)$$

Therein α_s is the strong coupling constant and $\mathcal{P}(x)$ a so-called splitting function. Splitting functions incorporate the probability for a parton to emit another parton with momentum fraction x. Furthermore, 2ν is a real upper integral boundary (with ν the DIS standard variable defined earlier), more general one would use μ^2, while κ^2 is a small cut-off value introduced to regularize a divergence. It is clear that the integral is logarithmically divergent for $k \rightarrow 0$. This is called the collinear singularity, it arises when the radiated gluon is emitted parallel to the quark. Other divergences surface at different points in the calculation, but all of them are cancelled. It is important to realize that this one is different because it extends into the long-range part of QCD where perturbative calculations aren't valid.

To continue, we first write the quark structure function again, now including the already known, zeroth, leading order, next to the newly calculated first order. At the same time we rewrite $\ln(2\nu)$ as $\ln Q^2 - \ln x$ and put some finite corrections and $\ln(x)$ into function $c(x)$. The ellipsis indicates even higher orders.

$$\hat{F}_2(x,Q^2) = e_q^2 x \left[\delta(1-x) + \frac{\alpha_s}{2\pi}\left(\mathcal{P}(x)\ln\frac{Q^2}{\kappa^2} + c(x)\right) + \ldots\right]$$

It is obvious that at first order, $\hat{F}_2$ is scale dependent, breaking scale in logarithms of Q^2.

Using this expression for the quark structure function, we progress to the proton structure function $F_2(x, Q^2)$ by adding parton distribution functions and summing over all flavours.

$$F_2(x,Q^2) = x\sum_q e_q^2 \left[f_q^0(x) + \frac{\alpha_s}{2\pi}\int_x^1 \frac{d\xi}{\xi} f_q^0(\xi)\left(\mathcal{P}\left(\frac{x}{\xi}\right)\ln\frac{Q^2}{\kappa^2} + c\left(\frac{x}{\xi}\right)\right) + \ldots \right]$$

To get rid of the singular term in $\ln\kappa^2$, we split the logarithm in two parts at some scale μ^2 and absorb the part with κ^2 in a redefined parton distribution. This step is similar to renormalization, in field theory, but not the same. It's called factorization. With factorization scale μ^2 we have:

$$f_q(x,\mu^2) = f_q^0(x) + \frac{\alpha_s}{2\pi}\int_x^1 \frac{d\xi}{\xi} f_q^0(x)\left(\mathcal{P}\left(\frac{x}{\xi}\right)\ln\frac{\mu^2}{\kappa^2} + c\left(\frac{x}{\xi}\right)\right) + \ldots$$

and with that:

$$F_2(x,Q^2) = x\sum_q e_q^2 \int_x^1 \frac{d\xi}{\xi} f_q(\xi,\mu^2)\cdot\left[\delta\left(1-\frac{x}{\xi}\right) + \frac{\alpha_s}{2\pi}\mathcal{P}\left(\frac{x}{\xi}\right)\ln\frac{Q^2}{\mu^2} + \ldots\right]$$

It should be noted that the defined parton distribution cannot be calculated, as it contains contributions from the non-perturbative region of QCD. It can, however, be extracted from experimental structure function data. Another interesting fact is that, while we have shown only results up to first order, factorization (the possibility to separate perturbative and non-perturbative parts) can be proven to all orders. Finally, we mention that a certain arbitrariness remains in factorizing the expressions, as multiple options exist to arrange the finite parts into the remaining terms.

Before this section is really finished, we need to add one last set of diagrams and their matching expressions. However, as the idea behind this is entirely similar to the above, it can be done very quickly. We have considered the process $\gamma^* q \rightarrow q$ to zeroth order and its first order correction due to gluon radiation, $\gamma^* q \rightarrow qg$. Now, in the proton there is also the process $\gamma^* g \rightarrow q\bar{q}$ (figure 6). At zeroth order this is absent, but at first order it surfaces, the gluon structure function is non-zero:

$$\hat{F}_2^g(x,Q^2) = x\sum_q e_q^2 \frac{\alpha_s}{2\pi}\left(\mathcal{P}_{qg}(x)\ln\frac{Q^2}{\kappa^2} + c_g(x)\right)$$

Note that we have added an index to the splitting function, something we omitted in the discussion of the quark case. It denotes the emitted and original particle: quark emitted by gluon. In the quark case we should have written $\mathcal{P}_{qq}$, quark emitted by quark.

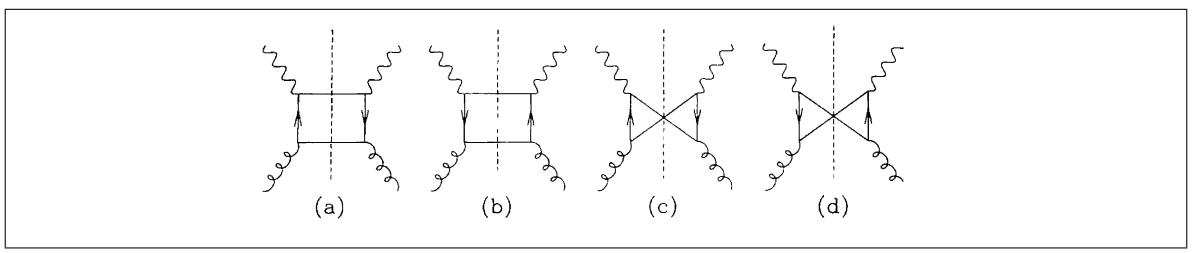


Figure 6: First order squared Feynman diagrams for $\gamma^* g \rightarrow q\bar{q}$, reproduced from [4]

Next we need to add a gluon term to the redefined quark distribution and redefine the gluon distribution in the same way:

$$f_q(x,\mu^2) = f_q^0(x) + \frac{\alpha_s}{2\pi}\int_x^1 \frac{d\xi}{\xi} f_q^0(x)\left(\mathcal{P}_{qq}\left(\frac{x}{\xi}\right)\ln\frac{\mu^2}{\kappa^2} + c_q\left(\frac{x}{\xi}\right)\right)$$
$$+ \frac{\alpha_s}{2\pi}\int_x^1 \frac{d\xi}{\xi} f_g^0(x)\left(\mathcal{P}_{qg}\left(\frac{x}{\xi}\right)\ln\frac{\mu^2}{\kappa^2} + c_g\left(\frac{x}{\xi}\right)\right) \tag{2}$$

$$f_g(x,\mu^2) = f_g^0(x) + \frac{\alpha_s}{2\pi}\int_x^1 \frac{d\xi}{\xi} f_q^0(x)\left(\mathcal{P}_{gq}\left(\frac{x}{\xi}\right)\ln\frac{\mu^2}{\kappa^2} + c_q\left(\frac{x}{\xi}\right)\right)$$
$$+ \frac{\alpha_s}{2\pi}\int_x^1 \frac{d\xi}{\xi} f_g^0(x)\left(\mathcal{P}_{gg}\left(\frac{x}{\xi}\right)\ln\frac{\mu^2}{\kappa^2} + c_g\left(\frac{x}{\xi}\right)\right) \tag{3}$$

The final result for the proton structure function is:

$$F_2(x,Q^2) = x\sum_q e_q^2 \int_x^1 \frac{d\xi}{\xi} f_q(\xi,\mu^2)\cdot\left[\delta\left(1-\frac{x}{\xi}\right) + \frac{\alpha_s}{2\pi}C_q^{\overline{MS}}\left(\frac{x}{\xi}\right) + \ldots\right]$$
$$+ x\sum_q e_q^2 \int_x^1 \frac{d\xi}{\xi} f_g(\xi,\mu^2)\left[\frac{\alpha_s}{2\pi}C_g^{\overline{MS}}\left(\frac{x}{\xi}\right) + \ldots\right]$$

We have changed to a different factorization scheme, the modified minimal substraction scheme ($\overline{MS}$). It is more common and more usable at higher orders than the previously used DIS scheme. That said, we shall consider anything else concerning this section as too specific for this introduction and refer the more interested reader to [1] and [4] for more details and explicit calculations.

1.2.6 Parton density functions

In the previous sections we introduced parton density functions (PDF) $f_i(x,\mu^2)$ as the probability distribution of partons i to be found carrying fraction x of the proton momentum, at factorization scale μ^2. In fact, when calculating the proton structure function, the PDFs provide a non-perturbative but universal contribution, while the partonic structure function provide the rest of the information and are perturbatively calculable. The separation of these two contributions is what we called factorization. This type of factorization, coming forth from taking care of the collinear singularity, is called collinear factorization. Note that PDFs cannot be calculated from first principles, they have to be determined experimentally, for example through the determination of the proton structure function. It is however possible, once an initial distribution is known, to calculate 'evolved' distributions in momentum fraction x or scale μ^2, using different types of evolution equations (section 1.2.7). Also note that in this and the following section we will work in the pp-scattering picture instead of the DIS picture we used before.

Collinear factorization and the use standard PDFs are fine in moderate-x regions, when considering total (inclusive) cross sections (e.g. looking at proton structure function F_2). Cross sections can be written as:

$$\sigma = \int dx_1 dx_2 \; f_i(x_1,\mu^2) f_j(x_2,\mu^2)\; \hat{\sigma}_{ij}(x_1,x_2,\mu^2)$$

and one can reliably use DGLAP evolution to evolve initial distributions at scale μ_0^2 to a higher scale Q^2. However, in low-x regions, when also looking at exclusive final state processes, another type of factorization becomes more important. It becomes necessary to include the transverse momentum k_T^2 of the partons and DGLAP evolution no longer works. In other words, when parton transverse momenta k_T^2 cannot be neglected with respect to Q^2 and k_T^2-integrals cannot be performed indepently, we need a new type of factorization, k_T^2 dependent PDFs and other evolution equations (BFKL, CCFM).

We are talking about k_T-factorization and unintegrated parton density functions (uPDFs). k_T-factorization handles structure functions as convolutions of uPDFs (function of x and k_T) and partonic structure functions, as opposed to the previous convolutions of PDFs (function of x) and partonic structure functions. This is still a combination of a non-perturbative and a perturbative part. Note that in this low-x region we do not require k_T-ordering in cascades, which is needed in the DGLAP moderate-x region. This will be clarified in the next section (1.2.7). With k_T-factorization we can write cross sections as:

$$\sigma = \int dx_1 dx_2 dk_{T,1}^2 dk_{T,2}^2 \; a_i(x_1, k_{T,1}, \mu^2) a_j(x_2, k_{T,2}, \mu^2) \; \hat{\sigma}_{ij}(x_1, x_2, k_{T,1}, k_{T,2}, \mu^2)$$

where the $a_i(x, k_T)$ are uPDFs and $\hat{\sigma}_{ij}$ the partonic cross section. Finally we remark that in the low-x context references often discuss the theory in terms of gluon distributions only, as in this region the gluon is by far the dominant parton in the proton. For further information we refer to [1] and [2].

1.2.7 Evolution equations

Parton distributions $f(x, \mu^2)$ have been defined in above equations (2) and (3). We stated that they can be extracted from the proton structure function $F_2(x, Q^2)$, experimentally. However, since this is a physical observable and scale μ^2 is un-physical, the scale dependence of the parton distributions cannot be function of μ^2. It is governed by the Dokshitzer-Gribov-Lipatov-Altarelli-Parisi (DGLAP) equations, a set of integro-differential equations. They can be constructed from the equations for the parton distributions by taking the partial derivative with respect to $\ln \mu^2$, often abbreviated as $\ln t$. This results in:

$$\frac{\partial f_q(x, \mu^2)}{\partial \ln \mu^2} = \frac{\alpha_s(\mu^2)}{2\pi} \int_x^1 \frac{d\xi}{\xi} \left[\mathcal{P}_{qq}\left(\frac{x}{\xi}, \alpha_s(\mu^2)\right) f_q(\xi, \mu^2) + \mathcal{P}_{qg}\left(\frac{x}{\xi}, \alpha_s(\mu^2)\right) f_g(\xi, \mu^2) \right]$$

$$\frac{\partial f_g(x, \mu^2)}{\partial \ln \mu^2} = \frac{\alpha_s(\mu^2)}{2\pi} \int_x^1 \frac{d\xi}{\xi} \left[\sum_q \mathcal{P}_{gq}\left(\frac{x}{\xi}, \alpha_s(\mu^2)\right) f_q(\xi, \mu^2) + \mathcal{P}_{gg}\left(\frac{x}{\xi}, \alpha_s(\mu^2)\right) f_g(\xi, \mu^2) \right]$$

Note that we added scale dependence for α_s and dependence on that $\alpha_s(\mu^2)$ for the splitting functions. This does not follow from what we have shown before but can be proven, references can be found in [4]. Splitting functions $\mathcal{P}$ may be seen as a power series, a perturbative expansion in coupling $\alpha_s(\mu^2)$:

$$\mathcal{P}(x, \alpha_s(\mu^2)) = \mathcal{P}^{(0)}(x) + \sum_n \left(\frac{\alpha_s(\mu^2)}{2\pi}\right)^n \mathcal{P}^{(n)}(x)$$

Now for the physical meaning of the equations. They indicate that a parton carrying momentum fraction x may originate from another parton, carrying a larger momentum fraction ξ. If we take the quark as an example this would translate to: quark q with distribution

$f_q(x, \mu^2)$ could be created either from another quark with distribution $f_q(\xi, \mu^2)$ which radiated a gluon, or from a gluon with distribution $f_g(\xi, \mu^2)$ through $q\bar{q}$ pair creation. The probabilities of either type of radiation happening are represented by one of the splitting functions $\mathcal{P}\left(\frac{x}{\xi}\right)$. The fact that we have these expressions governing the scaling of parton distributions means that, given a parton distribution $f(x, Q_0^2)$ at a certain scale Q_0^2 (say acquired experimentally), we are capable of evolving this distribution to a higher scale $Q^2 > Q_0^2$ by using a 'simple' equation. Note however that the x dependence is not governed by the equation an thus cannot be calculated, it has to determined experimentally.

Now if we consider perturbation theory again, it can be found that in certain limits some sums may be resummed analytically. Different limits and resummations match different evolution equations, the DGLAP equations are not the only possibility. Also, different situations match differently ordered ladders when considering Feynman diagrams.

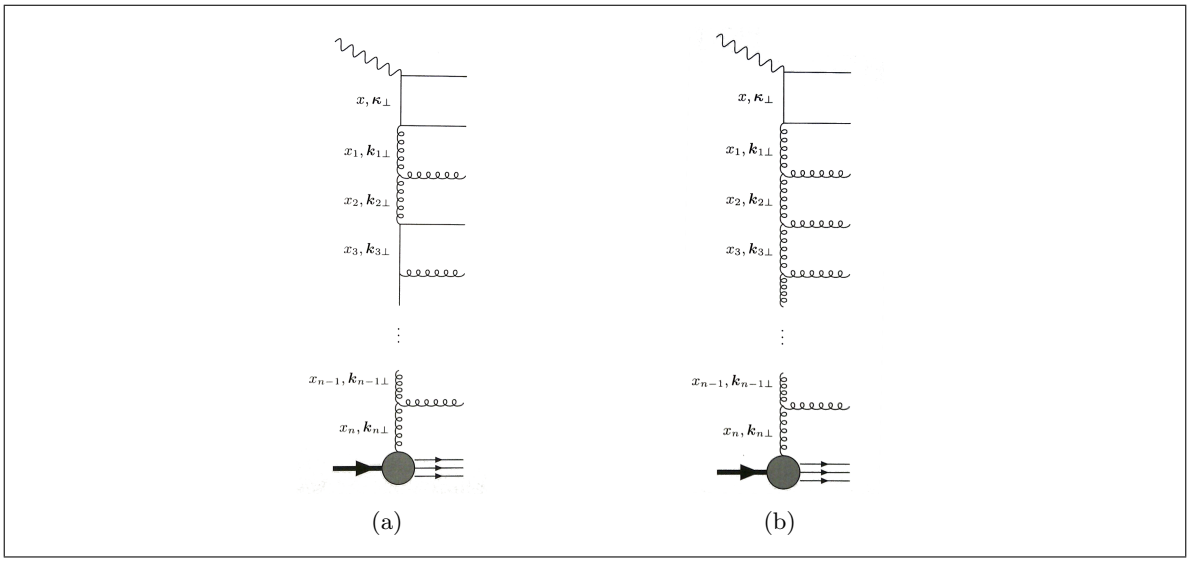


Figure 7: Feynman ladder diagrams, reproduced from [1]

DGLAP If we look at the DGLAP equation we already mentioned, we are talking large $\ln Q^2$ limit, or rather $\ln Q^2 \gg \ln \frac{1}{x}$. The resummation considers terms in $\left[\alpha_s^n \ln^n Q^2\right]$, retaining only the leading order in $\ln Q^2$ at each perturbative order. For the ordering in a cascade (which can be visualized with a Feynman diagram), this case means strong ordering in k_T^2 and normal ordering in x, see figure 7(a). Translating this to mathematics we have:

$$Q^2 \gg k_{n,T}^2 \gg \ldots \gg k_{2,T}^2 \gg k_{1,T}^2 \qquad \text{and} \qquad 1 > x_n > \ldots > x_2 > x_1$$

leading to approximation:

$$\int^{Q^2} \frac{dk_{n,T}^2}{k_{n,T}^2} \cdots \int^{k_{3,T}^2} \frac{dk_{2,T}^2}{k_{2,T}^2} \int^{k_{2,T}^2} \frac{dk_{1,T}^2}{k_{1,T}^2} \sim \frac{\ln^n Q^2}{n!}$$

Retaining only the leading order in $\ln Q^2$ does not affect x. In any perturbative expansion in this limit (omitting coefficients) we still have:

$$\sum_n \alpha_s^n \ln^n Q^2 \left(\ln^n \frac{1}{x} + \ln^{n-1} \frac{1}{x} + \ldots\right)$$

This is called the Leading Log Approximation (LLA). Naturally, one can also expand the treatment to next-to-leading order in $\ln Q^2$, which would be the Next-to-Leading Log Approximation (NLLA) and so on. These approximations match the DGLAP equations at different orders: leading order DGLAP and next-to-leading order DGLAP respectively.

BFKL Opposite to the large $\ln Q^2$ limit there is the large $\ln \frac{1}{x}$ limit, or $\ln \frac{1}{x} \gg \ln Q^2$. This is what one is dealing with for example at HERA (see section 2.1) when looking at low-x physics. $\ln Q^2$ is then not so large any longer and it is appropriate to consider another resummation. The matching ordering is strong ordering in x and no ordering in k_T^2. One can call this the Leading Log_x Approximation (LL_xA) and it matches a second evolution equation, the Balitsky-Fadin-Kuraev-Lipatov (BFKL) equation.

We can write:

$$1 \gg x_n \gg \ldots \gg x_2 \gg x_1 \qquad \text{and} \qquad k_{1,T}^2 \simeq k_{2,T}^2 \simeq \ldots \simeq k_{n,T}^2$$

and retain perturbative expansion form:

$$\sum_n \alpha_s^n \ln^n \frac{1}{x} \left(\ln^n Q^2 + \ln^{n-1} Q^2 + \ldots \right)$$

Once again one can improve the treatment by including higher orders and considering NLL_xA and so on.

DLLA One does not necessarily have to have either one or the other limit, it's possible to consider both of them together: large Q^2 and small x. This is the Double Leading Log Approximation (DLLA). Diagramatically there needs to be strong ordering in both x and k_T^2, see figure 7(b). We find:

$$1 \gg x_n \gg \ldots \gg x_2 \gg x_1 \qquad \text{and} \qquad Q^2 \gg k_{n,T}^2 \gg \ldots \gg k_{2,T}^2 \gg k_{1,T}^2$$

and for perturbative expansions:

$$\sum_n \alpha_s^n \ln^n Q^2 \ln^n \frac{1}{x}$$

CCFM Both the DGLAP and BFKL equations discussed above have limited regions of applicability. The BFKL equation is not valid in the phase space region where $\ln Q^2$ becomes too large, while the DGLAP equation has the same problem for $\ln \frac{1}{x}$. There is the DLLA taking the middle road, looking at the phase space region where $\ln Q^2$ and $\ln \frac{1}{x}$ are more or less equally important, but isn't there a description valid in the whole phase space (x, Q^2)? Here the CCFM equation comes in. We will not discuss it in detail, but mention a few general properties.

To begin with the CCFM equation reduces to the DGLAP and BFKL equations in the appropriate regions of phase space. The equation considers a new type of ordering, angular ordering of the gluons in the cascade, taking angles with respect to the original direction. This adds a second scale Q^2 to the equation. The reduction to the old equations can happen because on the BFKL side, at small x, angular ordering loses importance and on the DGLAP side, at larger x, angular ordering and ordering in transverse momentum become equivalent. In both cases, the second scale falls away.

1.2.8 Saturation

At small x, evolution equations predict increasing parton density functions. We discussed previously that partons within the proton can be considered free particles. Yet, if their number density would become too large, this would no longer be true. At very small x one may expect the gluon (which is at that point more abundantly present in the proton than the sea quarks) to saturate the proton. If that were the case, gluon recombination effects would become more important, limiting further growth of the gluon density. We have not however seen any real evidence of saturation yet, which is one of the elements that makes accelerator experiments' outreach to progressively lower x regions very interesting.

To end this section we show a diagram of the (x, Q^2) phase space on which the validity regions of the discussed evolution equations, the region where one expects saturation and the non-perturbative region are marked (figure 8).

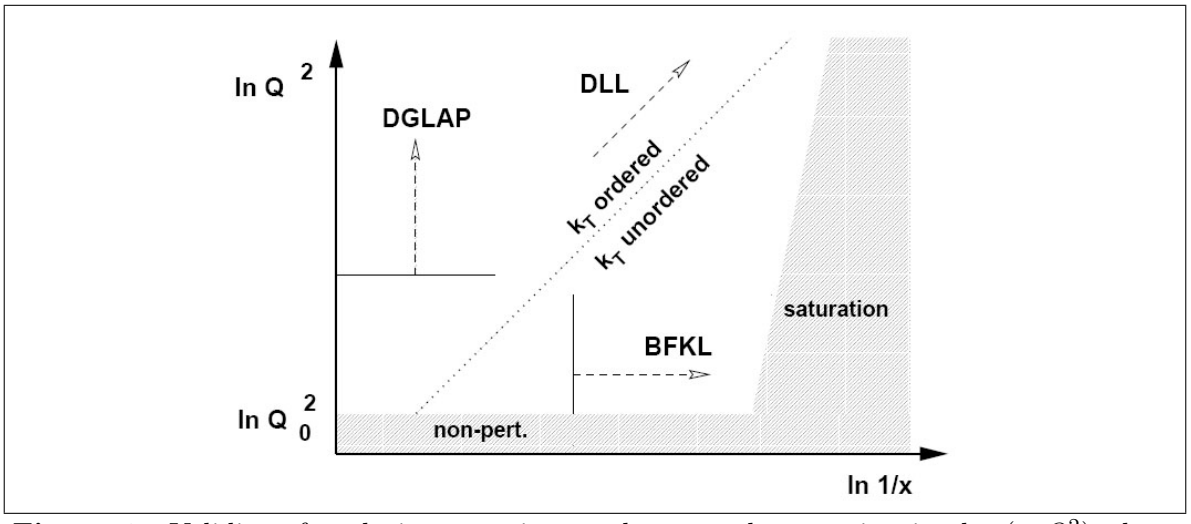


Figure 8: Validity of evolution equations and expected saturation in the (x, Q^2) phase space, reproduced from [10]

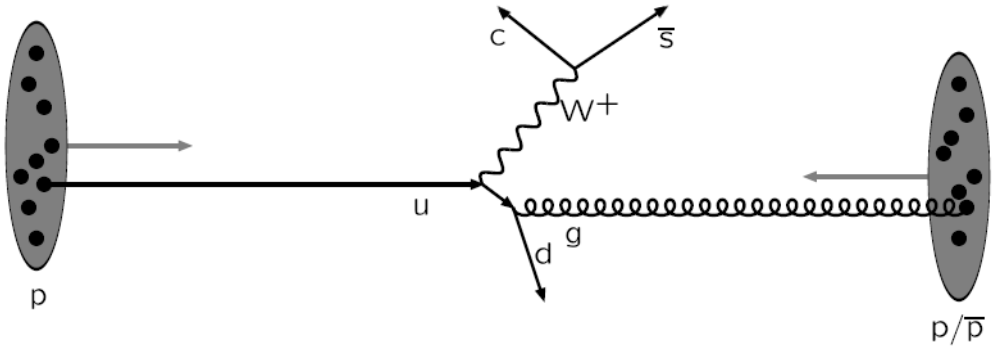
1.3 Underlying events

This overview is based mainly on [23], in turn referencing [36]. When talking about collisions one often discusses primarily the hard process. This is however not the complete event. With the aid of figures 9(a)-9(e) we try give a picture of the different parts of a collision.

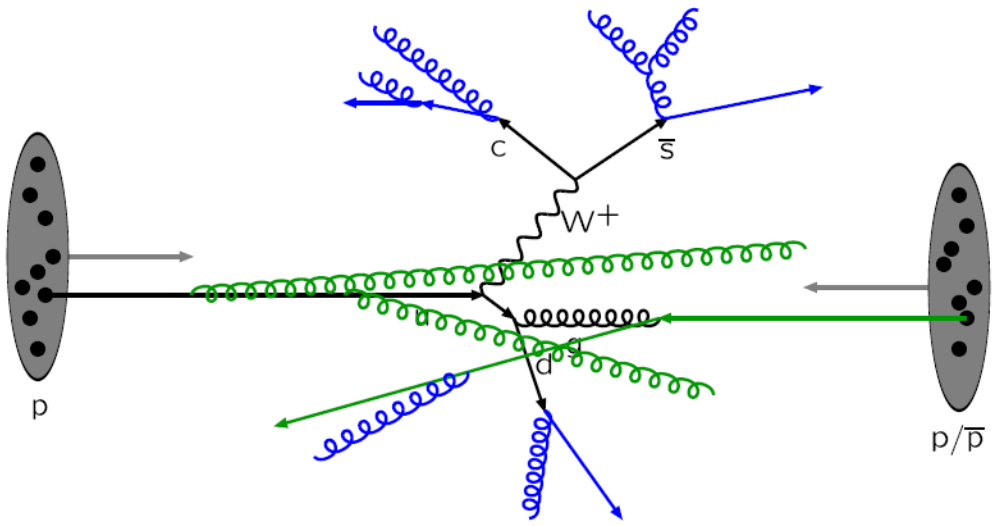
In figure 9(a) we have the ingoing beam particles and the hard subprocess between two partons from the ingoing particles, also including a W^+ resonance. A first addition to complete the picture are initial- and final state radiation, adding a whole bunch of particles to the final state. This is figure 9(b). However, nothing keeps other partons from the ingoing beam particles to interact too. An example of such so-called multiple parton-parton interaction is the gluon-gluon interaction in figure 9(c). Then of course these particles can also radiate, adding even more particles to the final state. Finally in figure 9(d) we hide all the intermediate stages and keep only the ingoing and outgoing particles. Some of the outgoing particles come from the hard subprocess (marked in red), others are beam remnants or come from initial- or final state radiation. If we take out all outgoing particles linked to the hard subprocess (the leading order process), we are left with the underlying event, shown in figure 9(e). Note that due to QCD confinement all outgoing hadrons have to be colour neutral.

This links together all coloured partons in the final state and makes the separation of hadrons coming from the hard subprocess and those part of the underlying event, a bit ambiguous. It is ambiguous because partons coming from the hard subprocess and partons part of the underlying event (this is a clear separation) can easily fragment into hadrons together, ending the clear separation between hard subprocess and underlying event. Also important to take into account, not shown in the figures, would be pileup. At high-energy accelators such as LHC it is possible that more than one pp-collision takes place when the proton bunches cross. Some of the interactions originating from pileup are not hard scatterings and thus can't be part of the hard process, but neither are they generally considered as part of the underlying event. A final point of discussion is the handling of multiple interactions in event generators. As it is not fully understood, it has to be handled according to a phenomenological model (just like fragmentation, see section 3).

In any case it can be concluded that for experimentators, it is important to properly understand underlying events (including common QCD processes), to be able to look for interesting and exotic hard processes in an efficient way.

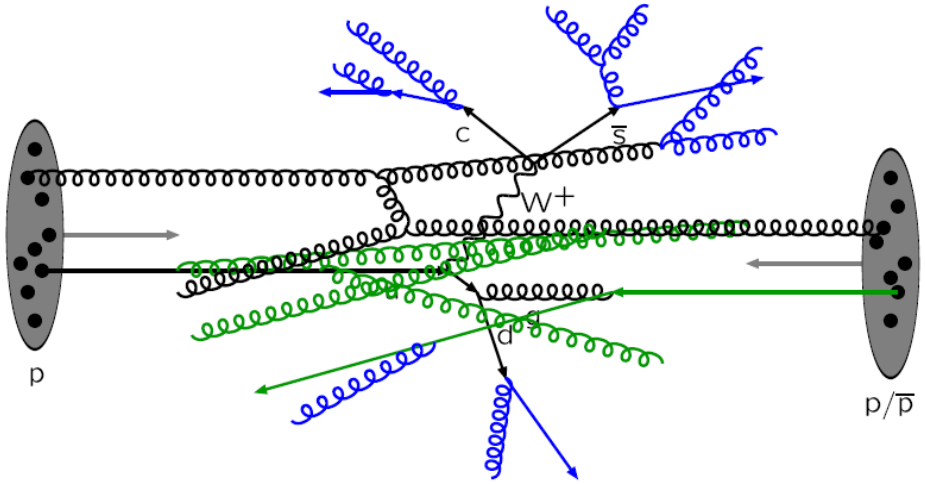


(a) Incoming beam particles and hard process

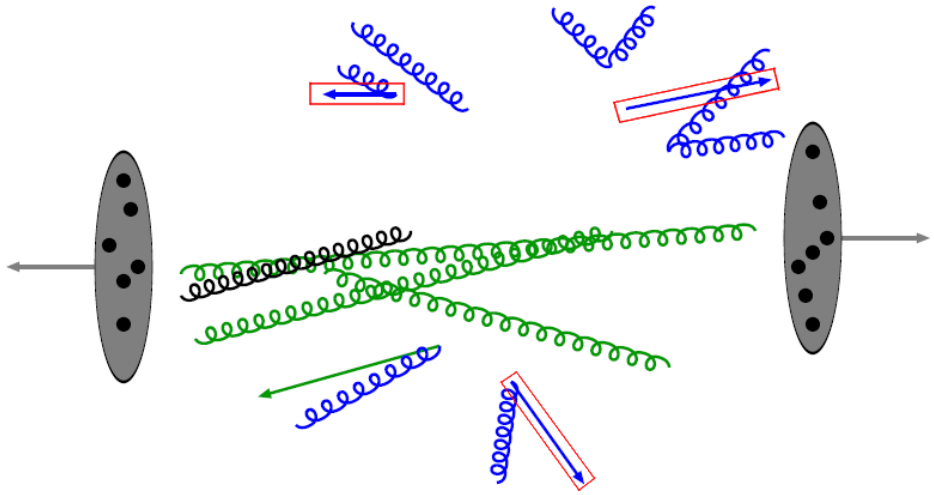


(b) Initial- (green) and final state radiation (blue)

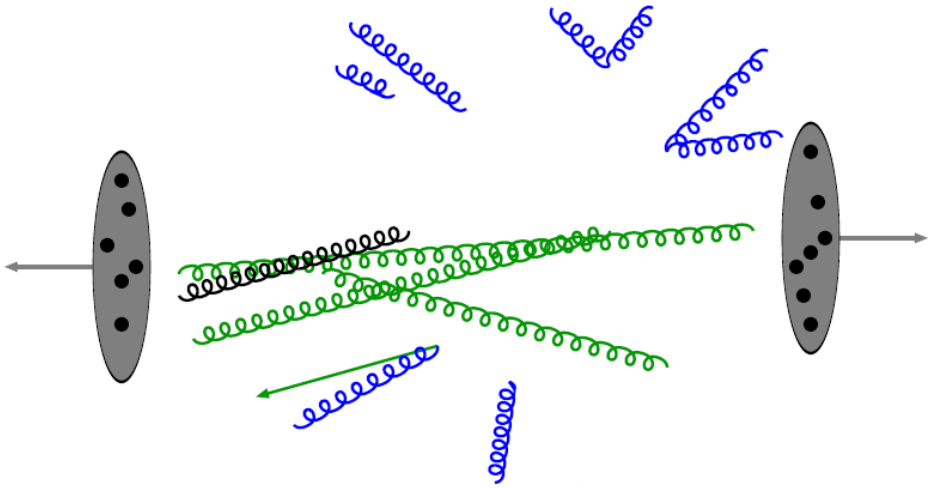
Figure 9: Different parts of a hadronic collision [23]



(c) Additional interactions (black) with their own ISR and FSR



(d) All outgoing particles



(e) Underlying event

Figure 9: (cont.) Different parts of a hadronic collision [23]

2 Particle accelerators

2.1 Hadron-Elektron Ring Anlage

2.1.1 Introduction

HERA (Hadron-Elektron Ring Anlage) was a particle accelerator operated at DESY (Deutsches Elektronen Synchrotron) in Hamburg, Germany, between 1992 and 2007. HERA was a lepton-proton collider, capable of both e^-p- and e^+p-collisions. This is different from other accelerators. LEP (Large Electron-Positron collider, CERN, Geneva) produced electron-positron collisions, Tevatron (Fermilab, Illinois) produces proton-antiproton collisions and LHC (Large Hadron Collider, CERN, Geneva) produces proton-proton collisions.

More specifically, HERA was a circular accelerator with two rings with a circumference of 6.3 km, one for the leptons and one for the protons. During the final period of its operation, the accelerator produced collisions at a centre-of-mass energy of 318 GeV. In several stages, first before and then within the HERA main rings, protons were accelerated to 920 GeV and electrons to 27.6 GeV, resulting in said centre-of-mass energy. We show a schematic overview of the rings and the interaction points in figure 10.

A circular accelerator also requires the use of magnets to keep the particles on their circular orbit, which is in reality more square with rounded corners than circular. On a curved path particles (the electrons more so than the protons because of their smaller mass) produce synchrotron radiation, which would damage detector equipment placed around the beampipe. To protect the detectors, which we will discuss below, they are located on the straight parts of the accelerator ring, interaction regions, one on each side of the square. Notwithstanding these straight parts, the accelerated particles still had to follow a circular path in general, which is where the magnets came in. In the case of the protons, the bending required a rather strong magnetic field and superconducting magnets producing a field strength of 4.7 T were used. In the case of the leptons, non-superconducting sufficed to reach the required field strength of 0.17 T.

Of the four detector experiments, two were colliding beam experiments (H1 and ZEUS) and two were fixed target experiments (HERMES and HERA-B). As this work concerns mainly the structure of the proton, we will focus on the H1 and ZEUS detectors and not the HERMES experiment, which focussed on the spin structure of nucleons, or the HERA-B experiment, which focussed on CP-violation.

2.1.2 H1 detector

The H1 detector was located in the North Hall of the HERA accelerator. It was built to measure the properties of particles resulting from the ep-collisions and thus study the structure of the proton. These measurements included measurements of energy, momentum and charge as well as tracking. Combination of multiple properties then aided particle identification. To acquire all these different kinds of information, multiple elements needed to be part of the H1 detector: a tracking system, a calorimeter and muon chambers.

The tracking system made up the inner part of the detector, including a silicon tracker and several kinds of wire chambers combined in a layered way. Tracks were measured with an accuracy of about 100 μm, and determination of their curvature allowed calculation of the charge and momentum of charged particles. The calorimeter was located around the tracking system. Its two main tasks were energy measurements and track reconstructing for neutral

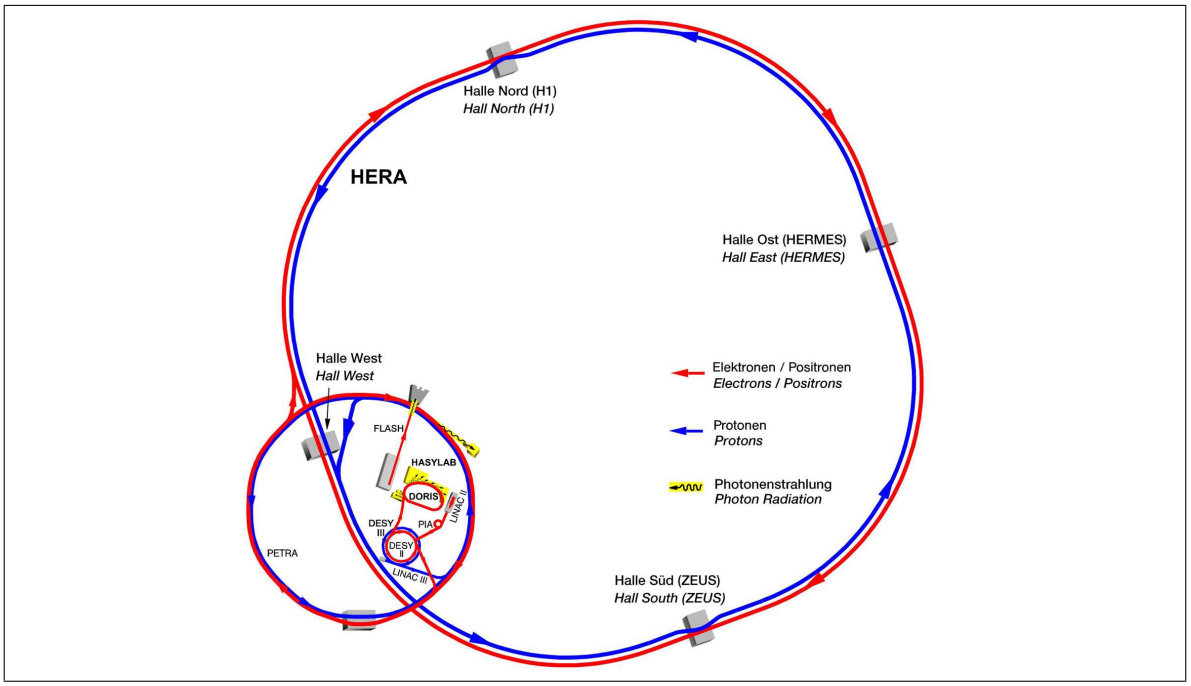


Figure 10: Schematic view of the HERA accelerator [14]

particles. The distinction between tracking of charged (in the tracker) and neutral (in the calorimeter) particles had to be made because electrically neutral particles were invisible in the tracker and thus needed to be tracked elsewhere. The calorimeter consisted of two parts: one electromagnetic and one hadronic. In both cases the study of particle showers produced in absorber materials lead to the needed measurements to calculate the particle energy. Note that for the measurements another material was needed, an active material next to the absorbing (passive) material. Active material (liquid argon and scintillating fibers) and absorber (lead and steel) were alternated and divided into small cells to ensure a good granularity for the detector. The outer part of the H1 detector was the muon detector. Muons were capable of passing through the entire calorimeter unstopped, continuing past it to exit the detector. The positioning of drift chambers on the outer shell and at the ends of the detector allowed the detection of those passing muons, which helped identify certain particle reactions. A schematic view of the detector is given in figure 11.

2.1.3 ZEUS detector

The ZEUS detector was located in the South Hall of the HERA accelerator. In general, it had the same purpose as the H1 detector, studying the ep-collision products and therewith the proton structure. Its central tracking system also included a silicon track detector, used for vertex detection and several wire chambers. The calorimeter again surrounded the tracking system, with first the electromagnetic part and then the hadronic part. The system made use of uranium as active material for the calorimeter. This is interesting because it helps improving the energy measurement by enhancing the hadronic signal so that it becomes closer to level of the electromagnetic signal. For optimal energy measurements the fraction electromagnetic/hadronic signal should be as close to one as possible, while in most detectors it is bigger. Calorimeters including methods such as this one to enhance the hadronic signal are called compensating calorimeters. The muon chambers and spectrometer again comprise the outer part of the detector. The detector is shown schematically in figure 12.

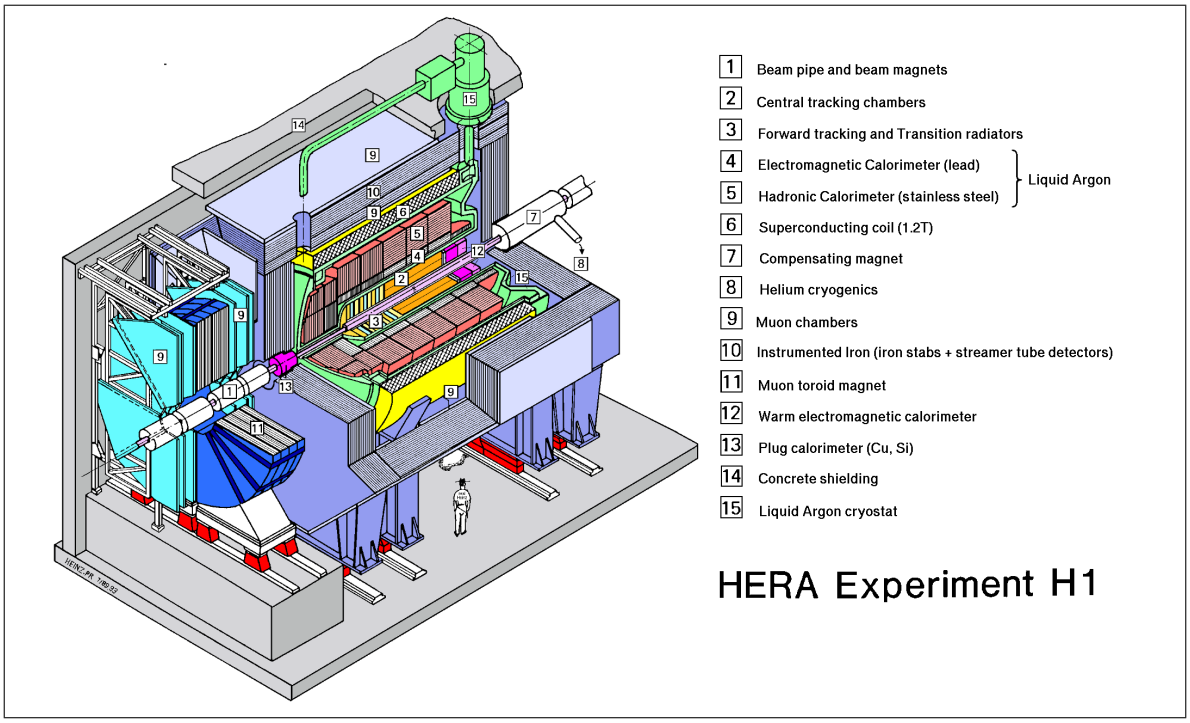


Figure 11: Schematic view of the H1 detector at HERA [22]b

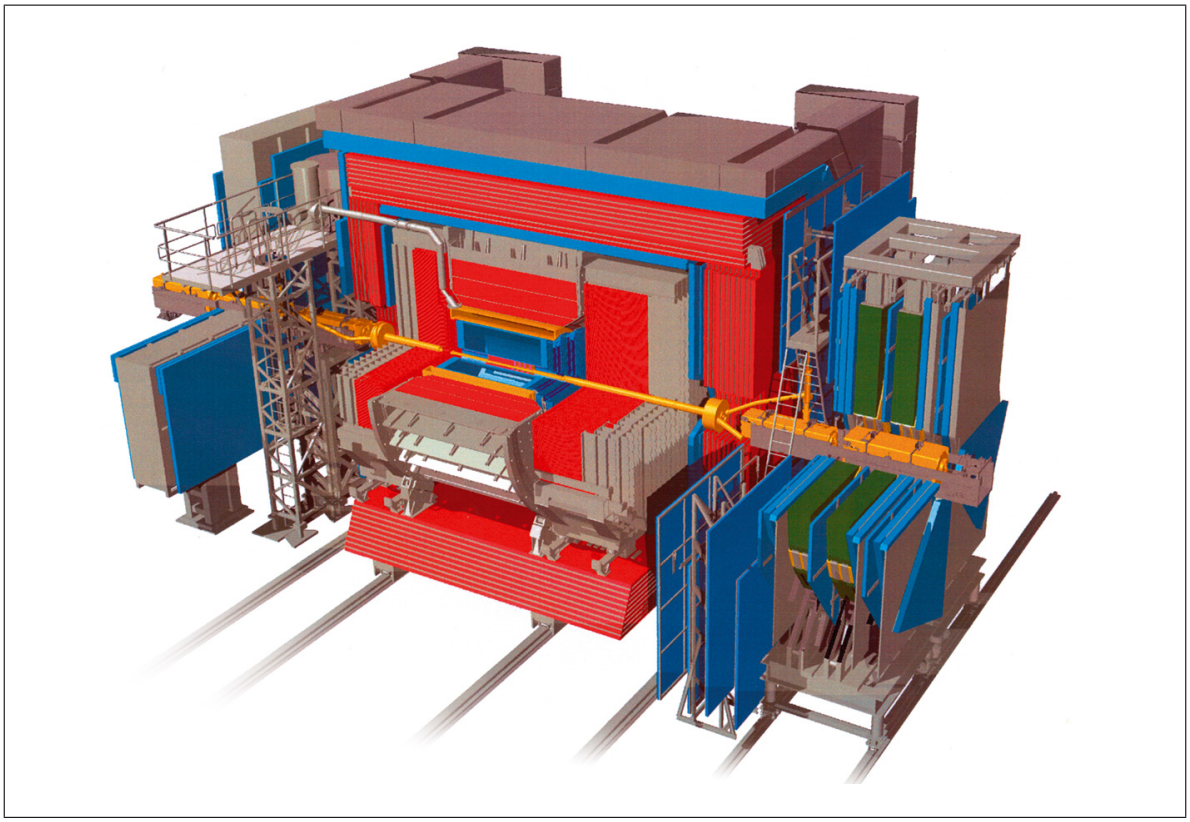


Figure 12: Schematic view of the ZEUS detector at HERA [22]c

2.2 Large Hadron Collider

LHC (Large Hadron Collider) is the most powerful particle accelerator built to date. It's located at CERN in Geneva. Its operation first started in september 2008, but technical problems required for the accelerator to be shut down again until november 2009. The LHC is a circular collider producing proton-proton collisions. In two vacuum beampipes in the accelerator tunnel, both hadron beams are accelerated in opposite directions to energies up to 7 TeV. This results in a centre-of-mass collision energy of 14 TeV. Note that these numbers are design values and that currently experiments are being performed at half power, so with a centre-of-mass energy of 7 TeV. It is planned to continue like this until the end of 2012 and to progress to 14 TeV only after a long technical stop in 2013.

The LHC, built in the former LEP tunnel, has a circumference of about 27 km. On the ring there are four interaction points with experimental halls housing different detectors. Two general-purpose (high-luminosity) detectors are located at point 5 and point 1: CMS (Compact Muon Solenoid) and ATLAS (A Toroidal LHC ApparatuS). The two other detector experiments can be found at point 2 and point 8: ALICE (A Large Ion Collider Experiment) and LHCb (LHC beauty). Just like HERA, the LHC needs dipole magnets to keep the beams on their 'circular' track, while quadrupole magnets are responsible for focusing. For this performant accelerator high field strenghts are required to bend the beams. A field strength of 8.3 T can be reached with the superconducting magnets present in the tunnel, cooled to a temperature of 1.9 K using liquid helium. A schematic overview of the ring structure and the interaction points is given in figure 13.

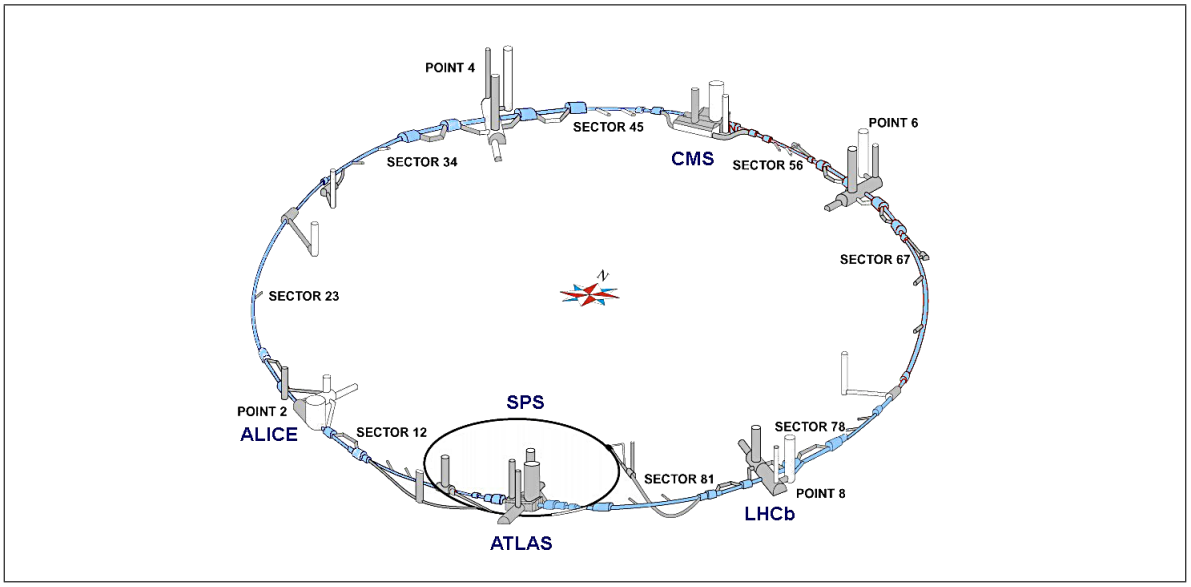


Figure 13: Schematic view of the LHC accelerator [27]

The research goals of the LHC include among others high energy physics, the Higgs Boson, dark matter and dark energy, the graviton and CP violation. Considering the higher energy regions in which it operates, the LHC can also continue to improve the picture we have of the proton structure. In figure 14 a fit of the proton structure to HERA experimental data is shown. It is clear that the structure function rises at smaller x. However, this rise cannot continue indefinitely. One of the tasks of LHC is to provide answers as to what happens in this lower x, higher energy region. In the following we will take a closer look at CMS, one of the general purpose detectors, capable of providing information to help reach that goal.

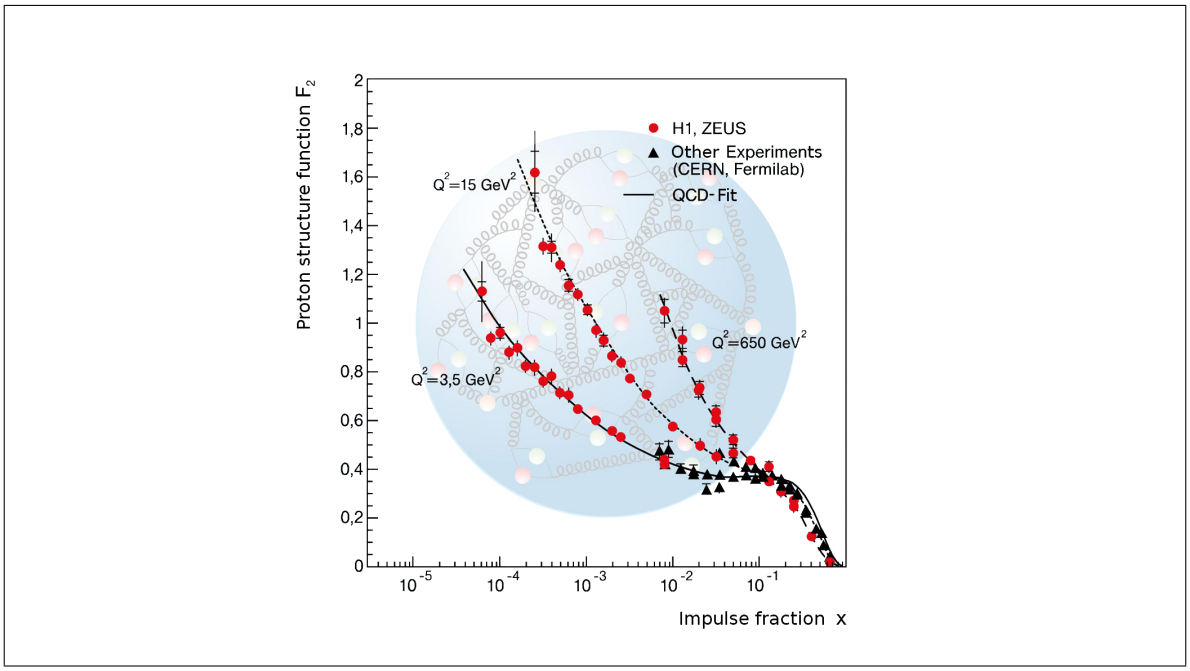


Figure 14: Proton structure function fit on experimental data [22]d

2.2.1 CMS detector

The CMS detector has a layered structure not unlike that of the previously described HERA detectors H1 and ZEUS. The inner part of the detector is the tracker, around it calorimeter layers are placed and on the outside, on both sides and inbetween parts of the iron return yoke, there are muon chambers. We show a slice of the detector in figure 15.

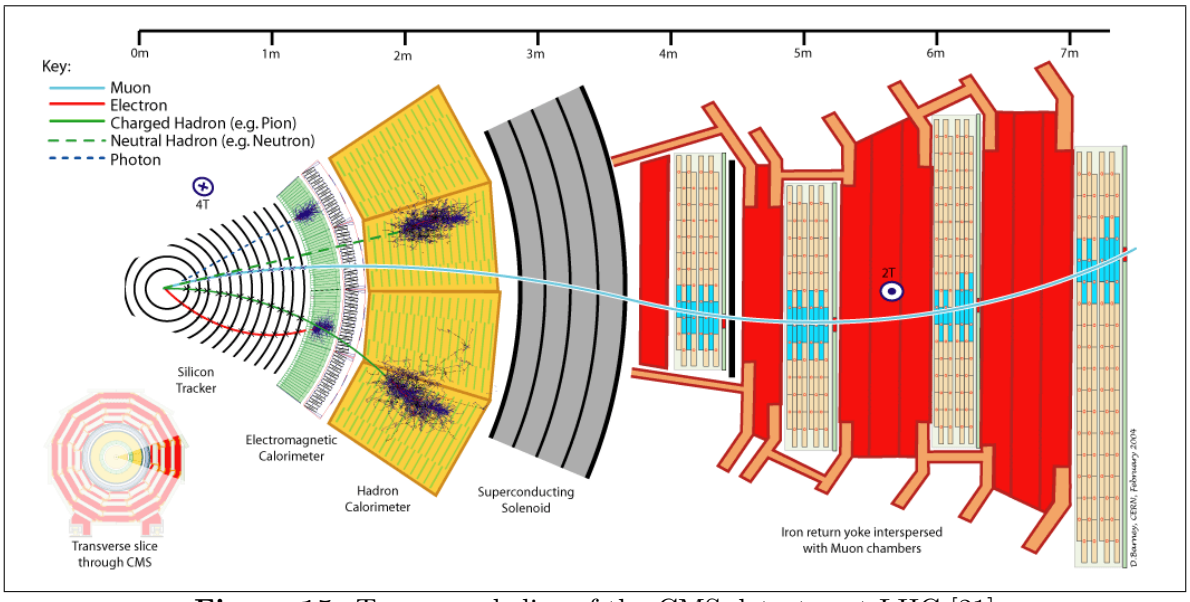


Figure 15: Transversal slice of the CMS detector at LHC [21]

To gather information with the highest possible efficiency, high resolution for tracking and electromagnetic calorimetry are important, as wel as having a hermetic hadronic calorimeter, stopping as many particles as possible.

The CMS tracker uses silicon pixels and silicon strips to detect the particles' paths. A charged particle passing through the silicon parts may induce an electron-hole pair which can be registered. From a set of such hits a path can be reconstructed. The electromagnetic calorimeter uses lead-tungstate scintillator crystals to measure the energy of electromagnetically interacting particles. The hadronic calorimeter is a sampling calorimeter, alternating active and passive materials to measure and slow the strongly interacting particles coming through. The task of the magnet, in figure 15 following the hadronic calorimeter, is to bend the particle tracks allowing a precise impulse measurement. This magnet is again a superconducting magnet, with a field strenght of 3.8 T. Finally the outer detector layer is responsible for muon detection. It is not possible to stop them, but their passage and path can be registered and helps with interaction identification. We show a schematic view of the detector in figure 16.

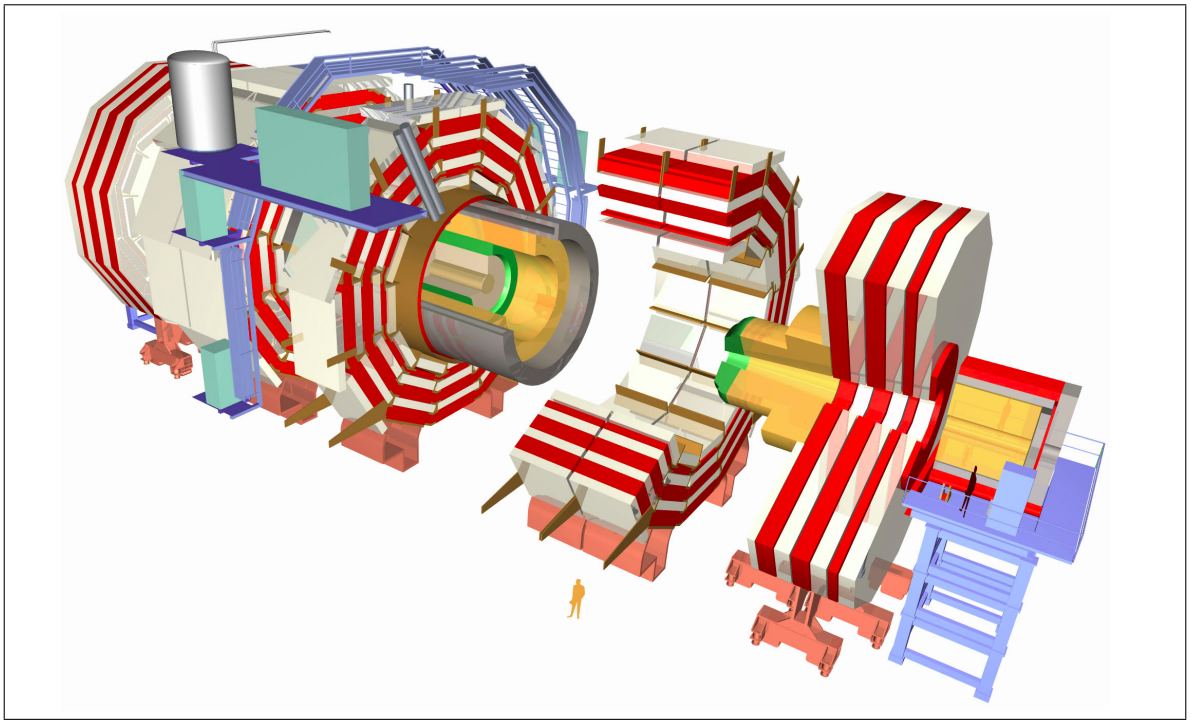


Figure 16: Schematic view of the CMS detector at LHC [20]

3 Event generators

An event generator has the task of simulating events, starting from information about the ingoing particles and working towards a result for the outgoing particles, including all relevant subprocesses and losing or omitting as little information as possible about particles at any stage of the process. We based this overview mainly on the PYTHIA 6.4 reference manual [31].

One of the problems is that full analytical treatment of all steps would be too complicated or even impossible. All processes can be simplified to the extent of "ingoing particles → some hard process → outgoing particles". Then however starts the list of possible amendments to that picture. We consider three main types of corrections. To begin with, both ingoing and outgoing particles may radiate particles which need to be added to the list of final-state particles and which alter the kinematic state of the original particles. These processes can be calculated perturbatively, order by order, but such calculations quickly become very complicated. Alternatively, they can be described probabilistically. Next, there are loop corrections and their combination with the radiative corrections. Again this is quickly too complicated to be handled perturbatively. Finally there is the problem with QCD and perturbation theory. On the side of ingoing particles, descriptions are needed of how a composite particle (a hadron), or rather its constituents (the partons), will participate in the interactions. On the side of the outgoing particles, models are needed which govern the hadronization of the partons produced in subprocesses, since the actual outgoing particles can only be colour neutral objects. It is clear that this list of manipulations cannot be simulated according to a simple "input → one function → output". Instead, event generators divide the leap from ingoing to outgoing particles into smaller, more manageable steps. Next to that, building an event generator is always about making compromises in what to include, and careful consideration about how to include it to gain the best possible result.

Another thing to consider is the step from quantum mechanics in reality, causing fluctuations in real data, to mimicking processes in simulations, causing fluctuations in simulated data. This is where the Monte Carlo tag we often put on event generators comes in. Monte Carlo techniques can be used to generate all relevant variables according to any desired probability distribution, and the resulting events will exhibit precisely the fluctuations we were looking for. Because this is not exactly the same randomness as in reality, we call it quasi-randomness. A handy consequence thereof is the possibility of reproducability. With a properly built random generator one can choose to generate exactly the same numbers twice in a row, which may be handy for example for testing purposes or comparisons. If we were dealing with real randomness this would be impossible.

Now for a slightly less superficial look at event generators. Figure 17 shows a schematic diagram of the different stages in generating an event with a Monte Carlo event generator. We will briefly discuss all of these steps from input to output. Note that in this section we will regularly talk about partons (quarks and gluons) but actually mean all elementary particles in play, so also electrons and photons. This doesn't of course work in any context, but here the electron- and photon-cases are entirely analoguous to the real parton-case, allowing us to generalize.

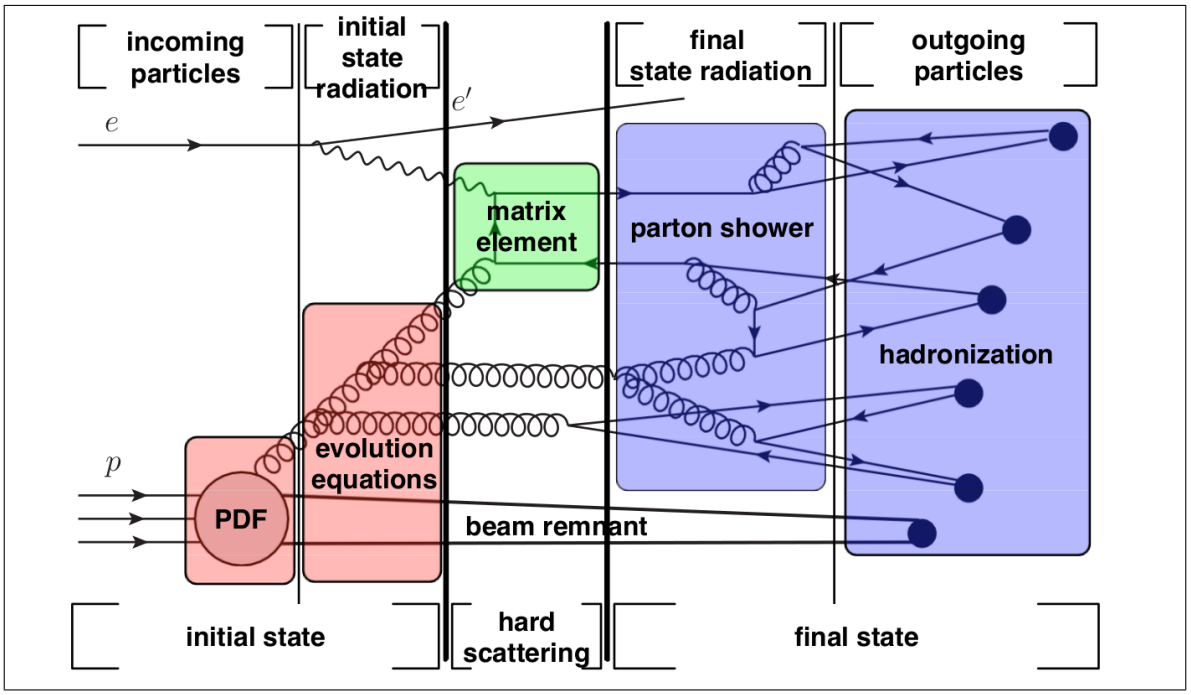


Figure 17: Schematic overview of the stages in Monte Carlo event generation [14]

Initial state The process starts with two particles approaching eachother. In case of a hadron, a set of PDFs determines the substructure, giving the distributions at a certain scale Q^2. In each beam, one parton may start a sequence of branchings (in the form of $q \to qg$ partonically or $e \to e\gamma$ leptonically), this is the initial state radiation building an initial state shower. The way in which this radiation is handled depends on the event generator. An example is the parton shower method, where evolution takes place from Q_0^2 to Q^2. Here for example DGLAP equations would not suffice for evolution because they can't handle parton transverse momenta properly (a problem we discussed in section 1.2.6), thus failing the kinematics side of the event generation. Finally from each initial state shower one parton will participate in the hard process.

Hard scattering The hard scattering, calculable using matrix elements (Feynman calculus) and convolution with the evolved PDFs, will result in outgoing partons, usually two. Before the outgoing partons there may have been a short-lived resonance (for example a Z^0 gauge boson) which decayed into partons.

Final state The (two) outgoing partons of the hard scattering may again start branching (final state radiation), creating final state showers. Then there are three more sources of outgoing particles which may also branch. Firstly semihard interactions may take place next to the hard scattering, between two other partons from the original beam hadrons. Secondly, the leftovers of a hadron, after the parton initiating the initial state radiation is taken out, are not colour neutral and have to be included in the processing of the outgoing partons. We call these leftovers the beam remnant. Thirdly, partons created in the initial state shower also have to be considered in the final state. Now QCD confinement forces hadronization: the outgoing partons will undergo fragmentation (creation of quark-antiquark pairs) and we end up with colour neutral hadrons. Finally some of these hadrons will be unstable and decay. At the end of the process we have the list of the outgoing particles.

Note that fragmentation is a process which is not fully understood theoretically and which has to be described according to a phenomenological model. In this work we will use the PYTHIA 6.4 and PYTHIA 8.1 event generators, both of which by default use the Lund string model for fragmentation. We will discuss the cases in which we use PYTHIA 6.4 and PYTHIA 8.1, and more details pertaining to the two event generators, when we need them in later sections. We end this section with a small elaboration on the parton shower method, the evolution method used by PYTHIA.

The parton shower method sees showering as a sequence of $a \rightarrow bc$ splittings. Therein mother a splits into daughters b and c and these daughters may split in turn. In general, evolution of the PDFs is handled with DGLAP equations, from a low Q_0^2 scale to a higher Q_{max}^2 scale for initial state radiation and from a (possibly different) high Q_{max}^2 scale to a lower Q_0^2 scale for final state radiation. As we stated before though, DGLAP equations cannot handle the whole picture, which is why the event generator in addition carefully tracks the kinematics at each splitting. We need to look at virtuality Q^2 here. $-Q^2 = m^2 = E^2 - \vec{p}^2$ is smaller than zero for spacelike particles and bigger than zero for timelike particles. In DGLAP evolution, virtuality changes at a splitting are neglected when progressing to the next splitting. However, for the kinematics these changes are important.

If we look at the initial state shower, we start with a spacelike particle a. The branch leading towards the hard interaction will gain spacelike virtuality with each splitting, while the other branches become more timelike. Also, the changes in virtuality allow the partons to acquire transverse momentum k_T, which would be impossible in a pure DGLAP description.

If we look at the final state shower, we have timelike particles coming from the hard interaction and timelike particles coming from the remainder of the initial state shower. Evolving down from Q_{max}^2, splittings occur, decreasing virtuality of the particles until all of them are on shell (virtuality zero) and we have our list of outgoing particles.

4 Tools

4.1 System setup

The system used to perform all calculations discussed in this work was set up with the following elements:

- Root Data Analysis Framework - version 5.28/00 - [34]
- Pythia 6.4 Monte Carlo - version 6.4.25 - [31]
- Pythia 8 Monte Carlo - version 8.145 - [32]
- Rivet package - version 1.5.0 - [33]
- Professor package - version 1.2.1 - [30]
- LHAPDF interface - version 5.8.5 - [26]
- OpenMP API - version 3.0 - [28]
- Fedora 14 (Laughlin) - 64bit version
- GCC compiler - version 4.5.1

4.1.1 Root

In nearly every part of this project we will use the Root data analysis framework for data handling. We start with a short introduction and will clarify some elements as we come across them. For more sources and details we refer to [34] and [35].

Root is a data analysis system that allows object-oriented programming and includes a built-in C++ interpreter. It is a substitute for the PAW system based on fortran. The options are virtually endless, but we will use it mainly for data handling. In some cases the easiest way to perform a task is to just open a Root instance and do it by hand or with a simple macro, making use of the built-in C++ interpreter. In other cases it will be more efficient to include one or more Root libraries in the header of a C++ file, hard code the steps that need to be done using Root-functions and then compile in the conventional C++ way.

One of the main benefits of using Root is its capacity to handle large quantities of data in a very efficient way. The use of objects to store data plays a significant role therein. The objects make it possible to call upon or edit only select parts of a data structure, without having to worry about the whole of it. Many mathematical functions and applications such as histogramming and fitting, as well as many options for graphical output are built-in, saving the user a lot of implementation work and the hassle of having to use seperate programs or libraries for some of these tasks.

4.1.2 Pythia 6.4 and Pythia 8.1

Originally the idea for this work was to use only Pythia 8.1 written in C++, and not Pythia 6.4 written in fortran. Both are general purpose event generators as discussed in section 3, based on both analytical results and QCD-based models. For the generation of *pp*-events we can use the newer Pythia 8.1 without any problem. However, Pythia

8.1 does not support the generation of *ep*-events, forcing us to fall back on PYTHIA 6.4 to generate such events. Other than the hassle of having to repeat some implementations in both versions, this has no important consequences. All packages used in the processing of the generated events are compatible with both versions of PYTHIA.

4.1.3 RIVET 1.5.0

For the validation of Monte Carlo event generators, the RIVET 1.5.0 toolkit [33] can be used. RIVET stands for Robust Independent Validation of Experiment and Theory. Again concerning the PDF4MC idea discussed in section 4.2, some way of checking the validity of results obtained with a new PDF is necessary. While this check could perfectly well be performed manually, the RIVET 1.5.0 toolkit provides a far more efficient and much cleaner way of working.

RIVET 1.5.0 is written in C++, in an object-oriented way. It includes a whole range of predefined analyses which are editable, as wel as capabilities to build completely new user-defined analyses. The main principle of the toolkit is the use of projections. In short projections are calculators for physical observables, starting from an event object. The term projection, based on the literal projection of high-dimensional spaces on drawable lower-dimensional objects, does include exactly those projections, but also more esoteric variants of the idea. For example simple event shapes can literally be projected and drawn. On the other hand it is equally possible to project only selective parts of the final state, with cuts for example, or to draw observables that aren't necessarily observable in the non-quantum mechanical sense of the word, such as missing energy carried by invisible particles.

4.1.4 PROFESSOR 1.2.1

Concerning the PDF4MC idea discussed in section 4.2, we will generate deep-inelastic ep-events with PYTHIA 6.4, using an LHAPDF parametrized PDF as input. The resulting cross sections will be function of the PDF parameters. Fitting this result directly to HERA data would require at least a minimal set of PYTHIA 6.4 cross section data points, each the result of a PYTHIA 6.4 run with a certain number of events that cannot be set too low for accuracy purposes. This is a very cpu-intensive task and even with modern computers and helpful tricks such as multi-threading, it is something one would rather avoid. Therefore, we use the PROFESSOR 1.2.1 package.

Generating data points and then fitting as described above would be considered a brute-force method to tune the PDF to the HERA data. The PROFESSOR 1.2.1 package [30] is a tool meant to approach this problem of optimization in a different way, using interpolation. The cpu-intensive function, in this case the PYTHIA 6.4 run, or rather it's response to changes in the input parameters, will be parametrized is a deterministic way. The optimization of the input parameters, requiring a lot of sampling, can then be performed in a more systematical and iteratively improving way.

The package name stands for PROcedure For EStimating Systematic errORs and it is written in Python. The tuning through interpolation method of PROFESSOR 1.2.1 can be seen as follows: N parameter points are sampled randomly in an n-dimensional space. For each of the parameter points the generator is run, producing histograms. Now for each bin of those N histograms, a quadratic- or cubic interpolation polynomial is fit.

A general χ^2 can be constructed and numerically minimized, resulting in an optimized parameter value.

$$\chi^2 \approx \sum_{bins} \frac{(\text{interpolation} - \text{data})^2}{\text{error}^2}$$

It is clear that for an increasing number of parameters, this is not an easy procedure. For minimization the Professor 1.2.1 package makes use of the SciPy python library and the Minuit package also included in Root.

4.2 Pdf4mc

Monte Carlo event generators need input parton distribution functions (PDFs). In short, these are used to calculate the input kinematics for matrix elements, in the steering of initial state radiation and in the calculation of multiple parton interactions. Today a multitude of different PDF sets, calculated at different orders and with different properties, are available for general use. An often surfacing question is which PDF set should be used with a certain event generator. In the first place this question concerns the order of the PDF set and the generator. A common choice is the use of next-to-leading-order (NLO) PDF sets with NLO generators and leading-order (LO) PDF sets with LO generators. However this choice is not the only possibility, and especially in the LO generator case there is room for improvement. One can consider the use of NLO PDF sets with LO generators, which is not without options, or look at another approach altogether, the use of PDFs optimized for use with a specific generator. In the following we will study the latter method.

What is the merit of tailoring PDFs to event generators and how does it work? To answer that we first consider two problems with LO PDFs.

- **Intrinsic limitations**: Conventional LO PDFs are determined using existing data. When moving to energies above the ones used in the calculation, or considering physical processes not included therein, the result becomes unreliable. There will be differences in shape and magnitude of distributions acquired through event generation using either LO or NLO PDF sets, mainly due to abnormalities in the LO PDFs at extreme x-values. However, higher energies and new processes are precisely what one is dealing with when making LHC predictions. To do away with this unreliability, the PDFs need to be improved.

- **Initial state radiation**: To begin with, different event generators handle initial state radiation (ISR) in different ways. Next, LO PDFs generally only include collinear gluon radiation, whereas ISR in Monte Carlos means gluon radiation at finite angles. Again this will result in differences in the shape of distributions when comparing calculations with and without ISR (both with conventional PDFs). Matching the PDFs with the event generator can improve the results. Also note that in comparison with the first problem, the ISR effect is the least significant.

Now how to improve? It was mentioned that the conventional LO PDFs, which need to be improved, use existing data. If one uses event generators, often for predictions at higher energies, one wants to use PDFs that produce predictions that come as close to reality as possible. Yet, as we are talking about predictions, how can we know which one matches the unknown reality best? It's here that NLO calculations come in. By definition we take them to be a reliable alternative for nature. The idea then is to optimize the PDFs for use

with an event generator by calculating them using, not only the existing experimental data, but also good estimates of future data. These latter data sets will need to be generated. This addition of generated data in the global PDF fit ensures better predictions, whereas the continued inclusion of existing experimental data ensures the match with nature at energies we know today. Note that in the calculations done for this work, we did not use generated data in a higher energy range, but only generated data in the same energy range as the experimental data.

A final observation to make is that, using LO generators and thus LO matrix elements, the limits of these LO matrix elements need to be kept in mind. LO means rather harsh approximations here. It is unrealistic to expect good matches with (pseudo-)data in both the low- and higher energy ranges. The actual result remains a compromise, attempting to improve the conventional PDFs. For more details and examples of differences and improvements we refer to [11].

More concrete, we will perform Pdf4mc using HERA data, optimizing PDFs for the Pythia 6.4 event generator. A next step can then be to use Pythia 6.4 to generate LHC data with these PDFs and then study the underlying event.

4.3 Multi-threading

4.3.1 Introduction

Most current PCs are multicore machines. This means that, when one is executing a CPU-time consuming task, a huge gain can be achieved by using multithreading. To be able to do that, the code has to be edited to allow parallel processing. The different threads can then be distributed over the different cores of the CPU, and the task will finish in a greatly reduced period of time. Of course some parts of a program have to be run consecutively, and the complete task cannot just be threaded randomly. Also note that this section pertains to the threading of C++ code only.

To define structure in the threading, we make use of the OpenMP API and libraries [28, 29]. The system works with "pragmas". Sections that can be executed in parallel are marked with

```
#pragma omp parallel
```

or

```
#pragma omp sections
```

These can be structures such as for- or do-loops, which allow real parallelization, or blocks of sequential code which are unrelated and can thus also be run in parallel.

More specifically, for-loops would be put in the `#pragma omp parallel` environment and be marked with their own more specific pragma:

```
#pragma omp for schedule(dynamic,chunk) nowait
```

The `chunk` variable sets the amount of work that will be allocated to each thread to be executed in one block. For example, 1000 iterations with 250 chunk size on a quad core machine means that each thread will only receive a task once, to do 250 iterations. For a

smaller chunk size, after the first round is done, the next chunk will be allocated to the first free thread until the whole operation is finished. Here the `dynamic` variable is active too. This allows allocation to the first free thread, as opposed to `static` allocation where threads are always called upon in the same sequence.

Sequential code blocks can be put in a `#pragma omp parallel` environment, or in a separate `#pragma omp sections` environment. They should be marked with

```
#pragma omp section
```

which will ensure their parallel execution.

Also important is the declaration of variables. It is necessary to differentiate between `shared` and `private` variables, such that for example thread one would not alter the loop-index used by thread two and vice versa. This index would be a private variable. On the other hand for example each thread could add to the same sum variable, which should then be a shared variable. Shared and private variables should be marked for each omp for/do- or section-code block with `shared(...,...)  private(...,...)`.

Finally, properties such as the number of threads can be set by editing their matching variables in the omp header, in this case `num_threads(...)`, or by calling their matching set-functions (e.g. `omp_set_num_threads()`). In the same way, these properties can be used in the parallelized code blocks by calling their matching get-functions (e.g. `omp_get _num_threads()`).

4.3.2 Observations and pitfalls

Altogether, it is clear that parallelizing code requires some rewriting. However, this is a small effort if one considers the speedup gained from utilizing multiple CPU cores instead of one. Of course one does not gain exactly a factor four (or another number of threads, depending on the machine), but a bit less. There are always parts of code that cannot be parallelized and also the threading takes some resources. The splitting (at the beginning) and synchronizing (at the end) of the threads is not costless. This might even be so extreme that with less time consuming processes, multithreading would not improve the efficiency. One should therefore always first understand the parallellizability of a piece of code before actually rewriting, so as not to waste time changing it into something that is not faster, or even slower.

On a different note, one should watch out with more complex objects when they are being used in the parallelized code. Often it is possible to define a shared variable, yet a lot easier to simple define multiple private variables and later on merge them. This can be done for example with a Pythia 8.1 object. If definitions are not made properly and different threads start calling `next()` on the object at the same time, things will go seriously wrong. An easy solution here is to utilize multiple Pythia 8.1 objects. Each thread will then have its own and results can be merged at the end. The main thing to watch out for with this solution is the initialization and the use of random numbers. Initialization should happen such that it is certain that one generates different sets and not multilple identical sets because the random number seeds were set the same for each object. An example that shows how reproducability can be tricky. Yes, seeds are handy, but set them wisely. Note that all these observations can also be made for Root objects and one should watch out especially with the naming and calling of files and histograms.

A third factor, maybe even more important, is the use of external libraries. It is clear that is is only possible to call library functions directly in a threaded structure if the library is built to allow multi-threading. If you try it with one that isn't, segmentation fault errors are an unwanted certainty. Luckily, the multi-threading incompatibility of some libraries doesn't mean one can't still use multi-threading. The OpenMP library includes a pragma to mark statements that should be handled as if there was no multi-threading on or about:

```
#pragma omp critical
```

We can conclude that in many cases there are multiple opportunities to economize on time use, which would be unwise to ignore. Thanks to the customizability with different types of pragmas and the possibility to uphold single-thread statements within multi-threading code blocks, there are few situations where one can't at least achieve some gain. It is up to the user to decide whether or not this gain justifies the extra work.

Part II - Parton Density Functions

5 Parton Density Functions with Pythia

In the following section we will give a detailed description of event generation and processing in Pythia 8.1. The scenario for Pythia 6.4 is entirely analogue, even though written in fortran instead of C++. The important thing to know are the respective names of the parameters than need to be set for the event generation. We will summarize these for both versions in a table at the end of section 5.1.

5.1 Setting up event generation in Pythia

The setup starts with creating a Pythia 8.1 object and initialising it. Initialization parameters can either be set seperately in the main program, or put in a *.cmnd*-file and then set with one call. When using the same analysis program with different event settings, it is more efficient to refer the parameters to a seperate file. It is then unnecessary to recompile the program when switching to another mode. The setup takes only three commands, shown below. The actual parameters in the *.cmnd*-file require a bit more explanation and are discussed in paragraph 5.1.1.

```
#include "Pythia.h"
using namespace Pythia8;

int main(int argc, char *argv[]) {
    Pythia p;
    p.readFile(argv[0]);
    p.init();

    ...
}
```

5.1.1 Pythia 8.1 initialization file

Even without differentiating between the event types we wish to generate, there are some global parameters to be set. We work with *pp*-collisions, at a centre of mass energy of 14 TeV and with initial state radiation but without final state radiation. Multiple interactions are switched of too. The validity of ignoring final state radiation will be discussed later in section 5.2.2. Another main parameter is the number of events. That number will be changed from time to time depending on the process under study, but it allways has to be included in the initialization file. Note that the comment character for *.cmnd*-files is the exclamation mark. All this results in the following skeleton file:

init.cmnd

```
Beams:idA = 2212                ! first incoming beam is a 2212, i.e. a proton
Beams:idB = 2212                ! second beam is also a proton
Beams:eCM = 14000.              ! the cm energy of collisions
PartonLevel:ISR = on            ! initial state radiation, yes
PartonLevel:FSR = off           ! final state radiation, no
PartonLevel:MI = off            ! multiple interaction, no
Main:numberOfEvents = 1000000   ! one million events
```

Random numbers play an important role in the event generation. Therefore it is advisable to take care setting the seeds for the random number generator. This means for example to set a different seed for each thread when using multi-threading so as not to add together four identical sets of events, as explained in section 4.3. We add two more lines to the initialization file:

init.cmnd

```
8  Random:setSeed = on              ! seeded random generation
9  Random:seed = 1                  ! 0: time based, 1-900.000.000: unique sequences
```

Also crucial is the PDFset. When comparing different methods one wants to compare results for the same PDFset. The default PDFset for Pythia 8.1 is *CTEQ5L*, but it is very easy to load either other built-in sets, or independent Les Houches Accord PDFsets. There are many different options here, but only one should be used at a time, hence the extra commenting in the code.

init.cmnd

```
10 PDF:pSet = 2                           ! the default Pythia PDFset, CTEQ5L
11 ! PDF:useLHAPDF = 0                    ! 0:don't use LHAPDFsets, 1:use LHAPDFsets
12 ! PDF:LHAPDFset = MRST2004FF4lo.LHgrid ! example of a LHAPDFset
```

Finally one needs to specify which type of events needs to be generated. To study quark distributions we look at $qq \to Z$, to study gluon distributions we look at $gg \to H$. For the former process, the default includes also photon production, which we don't want. Therefore we add another flag to the initialization file, setting the generator to Z-only mode.

init.cmnd

```
13 WeakZ0:gmZmode = 2                     ! Only Z0, no gamma
14 WeakSingleBoson:ffbar2gmZ = on         ! qq->Z
15 ! HiggsSM:gg2H = on                    ! gg->H
```

5.1.2 Using Root to store and represent data

To use Root elements it is necessary to link to the appropriate libraries in the program. Basics such as *TFile.h*, *TH1.h* and *TH2.h*, *THnSparse.h* and *TMath.h* would allow creating Root histograms and writing to files. To be able to also show the results at runtime without seperately having to start a Root instance, we need *TApplication.h* and *TCanvas.h*. It suffices to create one TApplication at the beginning of the main program to be able to open and close TCanvases during the rest of the program:

```
1  TApplication *rootapp = new TApplication("rootapp",0,0);
```

For more about general Root use we refer to the Root User's Guide at [35].

5.1.3 Event generation

For the actual event generation we extract the number of events from the initialization file and create a for-loop with an index running from zero to this number of events. In each loop entry we call the *next()*-function on our Pythia 8.1 object, effectively generating a new event. Once that is done, it's details can be printed, or we can extract properties for

immediate or later analysis. These properties can be available at event level, or they can be more specific, related to a certain particle within a certain event. To access the latter information, we extract the event size and create a second for-loop. A description of the properties we studied and how we got to them, will follow in the next section (5.2).

```
int nEvent = p.mode("Main:numberOfEvents");

for(int i=0; i < nEvent; i++) {
    if(!p.next()) continue;
    p.next();
    p.event.list();

    sEvent = p.event.size();
    for(int j=0;j < sEvent; j++) {
        ...
    }

    ...
}
```

5.1.4 Summary of parameters

Pythia 8.1	value set	Pythia 6.4	value set	explanation
Beams:idA	2212	PYINIT(‘’,‘’,‘e+’,‘’,‘’)	e^+	proton/electron beam
Beams:idB	2212	PYINIT(‘’,‘’,‘’,‘p’,‘’)	p	proton beam
Beams:eCM	14000	PYINIT(‘’,‘’,‘’,‘’,318D0)	318	centre of mass collision energy (GeV)
PartonLevel:ISR	on	MSTP(61)	1	initial state radiation
PartonLevel:FSR	off	MSTP(71)	0	final state radiation
PartonLevel:MI	off	MSTP(81)	0	multiple parton-parton interaction
		MSTU(101)	0	fixed α_{em}
Main:numberOfEvents	variable	NEV	variable	number of events
Random:setSeed	on			seeded random number generation
Random:seed	variable	MRPY(1)	variable	random number seed
PDF:pSet	2			default value: CTEQ5L set
PDF:useLHAPDF	0/1	MSTP(52)	2	use LHAPDF / don’t use LHAPDF
PDF:LHAPDFset	variable (name)	MSTP(51)	variable (number)	LHAPDF PDF set name
WeakSingleBoson: ffbar2gmZ	on			process choice: $qq \to Z$
WeakZ0:gmZmode	2			Z^0 only, no photons
HiggsSM:gg2H	on			process choice: $gg \to H$
		MSEL	0	user set process
		MSUB(10)	1	process choice: ep DIS

5.2 Extracting properties from Pythia 8.1 events

5.2.1 Z-Boson mass

The mass of the Z-Bosons generated within Pythia 8.1 follows a Breit-Wigner distribution. For later use in the calculation of the $qq \to Z$ cross section we need the parameters of this distribution. Of course these should match the real Z-Boson properties, but to check the generated data and to be able to use the Breit-Wigner curve best matching the dataset, we will fit the Z-Boson mass histogram we built during the event generation and use those fit parameters in later calculations.

First we parametrize the Breit-Wigner distribution with mass M_Z, width Γ and scale constant α:

$$BW(Q) = \frac{\alpha}{4} \cdot \frac{\Gamma^2}{(Q - M_Z)^2 + \left(\frac{\Gamma}{4}\right)^2}$$

Therein $\hat{s} = Q^2$ and using $\sqrt{\hat{s}} = Q \approx M_Z$ we can see this matches the relativistic invariant form (also mentioned in [31]):

$$BW(Q) = \frac{\alpha}{4} \cdot \frac{\Gamma^2 \left(Q + M_Z\right)^2}{\left[\left(Q^2 - M_Z^2\right)^2 + \left(\frac{\Gamma}{4}\right)^2\right] \left(Q + M_Z\right)^2} \approx \alpha \cdot \frac{\Gamma^2 M_Z^2}{\left(Q^2 - M_Z^2\right)^2 + \Gamma^2 M_Z^2}$$

Mass is an easily accesible property within the Pythia 8.1 *Particle*-class.

```
1 p.event[i].m();
```

We fit the histogram in the range 86.0-96.0 GeV/c^2 with the parametrized Breit-Wigner function. Before running the fit, we also add initial values and limits for the parameters. The Z-mass is parameter [0], the Z-width parameter [1]. Scale parameter [2] is fixed by retrieving the top value from the normalized histogram. The result is shown in figure 18.

```
1 TF1 *myBW = new TF1("myBW","[2]/4.*[1]*[1]/((x-[0])*(x-[0])+([1]/2.)*([1]/2.))",86.,96.);
2 int topbin = hZmass->GetMaximumBin();
3 double top = hZmass->GetBinContent(topbin);
4 myBW->SetParameter(0,91.2);
5 myBW->SetParameter(1,2.50);
6 myBW->FixParameter(2,top);
7 myBW->SetParLimits(0,90.0,92.0);
8 myBW->SetParLimits(1,2.4,2.6);
9 myBW->SetRange(84.,98.);        // drawing range
```

For the parameters we find:

- $M_Z = 91.1 \pm 1.6$ GeV/c^2
- $\Gamma = 2.50 \pm 0.14$ GeV
- $\alpha = 0.0144$

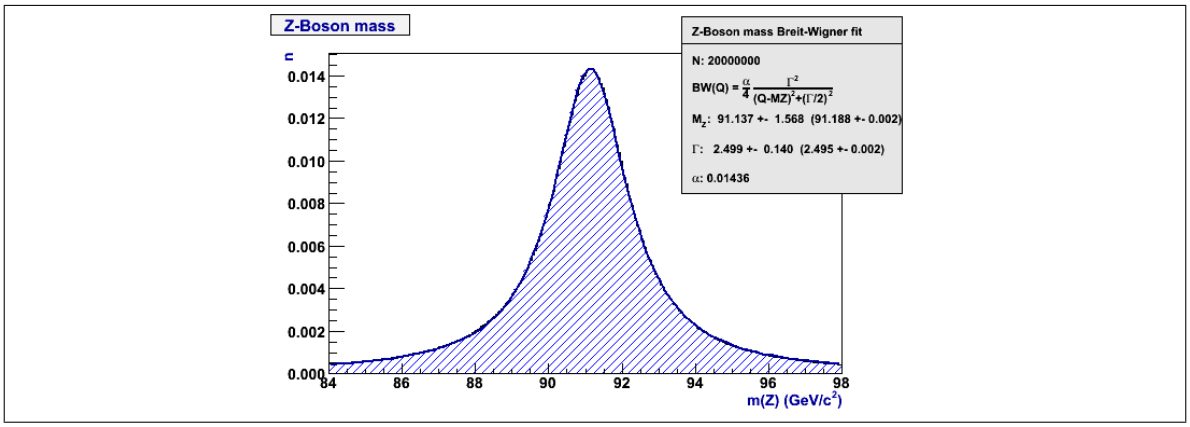


Figure 18: Z-Boson mass distribution with fitted Breit-Wigner curve

5.2.2 Z-Boson transverse momentum

Both protons of the generated event have only longitudinal momentum. Since we include initial state radiation, the quarks actually annihilating to a Z-Boson do have both longitudinal and transverse momentum. PYTHIA 8.1 does not include the correct value of this transverse momentum as a property, so we have to do some bookkeeping ourselves. The values we want can be retrieved by tracking all radiated particles and then subtracting their momentum from the quark momentum before radiation. The remainder is the annihilating quark momentum. As a check we can make a balance of the quark transverse momentum with the the Z-Boson transverse momentum, which is available as a PYTHIA 8.1 property. Of course the same balance can also be made for the longitudinal momentum and the energy. In this section we show only the Z-Boson transverse momentum, the rest of the process will be discussed in section 5.2.7.

The Z-Boson transverse momentum can be accessed through the PYTHIA 8.1 *Particle*-class.

```
1 p.event[i].pT();
```

Naturally, i should be the index for the Z-Boson within the event. Actually, this function is available for all particles in the event and it looks as though we could simply use it for the parton transverse momentum too. However, due to the way PYTHIA 8.1 handles initial state radiation, the transverse momentum of the partons is listed as being zero and we have to use a workaround. The Z-Boson transverse momentum is shown in figure 19.

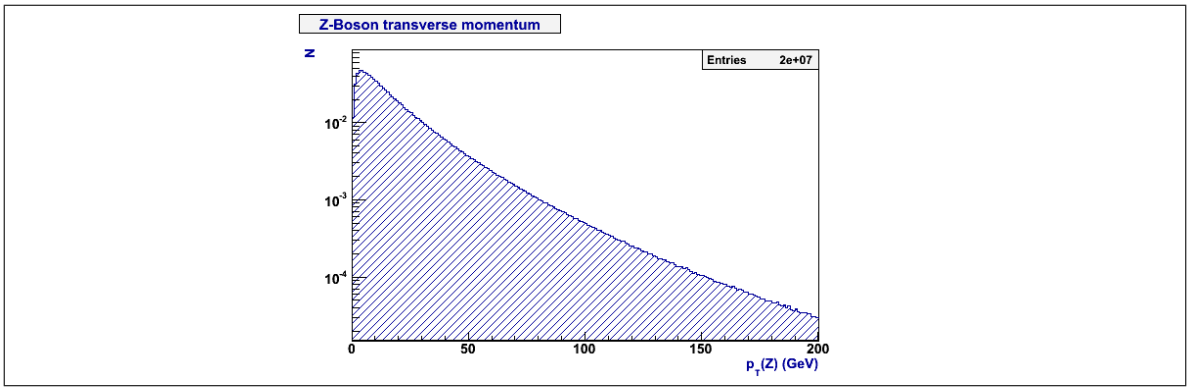
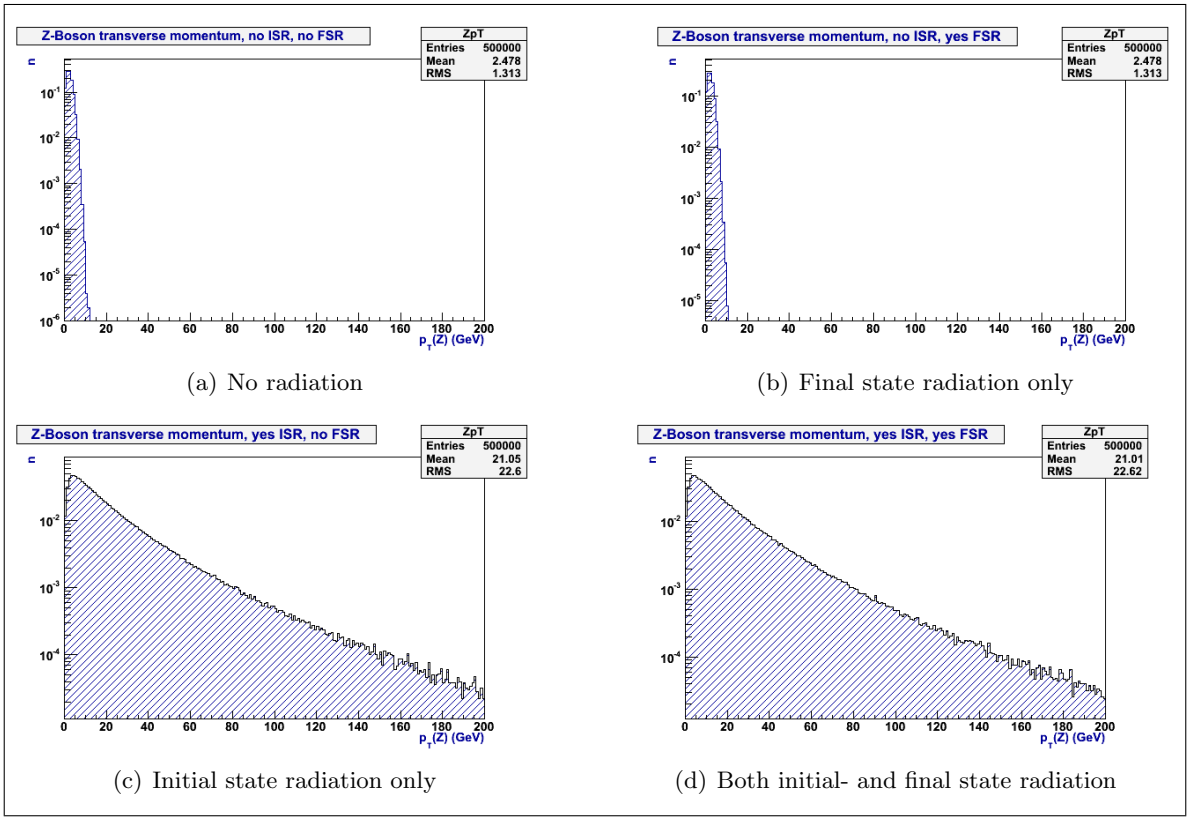


Figure 19: Z-Boson transverse momentum distribution, events including initial state radiation and excluding final state radiation, as discussed in section 5.1.1

Below the same plot for different initial- and final state radiation settings. It is clear that initial state radiation is responsible for the transverse momentum tail; ISR must remain switched on in the initialization file. Furthermore, since we are looking at initial state- and resonance properties, it is not unexpected that final state radiation doesn't generate visible differences. We can safely switch of FSR in the initialization file, which is lucky because it speeds up event generation. All four different settings combinations are shown in figure 20



(a) No radiation

(b) Final state radiation only

(c) Initial state radiation only

(d) Both initial- and final state radiation

Figure 20: Z-Boson transverse momentum distribution for different radiation settings

5.2.3 H-Boson mass

The H-Boson mass distribution has a very small width. We repeat the fit procedure used for the Z-Boson mass distribution in section 5.2.1. The initial parameter values and limits are changed. The result is shown in figure 21.

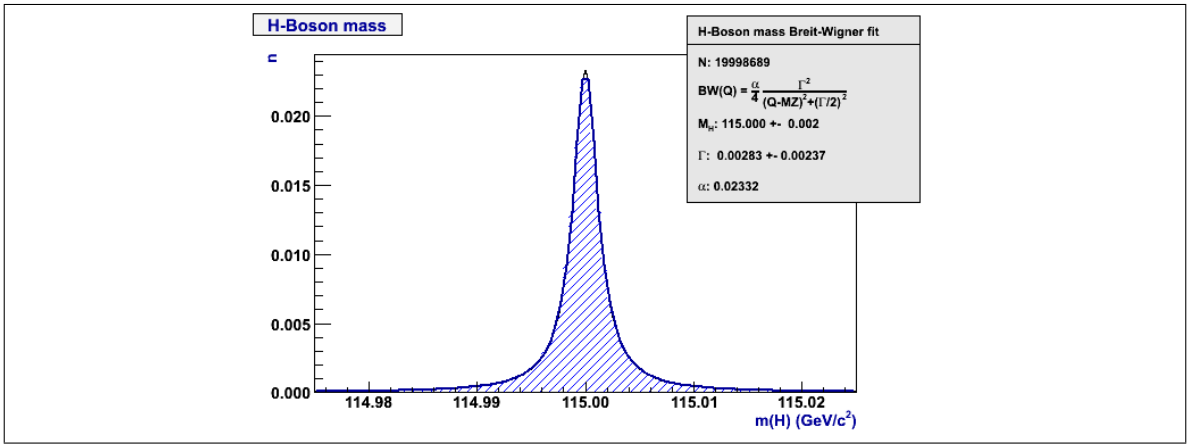


Figure 21: H-Boson mass distribution with fitted Breit-Wigner curve

For the parameters we find:

- $M_H = 115.000 \pm 0.002$ GeV/c^2
- $\Gamma = 0.0028 \pm 0.0024$ GeV
- $\alpha = 0.0233$

The M_H value is important for the calculation of the $gg \rightarrow H$ cross section. According to [3], if we assume $gg \rightarrow H$ via a top loop and $m_t \geq M_H$, we can use a simplified formula for the cross section. With $M_H = 115.0$ GeV/c^2 this is certainly the case ($m_t = 172.0 \pm 2.2$ GeV/c^2).

5.2.4 H-Boson transverse momentum

Entirely identical to the Z-Boson case, we need the H-Boson transverse momentum to determine the gluon transverse momentum. We show the H-Boson transverse momentum distribution in figure 22. It is clear that the tail is larger than in the case of Z-Boson production.

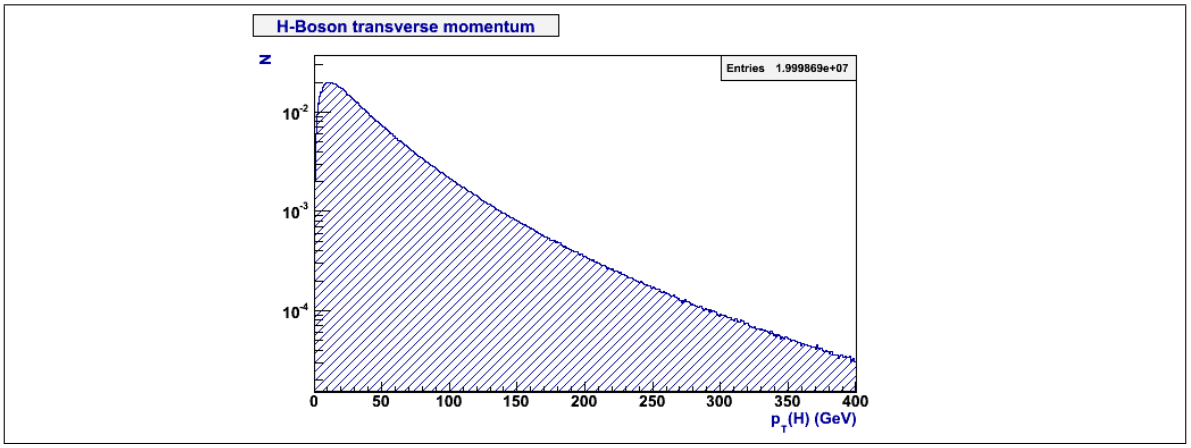


Figure 22: H-Boson transverse momentum distribution

5.2.5 The $qq \to Z$ cross section

We start from the cross section discussed in lecture 5 of [25]:

$$\hat{\sigma}_{qq\to Z}(Q) = \frac{8\pi}{3}\frac{G_F M_Z^2}{\sqrt{2}}\left(g_a^2 + g_v^2\right)\cdot\delta\left(Q^2 - M_Z^2\right)$$

However, our Z-Boson mass distribution is not a delta-peak, but a Breit-Wigner distribution. We discussed fitting this distribution in section 5.2.1. We substitute the delta-distribution in the cross section with a Breit-Wigner distribution and we use the parameters found in the fit.

$$\begin{aligned}\hat{\sigma}_{qq\to Z}(Q) &= \frac{8\pi}{3}\frac{G_F M_Z^2}{\sqrt{2}}\left(g_a^2 + g_v^2\right)\cdot BW\left(Q - M_Z\right)\\ &= \frac{8\pi}{3}\frac{G_F M_Z^2}{\sqrt{2}}\left(g_a^2 + g_v^2\right)\cdot\frac{\alpha}{4}\frac{\Gamma^2}{(Q-M_Z)^2 + \left(\frac{\Gamma}{2}\right)^2}\end{aligned}$$

5.2.6 The $gg \to H$ cross section

We start from the $ab \to H$ cross section discussed in chapter 4 of [3], equation (4.90). Because we have $m_t \geq M_H$ and are interested in gluons and not composite particles, we may simplify the equation and find:

$$\hat{\sigma}_{gg\to H}(Q) = \frac{G_F\pi}{32\sqrt{2}}\left(\frac{\alpha_s}{\pi}\right)^2\left(\frac{1}{3}\right)^2$$

So far, this cross section is constant. However, again we are dealing with a Breit-Wigner distributed mass spectrum and not a delta-peak. We therefore add a Breit-Wigner factor to the cross section, with the parameters found in section 5.2.3.

$$\hat{\sigma}_{gg\to H}(Q) = \frac{G_F\pi}{32\sqrt{2}}\left(\frac{\alpha_s}{\pi}\right)^2\left(\frac{1}{3}\right)^2\cdot\frac{\alpha}{4}\frac{\Gamma^2}{(Q-M_H)^2 + \left(\frac{\Gamma}{2}\right)^2}$$

5.2.7 Parton momentum and energy

As mentioned briefly in the section about the Z-Boson transverse momentum (5.2.2), this step requires some bookkeeping. We need to track the particles created through initial state radiation and make a momentum or energy balance at the level of the final Z-Boson. This means that if, for example, it takes six steps for the generated Z-Boson to evolve to its final state (the one that decays), we also have to track back six steps in all directions starting from both annihiliting quarks.

Looking back is not difficult, one just looks at the PYTHIA 8.1 information about the mother. However, in doing so, we may not forget to look for side-tracks too. This means that for each step we go back (by reading the mother information), we also need to look in the forward direction starting from that mother particle to see whether this particle produced only the particle which we started from, or also another one to the side. This can be done by reading the daughter information.

This procedure may look quite daunting when explained like that, but by creating two simple functions which can be called recursively, it can be executed swiftly and automatically. We shall start by documenting those two functions and then provide a concrete example of the procedure. In figure 23 we show a schematic view of an event, which can be used to follow the steps of the procedure. We get back to the figure at the end of the description.

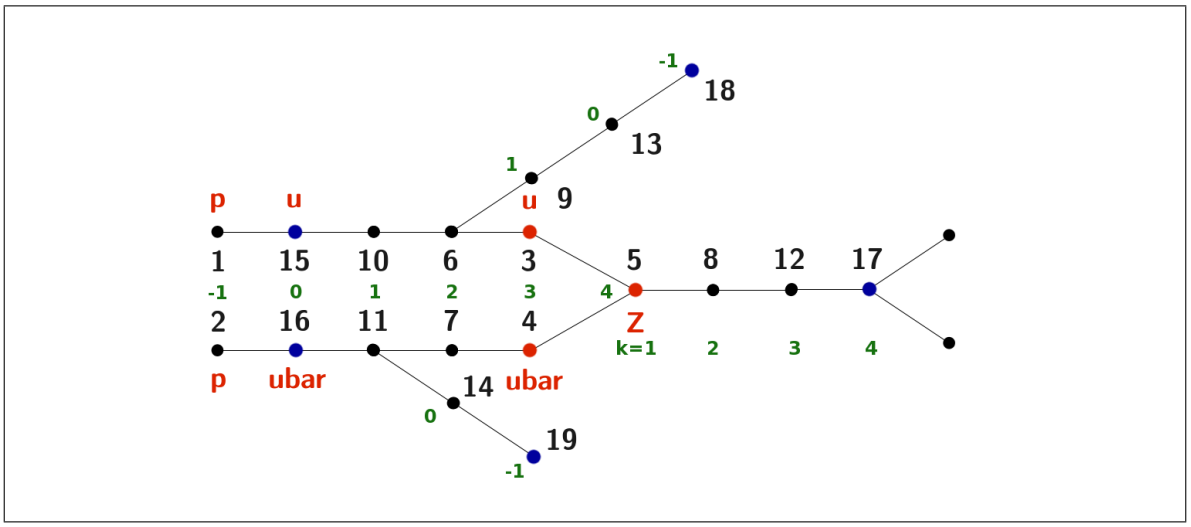


Figure 23: Schematic overview of an event

Everything starts by counting the number of Z-Boson evolutions in the event. This is a question of plainly looping over all entries and adding one to a counter when we come across a Z-Boson. One could remark that in this way it is possible to confuse two different Z-Boson evolution tracks within one event because we don't look at heritage, and that we should track the evolution through mother- and daughter information here too. However, while the former is true, following a more complicated Z-Boson tracking procedure is rendered unnecessary by the fact that there are no events with more than one Z-Boson track. Thus, we obtain a Z-Boson evolution count at the end of the loop, also shown in figure 23.

Now for those two functions. Function f_1 is intended to look back (from the quarks in the middle to the left in figure 23). The first step is to gather the information about the mothers, and to use it to decide which step to take next. At the same time we can also subtract one from the Z-Boson evolution counter (k). When the counter reaches zero, we have tracked back far enough and we can gather our momentum values. In the main program the tracking procedure starts by calling this function once for each of the two annihilating quarks. They are always numbered three and four.

```
1 f1(Pythia &p, int n, int k, double &tx, double &ty, double &tz, double &E,
2       double &tx0, double &ty0, double &tz0, double &E0) {
3     int m1=0, m2=0;
4     m1 = p.event[n].mother1();
5     m2 = p.event[n].mother2();
6     k-=1;
7     ...
```

We are alerted to side tracks by the appearance of id-number 0, the system. With simple if-statements we differentiate between chains with ($m1 = m2$) or without side tracks. In the latter case we continue (f_1), in the former case we study the side track ($m \neq 0$, f_2) and continue too (f_1). In any case, this tracking continues only as long as the Z-Boson evolution counter (k) has not reached zero. Along the main chain one of the initial colliding protons always lies at k=-1. Note also that it is important to pass k by value and not by reference. Throughout the recursive function calling, a side tracks should not be influenced by what is happening for k in another side track from the same or a different main chain.

```
6     ...
7     if(m1 == m2 && n < p.event.size() && m1!=0 && k>=0) {
8         f1(p,m1,k,tx,ty,tz,E,tx0,ty0,tz0,E0);
9     }
10    else if(n < p.event.size() && k>=1) {
11        if(m1!=0 && m1!=3 && m1!=4) {
12            f2(p,m1,k,tx,ty,tz,E,tx0,ty0,tz0,E0);
13            f1(p,m1,k,tx,ty,tz,E,tx0,ty0,tz0,E0);
14        }
15        if(m2!=0 && m2!=3 && m2!=4) {
16            f2(p,m2,k,tx,ty,tz,E,tx0,ty0,tz0,E0);
17            f1(p,m2,k,tx,ty,tz,E,tx0,ty0,tz0,E0);
18        }
19    }
20    ...
```

When we do reach Z-Boson evolution count zero, we save the momentum and energy values of the particle. Because we need the value for the initial quark contribution in positive, and all the other particle contributions in negative, this sum can be a bit tricky. However, we know the only contributing result of the backtracking function f_1 will be the initial quark. We can add the contribution to the sum. It's the other function, looking forward, that will track the particles adding negatively to the sum.

```
19    ...
20    if(k==0) {
21        tx0 = p.event[n].px(); tx += tx0;
22        ty0 = p.event[n].py(); ty += ty0;
23        tz0 = p.event[n].pz(); tz += tz0;
24        E0  = p.event[n].e(); E  += E0;
25    }
26}
```

Now let's get to the details of function f_2. As said, this one is meant to look forward, in the side tracks we found with function f_1. Analoguous to what we did with the mothers, we now gather the information about the daughters and use it to determine the next step. Also, with each consecutive function call, the Z-Boson evolution count should again be diminished by one. We study the side track in forward direction through a recursive calling of f_2. The call happens only once in each step since we know from manual checks that $d1 = d2$, there are no side chains to side chains.

```
void f2(Pythia &p, int n, int k, double &tx, double &ty, double &tz, double &E,
            double &tx0, double &ty0, double &tz0, double &E0) {
    int d1=0, d2=0;
    d1 = p.event[n].daughter1();
    d2 = p.event[n].daughter2();

    k-=1;
    if(d1 != d2 && n < p.event.size() && k>=0) {
        if(d1!=0 && d1!=3 && d1!=4) {
            f2(p,d1,k,tx,ty,tz,E,tx0,ty0,tz0,E0);
        }
    }
    else if(d1 == d2 && k>=0) {
        f2(p,d1,k,tx,ty,tz,E,tx0,ty0,tz0,E0);
    }
    ...
```

Finally, we reach Z-Boson evolution count zero again, this time in the forward direction. We save the particle properties in the same way we did when backtracking, only this time they contribute negatively. Endpoint $k = -1$ instead of $k = 0$ is a technical imperfection.

```
    ...
    if(k==-1) {
        tx0 = p.event[n].px(); tx -= tx0;
        ty0 = p.event[n].py(); ty -= ty0;
        tz0 = p.event[n].pz(); tz -= tz0;
        E0  = p.event[n].e(); E  -= E0;
    }
}
```

To end this section we print the results of the procedure for a random event. The same event is again depicted in figure 24. First the properties of the Z-Boson are given, next both quark tracks are examined with functions f_1 and f_2. It is clear from the results that the momentum- and energy balances are fine: for all four parameters $q_1 + q_2 = Z^0$ (in red).

```
Z-Bosons:    5   8  12  17
Number of Z-Bosons: 4
Final Z-Boson properties: x:  4.954 y: 12.139 z: 100.301 E: 138.810

f1 called on annihilating quark 3
f1:     u   3 has m1:   6 m2:   0 and k: 3
     f2:     u   6 has d1:   9 d2:   3 and k: 2
     f2:     g   9 has d1:  13 d2:  13 and k: 1
     f2:     g  13 has d1:  18 d2:  18 and k: 0
     f2:     g  18 has d1:  42 d2:  42 and k: -1
Particle   18 contributing x:  8.810 y: 14.785 z: -63.359 E: -65.655
f1:     u   6 has m1:  10 m2:  10 and k: 2
f1:     u  10 has m1:  15 m2:  15 and k: 1
f1:     u  15 has m1:   1 m2:   0 and k: 0
Particle   15 contributing x:  0.878 y:  0.140 z: 185.781 E: 185.784
Sums:                      x:  9.688 y: 14.925 z: 122.423 E: 120.129

f1 called on annihilating quark 4
f1:  ubar   4 has m1:   7 m2:   7 and k: 3
f1:  ubar   7 has m1:  11 m2:   0 and k: 2
     f2:  ubar  11 has d1:  14 d2:   7 and k: 1
     f2:     g  14 has d1:  19 d2:  19 and k: 0
     f2:     g  19 has d1:  41 d2:  41 and k: -1
Particle   19 contributing x: -2.250 y: -0.014 z:  -1.213 E:  -2.556
f1:  ubar  11 has m1:  16 m2:  16 and k: 1
f1:  ubar  16 has m1:   2 m2:   0 and k: 0
Particle   16 contributing x: -2.484 y: -2.772 z: -20.909 E:  21.238
Sums:                      x: -4.734 y: -2.786 z: -22.122 E:  18.682
```

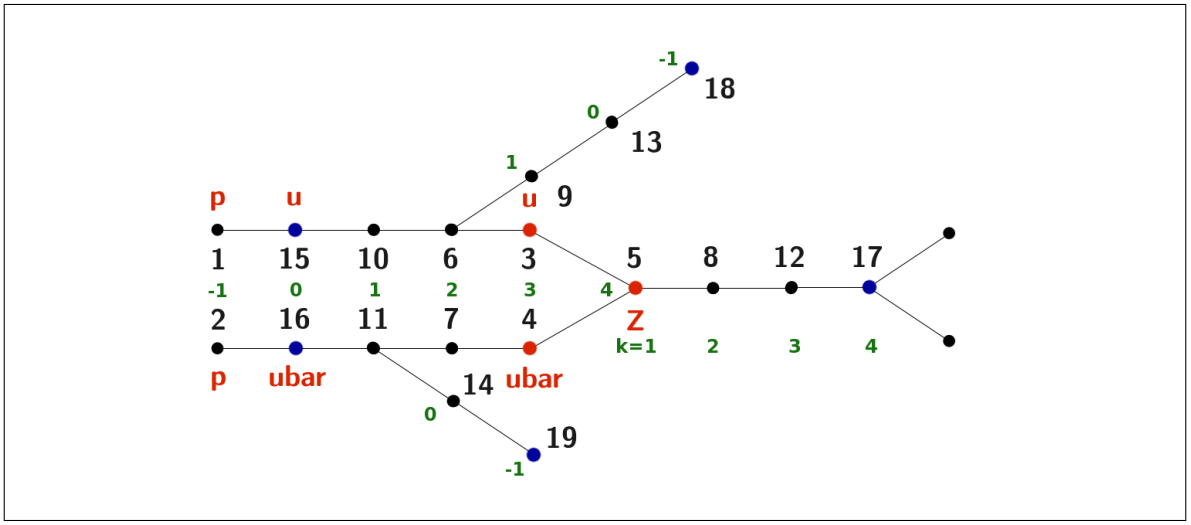


Figure 24: Schematic overview of the discussed event

5.3 Unintegrated Parton Density Functions

Both the studied $qq \to Z^0$ and $gg \to H$ processes are resonances. In the PYTHIA 6.4 manual [31] resonance processes are described in section 7.3. Stated for the cross section is:

$$\sigma = \iint \frac{d\tau}{\tau} dy \;\; x_1 f_1\left(x_1, Q^2\right) \;\; x_2 f_2\left(x_2, Q^2\right) \;\; \hat{\sigma}(\hat{s} = Q^2)$$

We also know the definitions of τ and y:

$$\tau = x_1 x_2 = \frac{\hat{s}}{s}$$
$$y = \frac{1}{2} \ln \frac{x_1}{x_2}$$

If we calculate the Jacobian, we find:

$$d\tau \; dy = dx_1 dx_2$$

So we can also write:

$$\sigma = \iint dx_1 dx_2 \;\; f_1\left(x_1, Q^2\right) \;\; f_2\left(x_2, Q^2\right) \;\; \hat{\sigma}(\hat{s} = Q^2) \tag{4}$$

It is cross section $\hat{\sigma}(\hat{s})$ that contains the physics information about the process. We discussed the cross sections for $qq \to Z^0$ and $gg \to H$ in sections 5.2.5 and 5.2.6. Remember that in both cases we needed to take into account a Breit-Wigner distribution.

If we extract properties of the two annihilating partons and of the event in general from PYTHIA 8.1, we can make histograms. Analoguous to equation (4) we have with Björken-x and $\hat{s}$:

$$N = N(x_1, x_2) = f_1(x_1) f_2(x_2) \; \hat{\sigma}\left(\hat{s}(x_1, x_2)\right)$$

and including the transversal momenta:

$$N = N\left(x_1, p_1^T, x_2, p_2^T\right) = a_1\left(x_1, p_1^T, Q^2\right) \;\; a_2\left(x_2, p_2^T, Q^2\right) \cdot \hat{\sigma}\left(\hat{s}(x_1, p_1^T, x_2, p_2^T)\right)$$

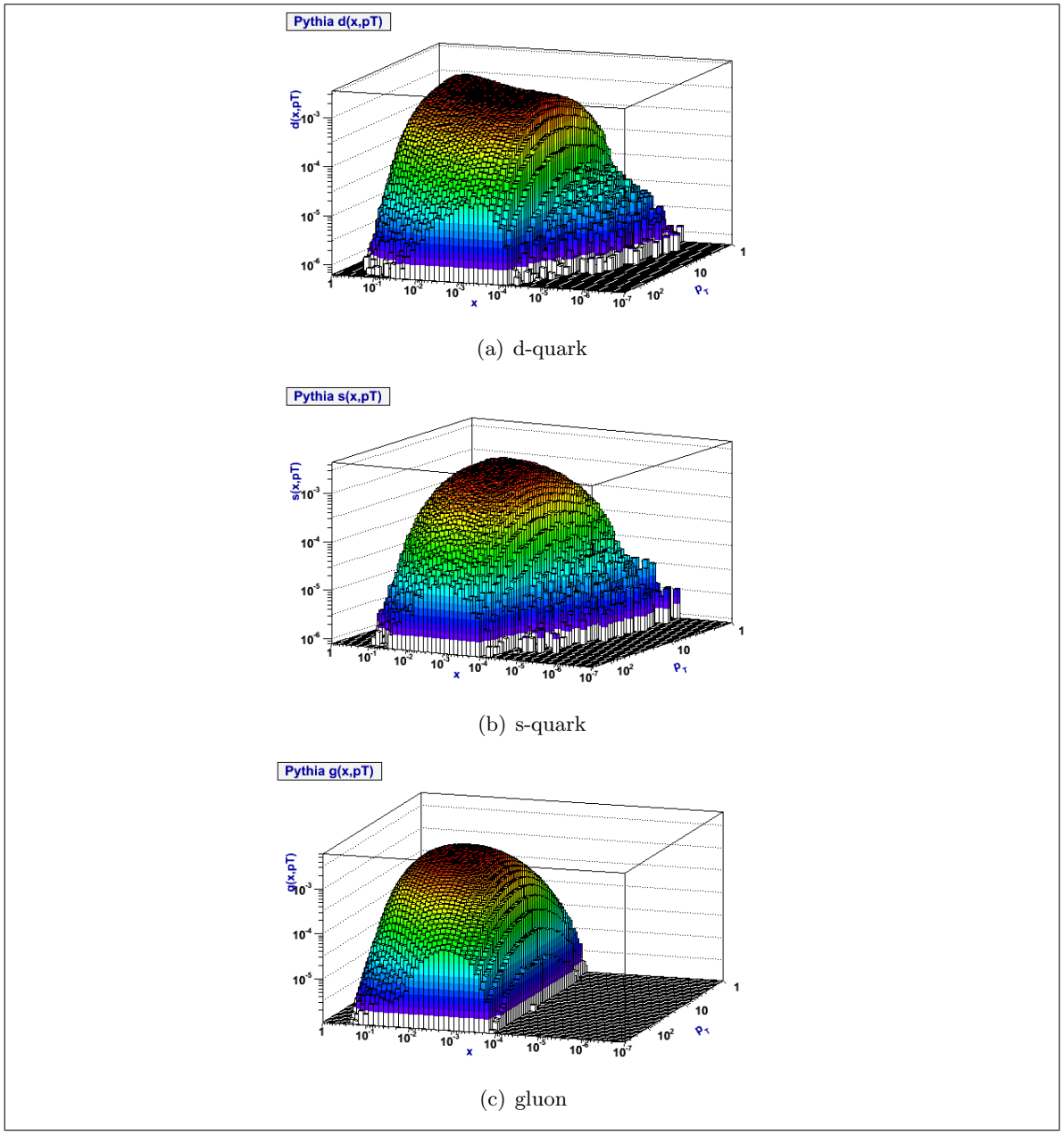
with integrated parton density functions f_i and unintegrated parton density functions a_i. Note that all types of histograms, both $2\text{D}(x_1, x_2)$ and $4\text{D}(x_1, p_1^T, x_2, p_2^T)$, can be created for all quarks (minus the top quark) separately by examining $qq \to Z^0$ events, and for the gluon by examining $gg \to H$ events.

We want to extract the unintegrated parton density functions. With the idea that everything factorizes one could try to get unintegrated parton distribution $a_1(x_1, p_1^T, Q^2)$ by integrating over x_2 and p_2^T:

$$a_1\left(x_1, p_1^T, Q^2\right) = \iint dx_2 dp_2^T \; \frac{N\left(x_1, p_1^T, x_2, p_2^T\right)}{\hat{\sigma}\left(\hat{s}(x_1, p_1^T, x_2, p_2^T)\right)}$$

From doing this and looking at the plots (shown in figure 25) we learned however that this is incorrect and that we are not in fact looking at unintegrated parton density functions. Because of the Breit-Wigner function in $\hat{\sigma}$, possible x_2 values depend on $\frac{M^2}{x_1 \cdot s}$ and taking the integral over x_2 in equation (4) from 0 to 1 is wrong. We will get back to this problem in section 5.4, but only for the integrated parton density functions. Doing the same for the unintegrated parton density functions will require more work.

Below we show the acquired distributions for the d- and the s-quark (a valence- and a sea-quark) and for the gluon. As stated, they are not the unintegrated parton density functions, but still show some of the properties. In the distribution of the d-(valence)-quark we can clearly see the valence quark at $x \approx \frac{1}{3}$. In the distribution of the s-(sea)-quark naturally we can't. The gluon distribution is similar to the sea-quark distributions, but not the same. There are no events in the gluon distribution below a certain x. With $\hat{s} = x_1 x_2 \cdot s$ we have $x_1^{max} \approx 4.2 \cdot 10^{-5}$. In the case of events with Z-Bosons, the width of the mass distribution ensures that we can still find events in a rather wide range around this x. The width of the H-Boson mass distribution however is a lot smaller, making events at lower x highly unlikely. As we got the quark distributions from Z-Boson events and the gluon distribution from H-Boson events, this difference is also visible in the plots. The quark distributions have events below $x = 4.2 \cdot 10^{-5}$, the gluon distribution doesn't.



(a) d-quark

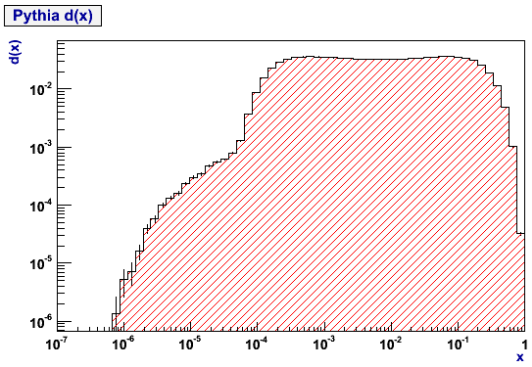
(b) s-quark

(c) gluon

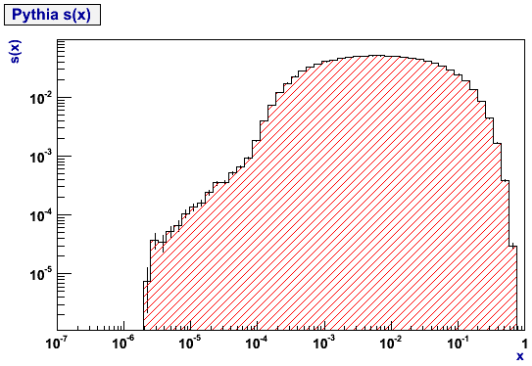
Figure 25: Distributions for the d-quark, s-quark and gluon, extracted from PYTHIA 8.1

5.4 Integrated Parton Density Functions

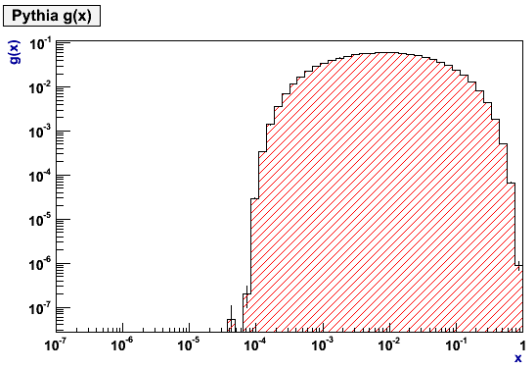
The unintegrated distributions we extracted from PYTHIA 8.1 can be integrated. Again we see the difference between valence- and sea-quarks, and the similarity of sea-quarks and the gluon, but we also see that these distributions do not match the known shape of parton density functions. In fact they still show a product of functions f_1 and f_2, called the parton luminosity function. We explain this below.



(a) d-quark



(b) s-quark



(c) gluon

Figure 26: Integrated distributions for the d-quark, s-quark and gluon, extracted from PYTHIA 8.1, showing the luminosity function

We have a product of f_1 and f_2 in:

$$N = N(x_1, x_2) = f_1(x_1) f_2(x_2)\ \hat{\sigma}\ (\hat{s}(x_1, x_2))$$

If we want the actual functions, further manipulations are required. If $\hat{\sigma}$ were a delta-function, we could integrate:

$$\begin{aligned} \int dx_2\ N(x_1, x_2) &= \int dx_2\ f_1(x_1) f_2(x_2)\ cte \cdot \delta\left(x_2 - \frac{M^2}{x_1 \cdot s}\right) \\ &= f_1(x_1) f_2\left(x_2 = \frac{M^2}{x_1 \cdot s}\right) \end{aligned} \tag{5}$$

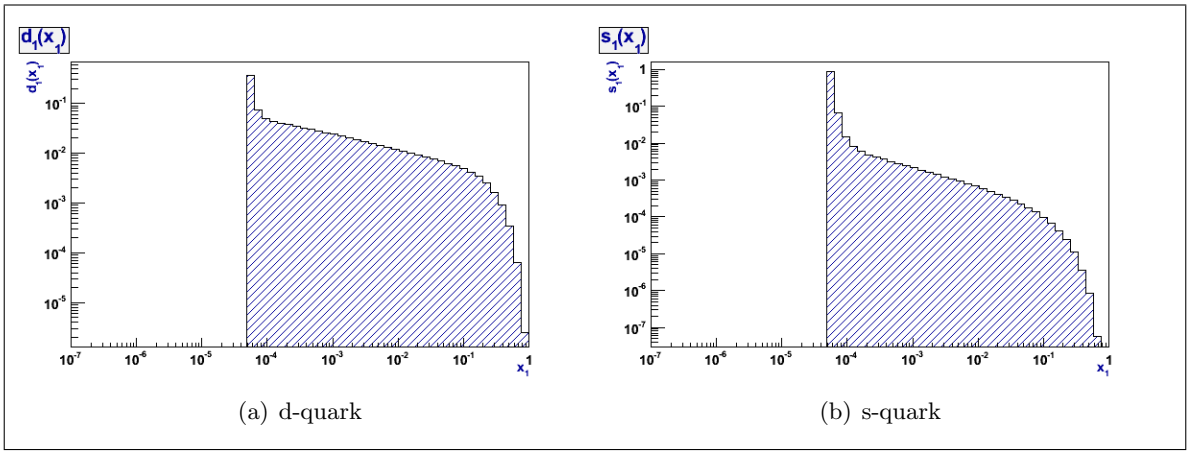
This product is what is called the parton luminosity function. We should note however, that $\hat{\sigma}$ includes a Breit-Wigner distribution and we should write:

$$\int dx_2\ N(x_1, x_2) = f_1(x_1) f_2\left(x_2 = \frac{M^2}{x_1 \cdot s}\right)\ BW(x_1, x_2)$$

Luckily the effect of the Breit-Wigner distribution is small though, and now that we have acknowledged its existence, we will further ignore it. We continue our calculation of $f_1(x_1$. Both f_i's are unknowns, but because we are interested in small x_1 values, we can argue that $f_2(x_2)$ is actually known. Small x_1 matches large x_2 and for large x parton density functions are better known. For our calculations we will get $f_2(x_2)$ from LHAPDF (introduced in section 6) and do:

$$f_1(x_1) = \frac{\int dx_2\ N(x_1, x_2)}{f_2\left(x_2 = \frac{M^2}{x_1 \cdot s}\right)} \tag{6}$$

Both this form (6) and the parton luminosity function from (5) may later be compared to parton distributions and parton luminosity functions from other sources. This will be done in section 9. Below we show the results for both functions for the d-quark, the s-quark and the gluon. The peak at the minimal x in 27 is probably due to the simplified treatment with a delta-function. For improvement the Breit-Wigner distribution needs to be taken into account. For the rest of the range we shall see in later comparisons that the shape is fine.

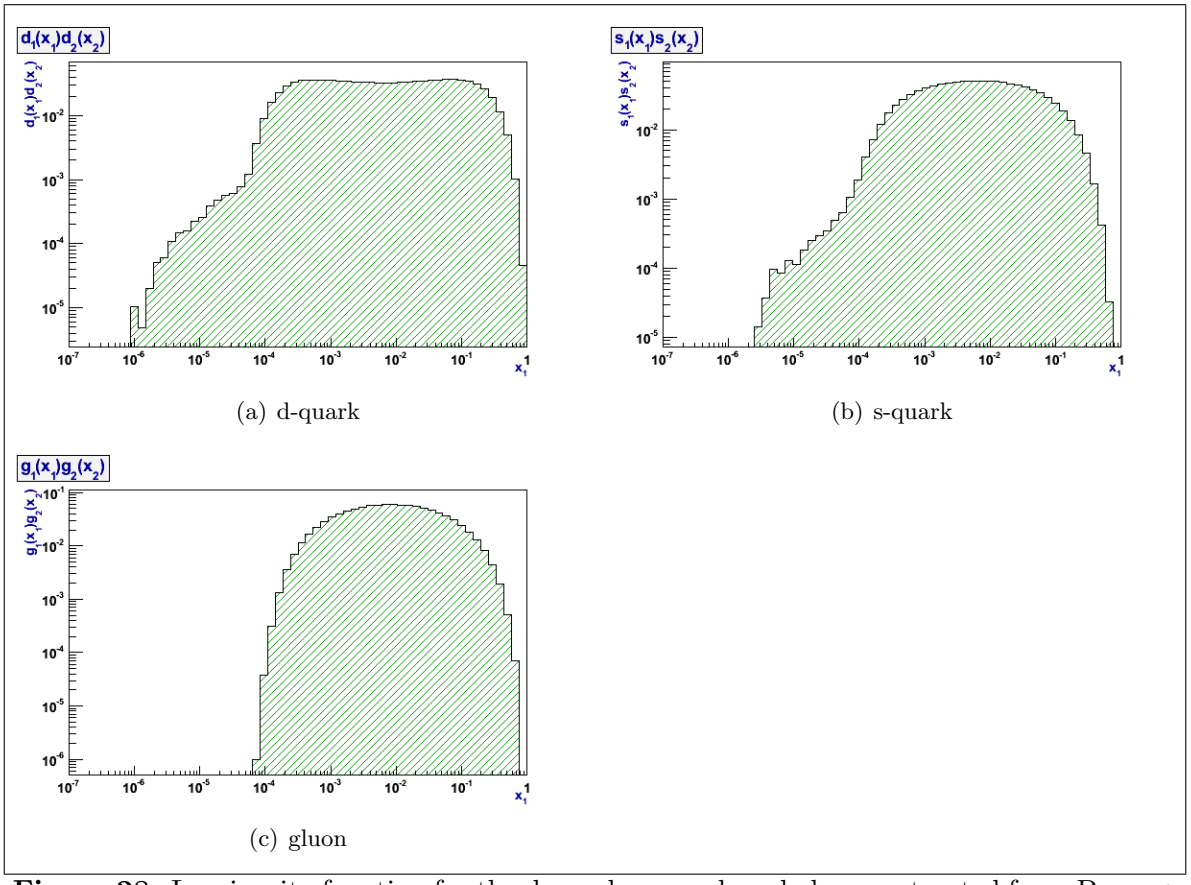


(a) d-quark (b) s-quark

Figure 27: Integrated parton density functions for the d-quark and s-quark, extracted from PYTHIA 8.1 and using LHAPDF

5.5 Parton Luminosty Functions

In figure 28 we see it confirmed that the distribution we got from the integration of the 4D histogram in section 5.3 indeed matches the parton luminosity function, which is what we calculated here. Also, there is again the influence of the Z-mass distribution width on the quark distributions (values below $x = 4.2 \cdot 10^{-5}$). Remedying this would also require a proper treatment of the Breit-Wigner components.



(a) d-quark

(b) s-quark

(c) gluon

Figure 28: Luminosity function for the d-quark, s-quark and gluon, extracted from PYTHIA 8.1

6 LHAPDF Integrated Parton Density Functions

6.1 The LHAPDF interface

The LHAPDF interface is an interface which allows us to use different PDF sets in a uniform manner. Originally it is all Fortran code, but we will use the C++ wrapper class. The interface and its documentation are available through [26]. Everything discussed in this section is relatively straightforward. We don't edit anything concerning the PDF sets, we just access preexisting information. This can be done with two purposes. To begin with we can use the parton density functions when implementing the Kimber-Martin-Ryskin approach to get unintegrated parton density functions (section 7.3) and secondly we can simply use them as they are, integrated, to compare to our integrated results from Pythia 8.1 (section 5.4).

6.2 Integrated Parton Density Functions

To be able to access any PDF set, we need to include the LHAPDF header files:

```
1 #include "LHAPDF/LHAPDF.h"
2 using namespace LHAPDF;
```

and initialize the set using:

```
1 initPDFSet(LNAME,LHGRID,LSUBSET);
```

Therein LNAME is the name of the PDF set, taken from the list of available PDF sets at [26]. This can for example be "cteq5l", the default PDF set in Pythia 8.1. When comparing different results it will be necessary to note which PDF set has been used. The second parameter, LHGRID, defines the way in which the parton density functions are calculated. When using LHGRID, the interface will access the .LHgrid file matching the PDF set's name and the parton density functions will be generated through interpolation within the precalculated tables in the file, following interpolation code also given by the author of the file. An alternative is to use LHPDF, in which case the parton density functions would be generated using an evolution package and a parametrization given in the .LHpdf file matching the PDF set's name. We will use LHGRID.

The final parameter is LSUBSET. The LHAPDF interface allows for multiple PDF sets to be loaded into memory at the same time, which saves time when frequent switching between sets is necessary. However, since we only need nucleon PDF sets (not nuclear- or photon- or pion sets) and will use only use one PDF set at a time for comparison with one Pythia 8.1 result obtained by using that same set, the LSUBSET variable will in our case always be set to zero.

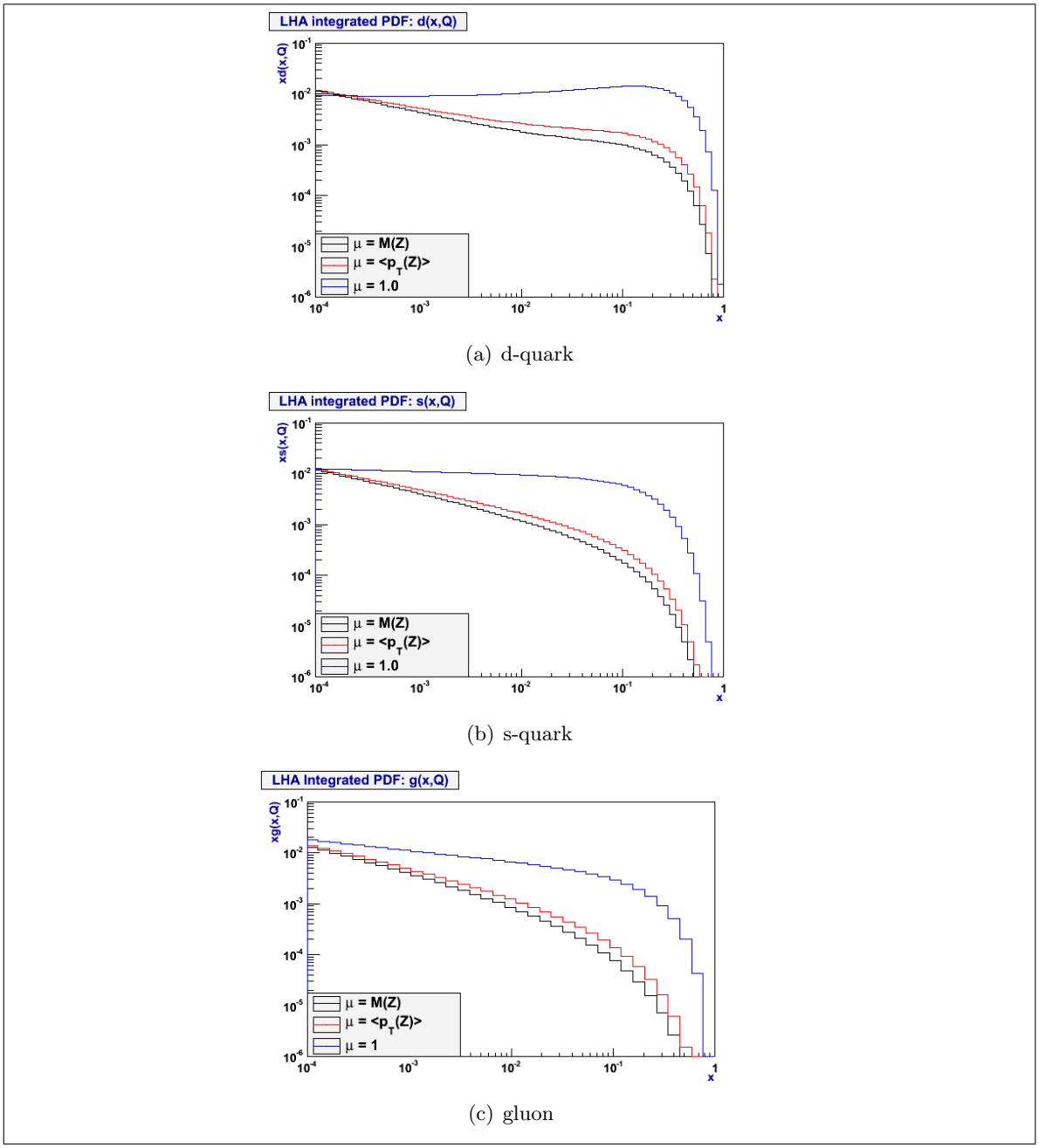
Many different functions are available in the LHAPDFWrapper class, but in general we will use only one of those. We wish to access the nucleon PDF sets, which can be done with:

```
1 double xfx(double x, double Q, int fl)
```

The function and its parameters are rather self-explanatory. As input we give Björken-x, scale Q and parton flavour. When using LHAPDF, flavours are always numbered in the

same way. Anti-quarks from $\bar{t}$ to $\bar{d}$ are -6 to -1, the gluon is 0 and quarks from d to t are 1 to 6. As output we get the parton density function $f(x, Q)$, multiplied by x.

We plot the parton density functions for the leading order CTEQ5l.LHgrid set in function of x and at different scales, with scale Q named μ. The three plotted scales, $\mu = M(Z)$ (the Z-Boson mass), $\mu = \langle p_T(Z) \rangle$ (the Z-Boson average transverse momentum, approximately 20 GeV) and $\mu = 1$ GeV, produce clearly different results. This is interesting as a reference, to quickly see if scale settings in other treatments (PYTHIA, KMR, CCFM) happened correctly. We will get back to this when we make the comparisons in section 9. The results here are shown for the d-quark, the s-quark and the gluon.



(a) d-quark

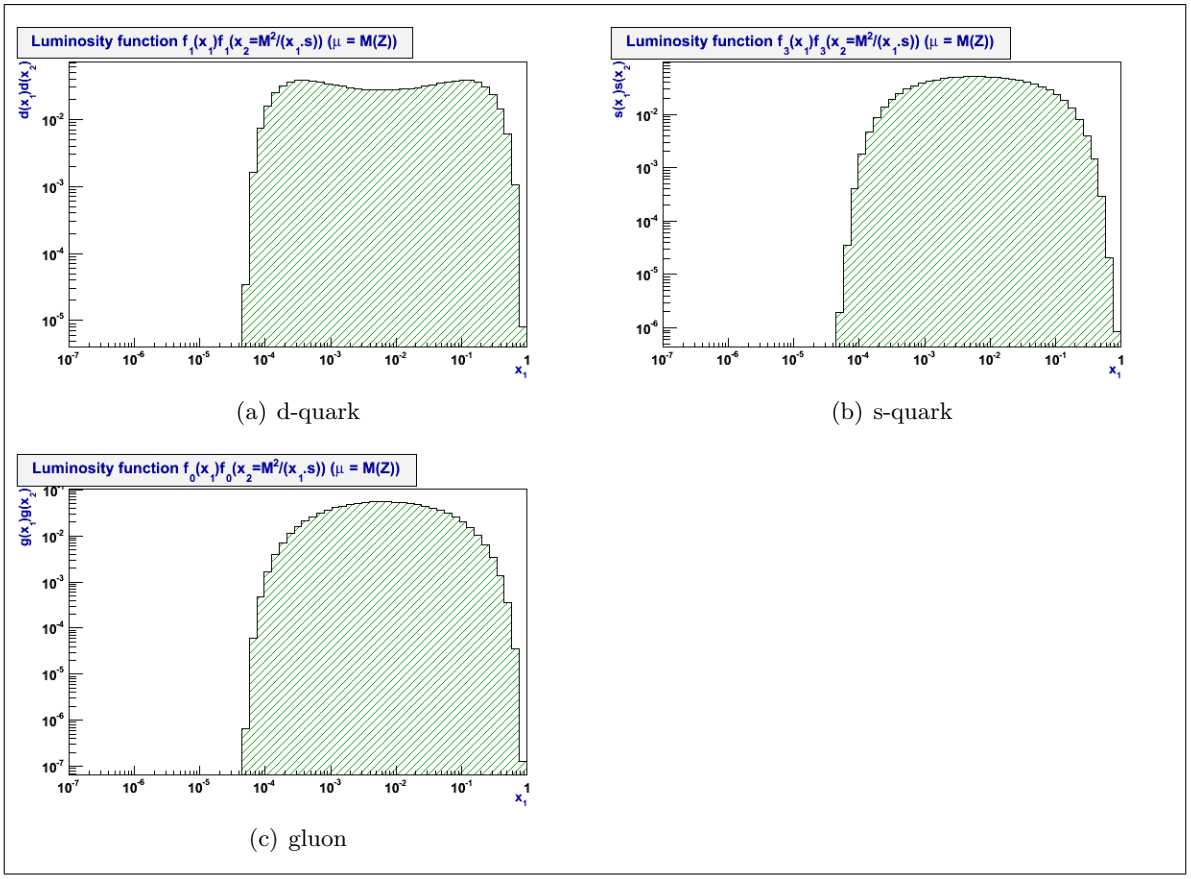
(b) s-quark

(c) gluon

Figure 29: Integrated parton density functions extracted from CTEQ5l.LHgrid with the LHAPDFWrapper C++ class

6.3 Parton Luminosity functions

As a reference for the parton luminosity functions obtained from PYTHIA 8.1, we can get the same functions using LHAPDF with the CTEQ5l.LHgrid set simply by making the product of two `xfx(x,Q,fl)` calls, with the proper settings for x_1 and $x_2 = \frac{M^2}{x_1 \cdot s}$. We obtain the results shown in figure 30.



(a) d-quark

(b) s-quark

(c) gluon

Figure 30: Luminosity function for the d-quark, s-quark and gluon, extracted through LHAPDF from the CTEQ5l.LHgrid set

7 Kimber-Martin-Ryskin approach

7.1 Introduction

As a second way of obtaining unintegrated parton density functions, next to our not entirely succesfull PYTHIA 8.1 attempt, we implement the Kimber-Martin-Ryskin approach as described in [9]. They present a procedure to calculate unintegrated distributions from known integrated distributions. If one starts with a standard DGLAP equation, there is a virtual and a non-virtual contribution. The virtual contribution does not affect the parton transverse momentum and may be split from the rest of the equation and put into a survival probability multiplicative factor. We will call this factor a Sudakov form factor. The non-virtual contribution remains as is, so that our final result will consist of two factors, discussed in sections 7.2 and 7.3.

7.2 The Sudakov form factor

This factor should describe the probability that a parton remains untouched during evolution from transverse momentum k_T up to factorization scale μ; the probability that there are no emissions in that interval. A Sudakov form factor fulfills exactly that aim. The survival probability is given by:

$$T_a(k_T, \mu) = exp\left[-\int_{k_T^2}^{\mu^2} \frac{\alpha_s(k_T)}{2\pi} \frac{dk_T^2}{k_T^2} \sum_{a'} \int_{z_{min}}^{z_{max}} \mathcal{P}_{a'a}(z) dz \right]$$

where a' indicates a sum over all partons. The factor in the exponent can be seen as the average number of emissions, $\langle n \rangle$. Assuming then that the distribution is poissonian, $exp\left(-\langle n \rangle\right)$ indicates the probability for no emission, the probability for survival. With the variable $\alpha_s(k_T)$ the first part of the integral cannot be done analytically, but since we know the splitting functions the second part can. We discuss the case for both quarks and gluons. For quarks we have:

$$T_q(k_T, \mu) = exp\left[-\int_{k_T^2}^{\mu^2} \frac{\alpha_s(k_T)}{2\pi} \frac{dk_T^2}{k_T^2} \int_{z_{min}}^{z_{max}} \left(\mathcal{P}_{qq} + \mathcal{P}_{gq}\right) dz \right]$$

with splitting functions:

$$\mathcal{P}_{qq} = \frac{4}{3} \frac{1+z^2}{1-z}$$
$$\mathcal{P}_{gq} = \frac{4}{3} \frac{1+(1-z)^2}{z}$$

Making use of a partial substitution $y = 1 - z$, we can easily do the z-integral:

$$\begin{aligned}
I_z &= \int_{z_{min}}^{z_{max}} \frac{4}{3}\frac{1+z^2}{1-z}dz + \int_{z_{min}}^{z_{max}} \frac{4}{3}\frac{1+(1-z)^2}{z}dz \\
&= -\int_{1-y_{min}}^{1-y_{max}} \frac{4}{3}\frac{1+(1-y)^2}{y}dy + \int_{z_{min}}^{z_{max}} \frac{4}{3}\frac{1+(1-z)^2}{z}dz \\
&= \int_{z_{min}}^{z_{max}} \frac{8}{3}\left[\frac{2}{z} - 2 + z\right]dz = \frac{16}{3}\ln\left(\frac{z_{max}}{z_{min}}\right) - \frac{16}{3}\left(z_{max} - z_{min}\right) + \frac{4}{3}\left(z_{max}^2 - z_{min}^2\right) \\
&= \frac{16}{3}\left[\ln\left(\frac{z_{max}}{z_{min}}\right) - (1 - 2z_{min})\right] + \frac{4}{3}\left(1 - 2z_{zmin}\right) = \frac{16}{3}\ln\left(\frac{z_{max}}{z_{min}}\right) - 4\left(1 - 2z_{min}\right)
\end{aligned}$$

For gluons we have:

$$T_g(k_T, \mu) = exp\left[-\int_{k_T^2}^{\mu^2} \frac{\alpha_s(k_T)}{2\pi}\frac{dk_T^2}{k_T^2}\int_{z_{min}}^{z_{max}} \left(\mathcal{P}_{qg} + \mathcal{P}_{gg}\right)dz\right]$$

with splitting functions:

$$\begin{aligned}
\mathcal{P}_{qg} &= \frac{1}{2}\left(z^2 + (1-z)^2\right) \\
\mathcal{P}_{gg} &= 6\left(\frac{1}{z} + \frac{1}{1-z} - 2 + z(1-z)\right)
\end{aligned}$$

Again making use of a partial substitution $y = 1 - z$, we can do the z-integral:

$$\begin{aligned}
I_z &= \int_{z_{min}}^{z_{max}} \left(z^2 - z + \frac{1}{2}\right)dz + \int_{z_{min}}^{z_{max}} \left(\frac{6}{z} + \frac{6}{1-z} - 12 + 6z - 6z^2\right)dz \\
&= \int_{z_{min}}^{z_{max}} \left(-5z^2 + 5z - \frac{23}{2} + \frac{6}{z}\right)dz \; - \int_{1-y_{min}}^{1-y_{max}} \frac{6}{y}dy = \int_{z_{min}}^{z_{max}} \left(-5z^2 + 5z - \frac{23}{2} + \frac{12}{z}\right)dz \\
&= -\frac{5}{3}\left(z_{max}^3 - z_{min}^3\right) + \frac{5}{2}\left(z_{max}^2 - z_{min}^2\right) - \frac{23}{2}\left(z_{max} - z_{min}\right) + 12\ln\left(\frac{z_{max}}{z_{min}}\right) \\
&= 12\ln\left(\frac{z_{max}}{z_{min}}\right) - \frac{32}{3} + 18z_{min} + 5z_{min}^2 - \frac{10}{3}z_{min}^3
\end{aligned}$$

In both cases, for the k_T^2-integral with variable $\alpha_s(k_T)$ we will use Monte Carlo integration. Important is that the z-integral is also function of k_T^2, through its boundaries. For those boundaries we have, still following the Kimber-Martin-Ryskin approach:

$$z_{max} = \frac{\mu}{\mu + k_T}$$

In general, we take the boundaries to be symmetrical:

$$z_{min} = 1 - z_{max}$$

Now to get the actual Sudakov form factors $T_q(k_T, \mu)$ and $T_g(k_T, \mu)$ we fill in our results for the z-integral. These are the final integrals which we will solve using Monte Carlo integration:

$$T_q(k_T, \mu) = exp\left[-\int_{k_T^2}^{\mu^2} \frac{\alpha_s}{2\pi}\frac{dk_T^2}{k_T^2} \cdot \left(\frac{16}{3}\ln\left(\frac{z_{max}}{z_{min}}\right) - 4\left(1 - 2z_{min}\right)\right)\right]$$

$$T_g(k_T, \mu) = exp\left[-\int_{k_T^2}^{\mu^2} \frac{\alpha_s}{2\pi}\frac{dk_T^2}{k_T^2} \cdot \left(12\ln\left(\frac{z_{max}}{z_{min}}\right) - \frac{32}{3} + 18z_{min} + 5z_{min}^2 - \frac{10}{3}z_{min}^3\right)\right]$$

In the following we will describe the Monte Carlo integration in detail for the quark case. The gluon case is entirely identical except for the different z-part. Monte Carlo integration is based on the law of large numbers. Given enough sample points, a Monte Carlo estimate will converge to the true integral:

$$I_{MC} = \frac{b-a}{N}\sum_{i=1}^{N} f(x_i) \;\longrightarrow\; I = \int_a^b f(x)dx$$

The central limit theorem allows for the distribution to be taken as being Gaussian: "*for large numbers the sum of independent random numbers is always normally distributed*". This means we can write for the effective variance:

$$\begin{aligned} V[I_{MC}] &= V\left[\frac{b-a}{N}\sum_{i=1}^{N} f(x_i)\right] = \frac{(b-a)^2}{N^2} N \cdot V[f(x)] \\ &= \frac{(b-a)^2}{N}\left(E[f(x)^2] - (E[f(x)])^2\right) \\ &= \frac{1}{N}\left[\frac{(b-a)^2}{N}\sum_{i=1}^{N} f(x_i)^2 - \left(\frac{(b-a)}{N}\sum_{i=1}^{N} f(x_i)\right)^2\right] \end{aligned}$$

where:

$$E[f(x)] = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

For the generation of random numbers we use RANLUX [12], a generator producing much better results than the built-in C++ generator or a simple congruential generator. Other than using a better random number generator, we can also benefit from generating random numbers along a well chosen distribution instead of uniformly. This distribution should match the function which is to be integrated as closely as possible and its analytical integral has to be known. The method is called importance sampling.

Since we have $\frac{dk_T^2}{k_T^2}$ in our integrals, let's use random number generation according to a normalized $\frac{1}{x}$-distribution:

$$g(x) = \left(\int_{x_{min}}^{x_{max}} \frac{1}{x}dx\right)^{-1}\frac{1}{x} \;\rightarrow\; G(x) = \left(\int_{x_{min}}^{x_{max}} \frac{1}{x}dx\right)^{-1}\int_{x_{min}}^{x} \frac{1}{x'}dx' = \frac{\ln\left(\frac{x}{x_{min}}\right)}{\ln\left(\frac{x_{max}}{x_{min}}\right)} = r_i$$

and:

$$x = x_{min}\left(\frac{x_{max}}{x_{min}}\right)^{r_i}$$

Next for the integral we can write:

$$\int_{x_{min}}^{x_{max}} f(x)dx = \int_{x_{min}}^{x_{max}} \frac{f(x)}{g(x)}\, dG(x) \;\approx\; \frac{1}{N}\sum_i \frac{f(x_i)}{g(x_i)} \ln\left(\frac{x_{max}}{x_{min}}\right) = f(x)w(x) \tag{7}$$

We have approximated $\frac{f(x)}{g(x)}$ as a constant, moving it in front of the integral, and then solved the remaining part analytically. If one wants to avoid this argument, an expectation value treatment may be used. Finally, what we gain from using the importance sampling method is a smaller variance. The effective variance is $V[f(x)/g(x)]$ instead of $V[f(x)]$ and because with our well chosen $g(x)$ the fraction $\frac{f(x)}{g(x)}$ is nearly constant, this new effective variance is a lot smaller.

Implementing this Monte Carlo procedure requires just a few lines of code. First we initialize our random number generator and define a variable to hold the generated random numbers. Later on we will use a loop to get the sum approximating the integral. In each loop entry we need only one random number, so our random number variable should be a vector of length one.

```
1  #include "ranlxd.h"
2
3  double Tq(double kmin, double kmax, int npoints) {
4      rlxd_init(2,32767);
5      double rvec[1];
6      ...
```

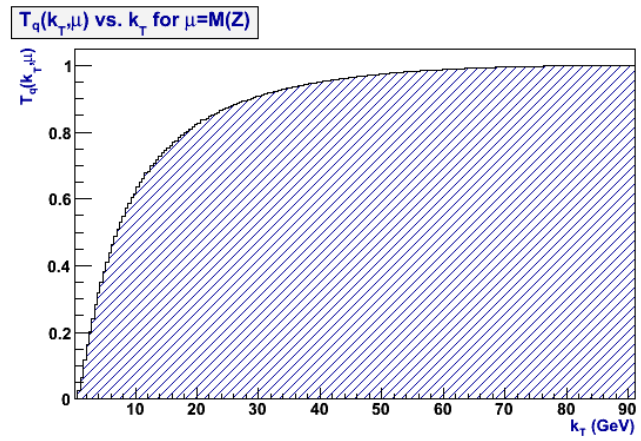
In the loop we generate a random number according to $g(x)$ as described before. For easy notation we substitute k_T^2 with t. Note that for the boundaries of the z-part we need k_T, while the generated random number matches k_T^2. A square root is required. The variable $\alpha_s(k_T)$ is extracted through the LHAPDF interface, used in the same way as described in section 6.2. The alphasPDF() function is called with variable k, the generated k_T.

```
5      ...
6      double  tmin=pow(kmin,2.), tmax=pow(kmax,2.), mu=kmax;
7      double t=0., alphas=0., sum0=0., sum00=0., sigma2=0., error=0., s=0.;
8      static const double PI = 3.1415926535897932;
9
10     for(int i=0; i<npoints; i++) {
11         ranlxd(rvec,rN);                                  // ranlux random
12         t       = tmin*pow(tmax/tmin,rvec0);              // random along 1/x
13         wt      = t*log(tmax/tmin);
14         k       = sqrt(t);
15         zmax    = mu/(mu+k);
16         zmin    = 1.-zmax;
17         alphas  = alphasPDF(k);
18         temp    = (alphas/2./PI)*(1/t)*(16./3.*log(zmax/zmin)-4.*(1.-2.*zmin))*wt;
19         sum0   += temp;
20         sum00  += temp*temp;
21     }
22     ...
```

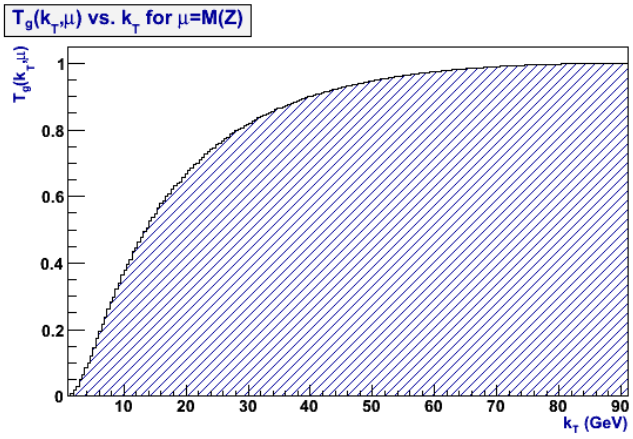
In line 18, we see function $f(x)$ times weight $w(x)$, as defined in equation (7) on page 56. Next we need to normalize our sum and we can calculate the error on the integral. Finally we can calculate the Sudakov form factor by taking the exponential of the calculated integral.

```
21      ...
22      sum0   /= npoints;
23      sum00  /= npoints;
24      sigma2  = abs(sum00 - sum0*sum0)/npoints;
25      error   = sqrt(sigma2);
26
27      s       = exp(-sum0);
28      return s;
29  }
```

The end result is a function giving the Sudakov form factor $T_q(k_T, \mu)$ according to input variables k_{min} and $k_{max} = \mu$, and for a given number of sample points $npoints$. We will be able to use it in the further implementation of the Kimber-Martin-Ryskin approach to unintegrated parton density functions. Below we show the Sudakov results in a histogram, for fixed $\mu = M_Z$ and for both quarks and gluons.



(a) $T_q(k_T, \mu)$ with $npoints = 10000$



(b) $T_g(k_T, \mu)$ with $npoints = 10000$

Figure 31: Histograms of the calculated Sudakov form factors

7.3 Unintegrated Parton Density Functions

With the Sudakov form factors calculated, we can take the next step and look at the formulas for the unintegrated parton density functions. The uPDF is constructed from number of emissions at scale k_T (represented by the integral), multiplied by the probability for no emission between k_T and μ (represented by the Sudakov form factor as calculated before):

$$f_q(x, k_T, \mu) = T_q(k_T, \mu) \cdot \left[\frac{\alpha_s(k_T)}{2\pi} \int_x^{z_{max}} \mathcal{P}_{qa'}(z)\ a'\left(\frac{x}{z}, k_T\right)\ dz \right]$$

and:

$$f_g(x, k_T, \mu) = T_g(k_T, \mu) \cdot \left[\frac{\alpha_s(k_T)}{2\pi} \int_x^{z_{max}} \mathcal{P}_{ga'}(z)\ a'\left(\frac{x}{z}, k_T\right)\ dz \right]$$

The splitting functions are the same as in section 7.2, but they appear in different combinations. In the quark case a' can either be the same quark as q, or a gluon. The integral will have two terms. In the gluon case a' can be any quark or gluon, the integral will have eleven terms. Eleven and not thirteen, since the top- and anti-top quark are excluded.

Another thing to watch out for are the integral boundaries. Upper limit z_{max} is still defined in the same way as a function of k_T, but x is now an input variable and not the symmetrical $1 - z_{max}$. Technically it is possible to come across a case where $x > z_{max}$, but physically this is nonsense: the momentum fraction a radiated parton cannot be larger than the momentum fraction of the parton it radiated from, $\frac{x}{z} \leq 1$. This is fixed by implementing a condition setting the integral result to zero when indeed $x > z_{max}$. Both integrals will be calculated using Monte Carlo integration.

We need values for the integrated parton density functions $a'(x, \mu' = k_T)$. We will extract them from a PDF set by making use of the LHAPDF interface again. The use of its function $xfx(x, Q)$ has been previously described in section 6.2. Note also that for the above formula we follow the notation in the Kimber-Martin-Ryskin paper [9] which has $a'(x, k_T) = x \cdot f_{a'}(x, k_T)$ and $\tilde{f}_{a'}(x, k_T, \mu) = k_T^2 \cdot f_{a'}(x, k_T, \mu)$. For the PDFset we again choose the same leading order CTEQ5l.LHgrid set, so that a difference in PDFset can not be the cause of any differences between results from different approaches.

This time for the Monte Carlo integration we won't use importance sampling. The product of the splitting functions and the integrated parton density functions in the integral is not as simple as the $\frac{1}{k_T^2}$ in the previous section. This means we cannot really benefit from using only one simple approximating function. To properly make use of importance sampling here we would have to split our integral in several parts and handle each of them with a different approximating function. We chose to drop the possible gains and use only uniform random numbers.

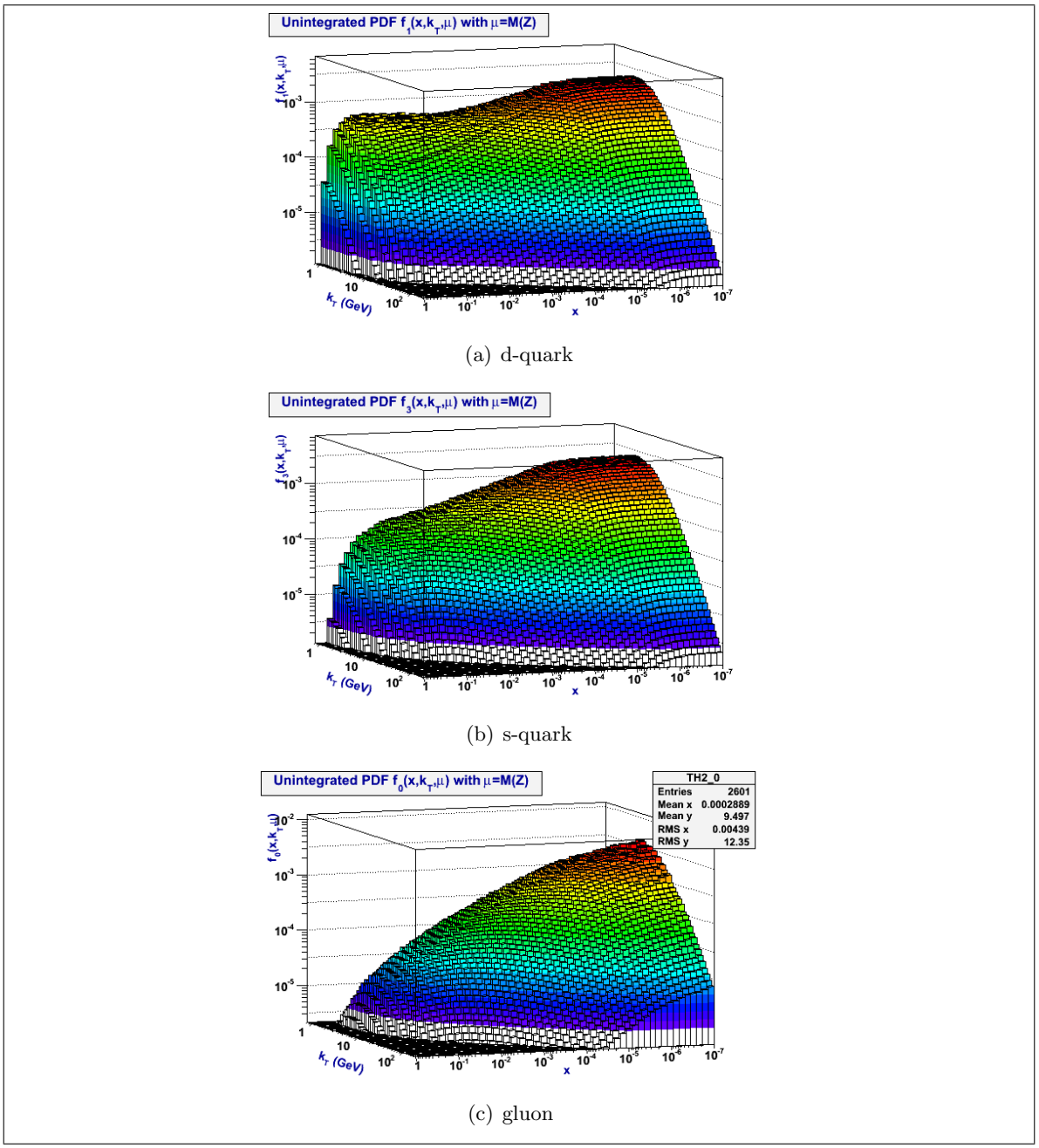
In the following we show how we implemented the procedure, calculating the unintegrated parton density functions on a grid and storing the result in a histogram.

```
1  ...
2  for(int ik=0; ik<=nk; ik++) {
3      for(int ix=0; ix<=nx; ix++) {
4          x1      = h1->GetXaxis()->GetBinCenter(ix+1);
5          k1      = h1->GetYaxis()->GetBinCenter(ik+1);
6          t       = Tq(k1,MZ,10000);      // sudakov form factor using function Tq
7          alphas  = alphasPDF(k1);        // variable alpha_s
8          zmax    = MZ/(k1+MZ);           // integral boundaries
9          zmin    = x1;
10
11         sum0    = 0.;
12         sum00   = 0.;
13         int rN=2;
14         double z=0., wz=0., rvecrN;
15         double temp1=0., temp2=0.;
16
17         for(int i=0; i<npoints; i++) {
18             ranlxd(rvec,rN);
19             z  = zmin+rvec0*(zmax-zmin);        // linear z
20             wz = zmax-zmin;                     // z weight
21             if(zmax > zmin) {
22                     temp1  = t*(alphas/2./PI)*(4./3.*(1.+pow(z,2.))/(1.-z))
23                                 *xfx(zmin/z,k1,fl)*wz;
24                     sum0  += temp1;
25                     sum00 += pow(temp1,2.);
26                     temp2  = t*(alphas/2./PI)*(1./2.*(pow(z,2.)+pow(1.-z,2.)))
27                                 *xfx(zmin/z,k1,0)*wz;
28                     sum0  += temp2;
29                     sum00 += pow(temp2,2.);
30             }
31         }
32         sum0   /= npoints;
33         sum00  /= npoints;
34         sigma2  = abs(sum00 - sum0*sum0)/npoints;
35         error   = sqrt(sigma2);
36         f       = sum0;
37
38         // change f from KMR notation to normal and fill histogram
39         h1->Fill(x1,k1,f/(k1*k1));
40     }
41 }
42 ...
```

In line 23 fl indicates the quark flavour, this is the first term of the integral. In line 27 we have 0 for the gluon, this is the second term of the integral. It is clear how running this piece of code for different fl will result in the histograms of the unintegrated quark density functions for all different quark flavours.

Obtaining the unintegrated gluon density function can be done in an entirely similar way. Only the splitting functions need to be changed and extra terms need to be added for all different quark flavours. Since all quark terms are identical except for variable fl, those ten terms can easily be grouped into one by using a loop over fl.

Below we show the results of our calculations. We used the CTEQ5l.LHgrid PDFset, with Q^2-scale set to $Q = M(Z)$, the Z-Boson mass, and α_s evaluated at k_T in both the calculation of the Sudakov form factor and the calculation of the integral for the unintegrated parton density function. We include the reintegrated functions next to the results for the unintegrated parton density functions, which will be helpful in the comparison in section 9.



(a) d-quark

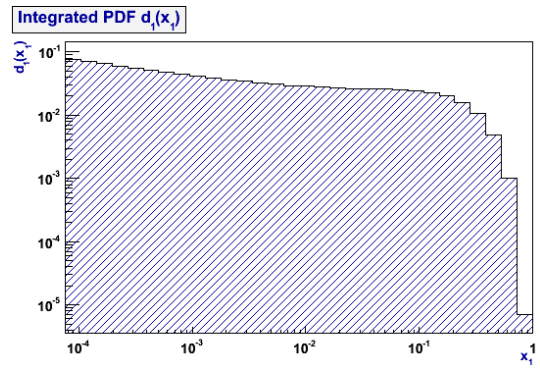
(b) s-quark

(c) gluon

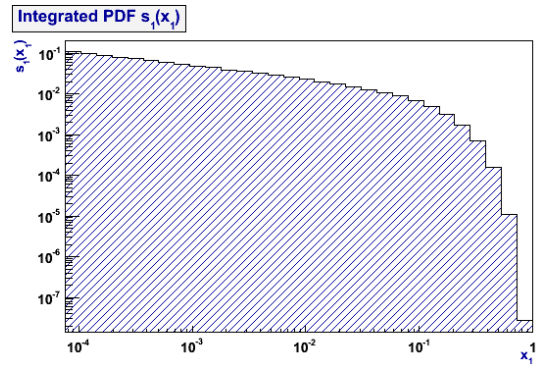
Figure 32: Unintegrated parton density functions for the d-quark, the s-quark and the gluon, obtained with the KMR approach

7.4 Integrated Parton Density Functions

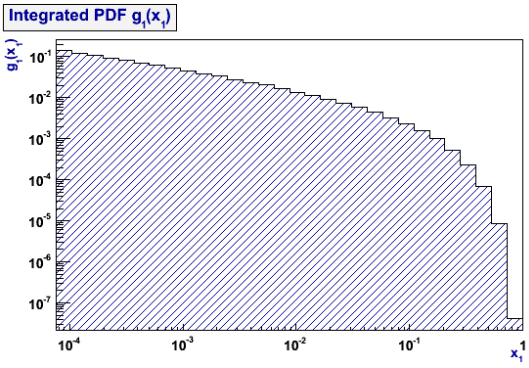
Results for the integrated parton density functions for the d-quark, the s-quark and the gluon, obtained by integrating the unintegrated KMR results.



(a) d-quark



(b) s-quark



(c) gluon

Figure 33: Integrated parton density functions for the d-quark, the s-quark and the gluon, obtained with the KMR approach

8 CCFM Integrated Gluon Density Functions

8.1 Introduction

As another source of integrated gluon density function which can be compared with the previous PYTHIA 8.1, LHAPDF and KMR results, we use an implementation of CCFM evolution. We use the ccfm-final-setA0 dataset obtained from [18] and a program performing interpolation of the dataset [19], providing an easily manageable function `get(x,`k_T^2`,`Q^2`)`.

8.2 Integrated Gluon Density Function

To get results we construct a two dimensional histogram in x and k_T using ROOT and fill it using `get(x,`k_T^2`,`Q^2`)`. Because we want to compare integrated distributions, we integrate over k_T. The output is shown in figure 34.

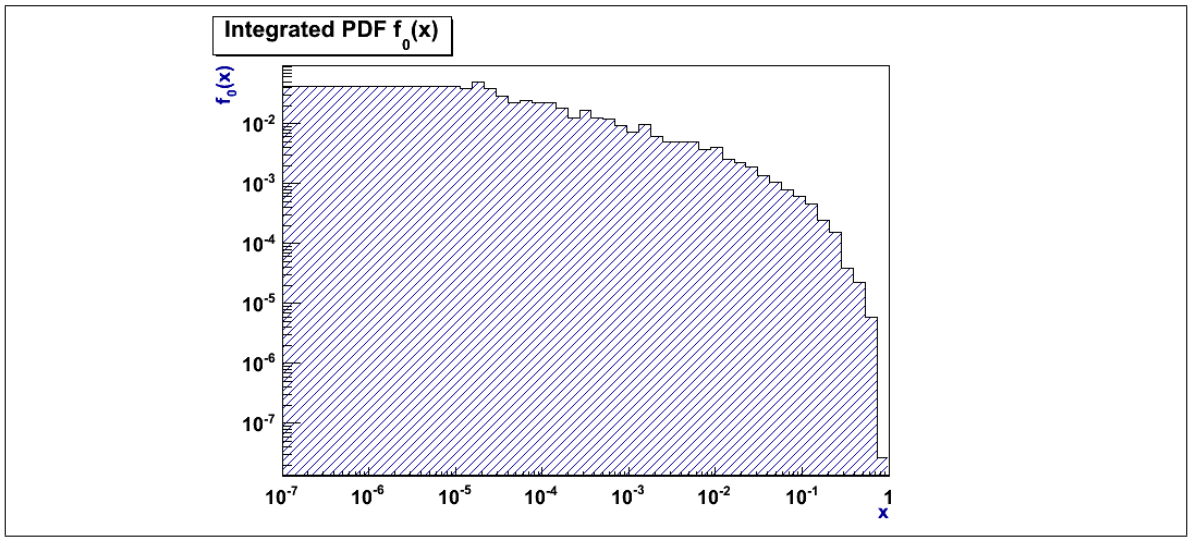


Figure 34: Integrated gluon distribution obtained from a CCFM dataset

9 Comparison of Parton Density and Luminosity Functions

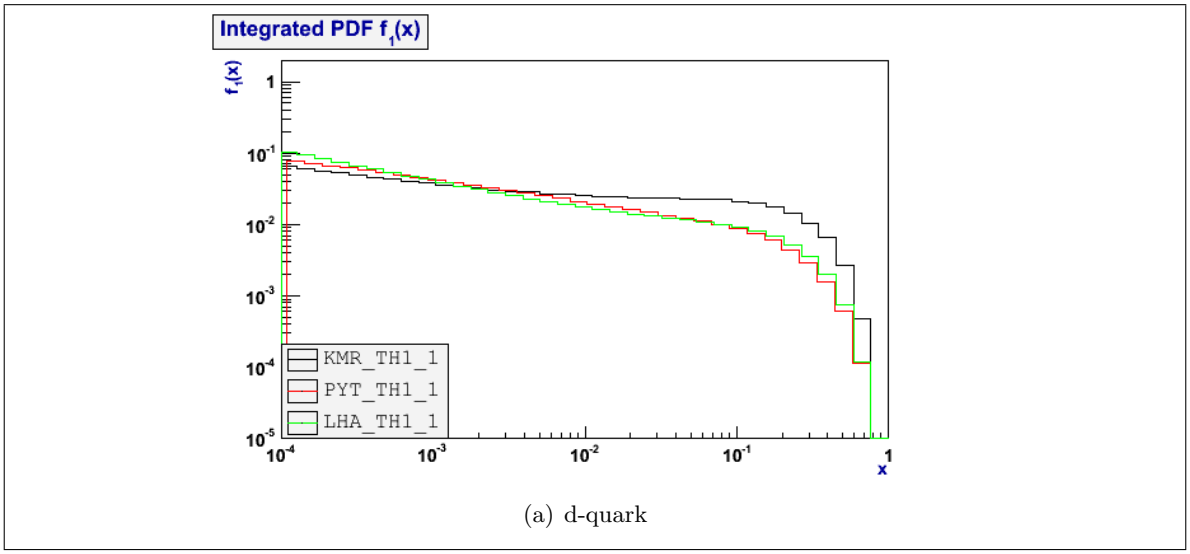
9.1 Integrated Parton Density Functions

To begin with we show all results for integrated parton density functions together. For the quarks this means PYTHIA 8.1, KMR and LHAPDF, for the gluon we have CCFM as a fourth result, but not the PYTHIA 8.1 result due to technical problems. As usual we plot for the d-quark, the s-quark and the gluon. In each of the three histograms, we normalized the results in the range $x = \left[10^{-4}, 1\right]$. Whenever a PDFset was needed, we used the CTEQ5l.LHgrid set, in an attempt to minimize the source of differences between the approaches. For the same reason, Q^2-scale was set to $Q = M(Z)$, the Z-Boson mass, whenever possible.

For both quark distributions, the agreement between the PYTHIA 8.1 (red) and the LHAPDF (green) result is good. The KMR result (black) on the other hand shows some deviation. However, integrating the KMR uPDFs (which used the same leading order CTEQ5l PDFs through LHAPDF as we used for the standalone LHAPDF integrated result), one would expect to find a matching result for the LHAPDF result. A possible explanation for the difference can be found in the treatment of α_s. In the KMR approach α_s is evaluated at k_T in both the Sudakov form factor and the following integral, while in DGLAP evolution (as in the PDFset) α_s is generally evaluated at Q.

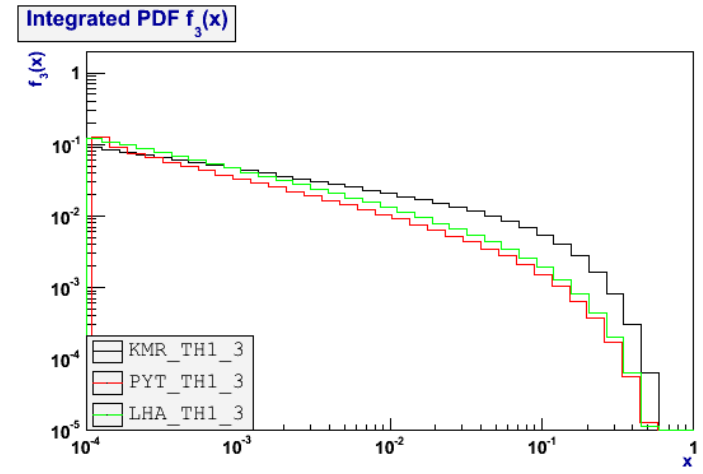
For the gluon distribution, there is the same difference between KMR and LHAPDF, KMR being a bit larger at large x. The CCFM result matches the KMR result best, but suffers a bit from bad statistics. For an explanation of the difference with the LHAPDF result, one could track back the PDF and α_s treatment choices used for the CCFM dataset [18].

In all cases it is clear that the LHAPDF result for M(Z) is the correct match (cfr. section 6.2), indicating that there are no problems with scale Q^2.

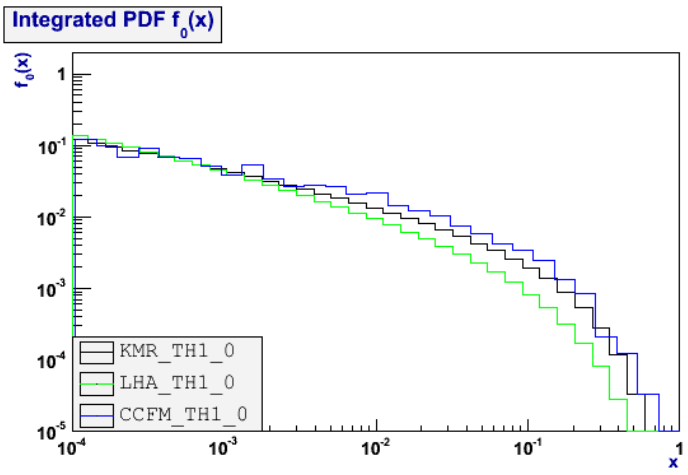


(a) d-quark

Figure 35: Comparison of PYTHIA 8.1, KMR, LHAPDF and CCFM integrated parton density functions



(b) s-quark

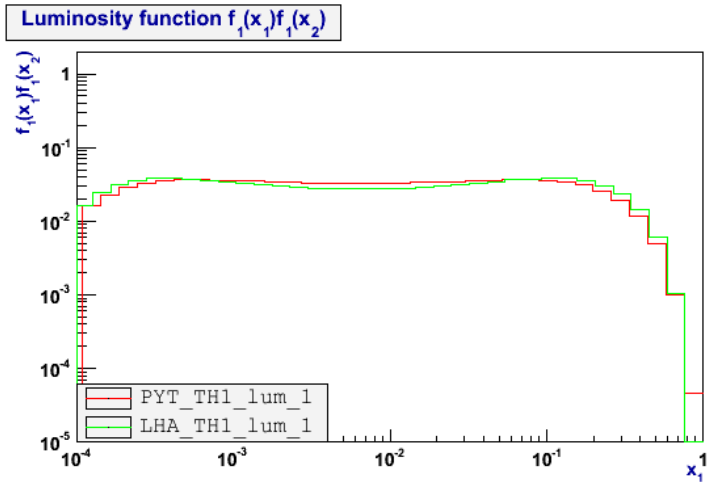


(c) gluon

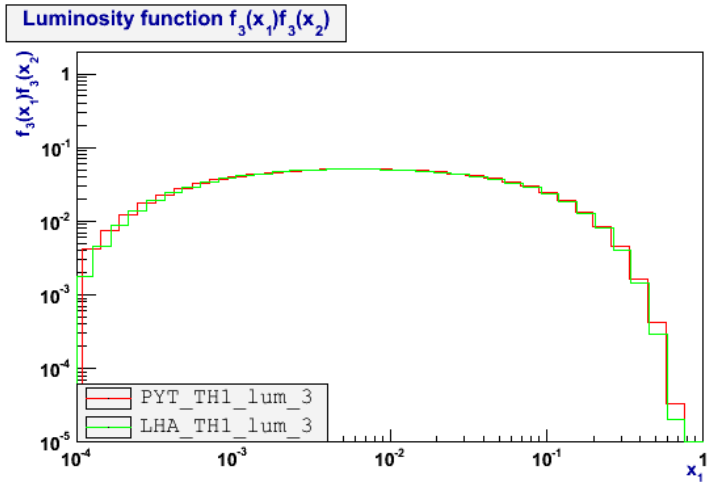
Figure 35: (cont.) Comparison of PYTHIA 8.1, KMR, LHAPDF and CCFM integrated parton density functions

9.2 Parton Luminosity Functions

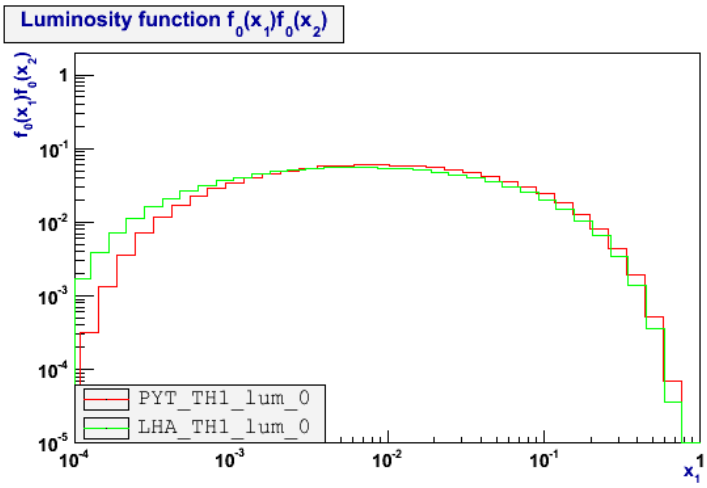
To end this part, we can compare the results for the parton luminosity function obtained with PYTHIA 8.1 and LHAPDF. The match is reasonable for all three cases, although better for the quarks than for the gluon. The fast drop at small x for the PYTHIA 8.1 gluon result may be explained by our ignoring of the Breit-Wigner distribution coming into play at that point in x, where we used a delta-function $\delta\left(x_2 - \frac{M^2}{x_1 \cdot s}\right)$.



(a) d-quark



(b) s-quark



(c) gluon

Figure 36: Comparison of PYTHIA 8.1 and LHAPDF parton luminosity functions

Part III - Determination of Parton Density Functions for Pythia

The idea of this part is to use generated event data produced with a Monte Carlo event generator, Pythia 6.4 in our case, create a PDF parametrization using the LHAPDF interface and then tune the parametrization to the reference data using the Professor package. To efficiently acquire data in the correct formats, we also use the Rivet package.

10 Data preparation

10.1 HERA Combined Data

For the tuning of PDFs with Professor we will use the HERA Combined Results (H1+ ZEUS). In particular we use the neutral current (NC) e^+p data set, which has better statistic than the NC e^-p set or the charged current (CC) sets. These data sets are available in text format on the HERA website [24]. However, for use with Professor, we need the data in AIDA xml histogram format [17], similar to the output of the Rivet analysis we will perform on generated Monte Carlo data.

To obtain the necessary .aida file containing the HERA Combined Results, we wrote a Root macro to perform crude conversion, directly usable for this case only. We read in the text file, create the required histograms in Root format an finally use those histograms to write a .aida file. The key elements of the AIDA data structure are the file intro:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE aida SYSTEM "http://aida.freehep.org/schemas/3.0/aida.dtd">
3 <aida version="3.0">
4 <implementation version="1.0" package="LWH"/>
5 <dataPointSet name="set1"
6 title="set1" path="/HeraData" dimension="2">
7 <dimension dim="0" title="xtitle" />
8 <dimension dim="1" title="ytitle" />
```

a list of datapoints:

```
9  <dataPoint>
10   <measurement value="4.7469e-04" errorPlus="3.8173e-05" errorMinus="3.8173e-05"/>
11   <measurement value="1.3420e+00" errorPlus="3.4355e-02" errorMinus="3.4355e-02"/>
12 </dataPoint>
13 ...
```

Therein the first measurement line describes the bin center and half width to the right and to the left, while the second measurement line gives the bin value and its error. The .aida file ends with some closing elements:

```
16 </dataPointSet>
17 </aida>
```

If multiple histograms need to be written for the same data set, there will simply be more than one `<dataPointSet>` in the file.

10.2 Choosing a grid for the representation of the HERA data

What we start from with the HERA combined results data is a list of results at given x and Q^2. While the x and Q^2 originate from nice histograms with regular bin widths, the combination of two data sets (H1+ZEUS) has done away with the niceness and leaves us with non evenly spaced x and Q^2 values. Of course these data points can be filled in any chosen histogram, but as we want to keep separate as much of them as possible (there are only so many data points and we don't want to throw any of them away by averaging), we are looking for a grid on which each point falls in its own proper bin. This could be done by just choosing a very fine grid, keeping all data points seperate but also leaving a lot of bins empty. This would however introduce problems on the side of the generated data, because we need to use the exact same grid there (for later fitting) and we would need an impossibly large number of events to get reasonable statistics for such a fine grid. Also, because of the higher Q^2 lower x and lower Q^2 higher x link, there are large empty fields in a grid that encompasses all data points. We attempt to compromise by dividing the Q^2 range in three (unevenly) and sticking to a certain x-spacing for each of the three ranges. One could argue that more Q^2 ranges would allow to close in on the filled bins better, cutting corners and doing away with even more empty bins, but each range needs seperate Monte Carlo runs, seperate analysis and seperate fitting. Choices have to be made keeping in mind the limits on having to repeat these CPU intensive processes. Below we illustrate how we divided the Q^2 range and how the HERA data falls in this grid.

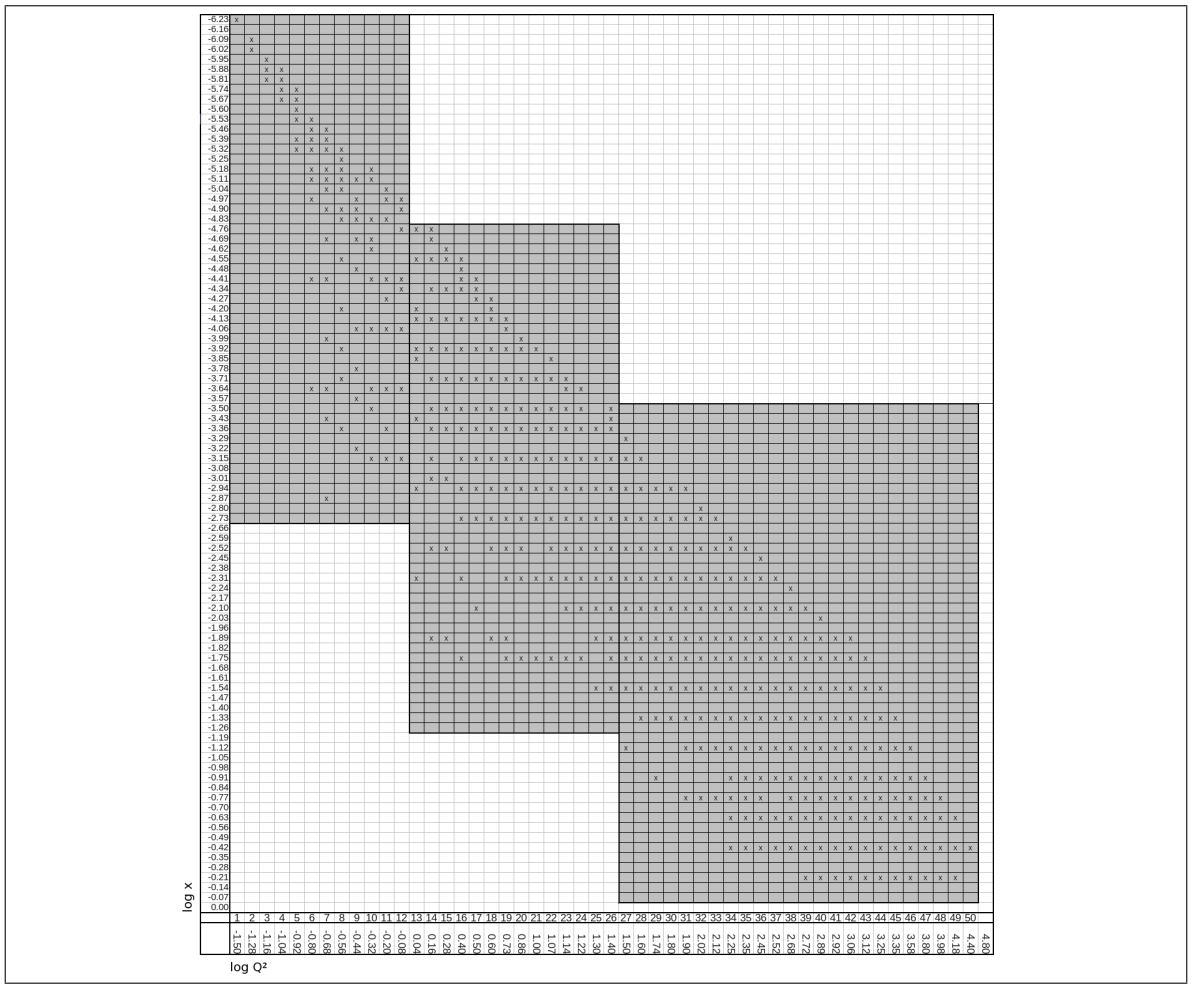


Figure 37: Grid for the representation of the HERA data, in three Q^2 ranges

The x-axis is standard logarithmic, with widths $\log \frac{x_{upp}}{x_{low}} = 0.07$ making a total of 100 bins between $\log x = -7$ and 0. For the actual histograms we take ranges of 50 bins from this set. The choice to keep the number of x-bins fixed on fifty for all three Q^2-ranges has to do with the data analysis, which would need more rewriting if the number changed each time. The bin widths on the Q^2 were chosen variable to accomodate the reference data table, the smallest being $\log \frac{Q_{upp}}{Q_{low}} = 0.03$ and the largest 0.40. It will turn out that the preparation we have done so far, including this binning of the generated and Monte Carlo data, does not suffice to produce reliable results with PROFESSOR. Improvement is possible on many fronts, one of which is this grid. Especially the many empty bins are uninteresting. A next step could be to carefully analyse the repercussions of different binning on the uncertainty of the results for generated data, compared to the uncertainty of the reference data.

11 LHAPDF PDFset

In part II we used the default and built-in CTEQ5l PDFset with PYTHIA, but also introduced the LHAPDF interface which allowed us to access PDFs from a chosen set whenever we wanted to. Other than this independent access, the LHAPDF interface also allows the user to choose a non built-in PDFset to be used with PYTHIA for the event generation itself. This is what we want to make use of in this part of the work, aiming to parametrize PDFs and fit with respect to HERA reference data to obtain optimized PDFs for PYTHIA 6.4 (PDF4MC idea). To get started on this we need to determine which parametrizations we can use for the PDFs and then build a file containing our parametrizations for the PDFs which can be understood by the LHAPDF interface. In the following we discuss both those elements, giving just one of many possibilities for a parametrization.

11.1 PDF parametrization

For our parametrizations we follow the LHAPDF 'x-taylor' format:

$$p_1 \cdot x^{p_2} \cdot (1-x)^{p_3} \cdot \left(1 + p_4\, x^{1/n} + p_5\, x^{2/n} + \ldots\right)$$

We do not parametrize all PDFs, but make use of the LHAPDF option to define some (or all) of them as combinations of other functions. We parametrize u_{val}, d_{val}, S, g and Δ, and then define all thirteen (12q + 1g) PDFs as a combination of those five functions. u_{val}, d_{val} and g speak for themselves, the distributions of the u- and d- valence quarks and the gluon. S represents the sea quarks; we exclude the top quark and assume there are five of them contributing equally. This is why in the composition of for example the $\bar{u}$-quark there is a contribution of 0.2 S (see codeline 34 on page 71). Finally Δ stands for the asymmetry between the $\bar{u}$- and $\bar{d}$- quark, visible in the Gottfried sum rule (for more details: chapter 4.5 in [4]). The contribution of Δ can be seen by comparing codelines 34 and 36 on page 71.

The 'x-taylor' format leaves room for a lot of parameters, however, keeping in mind that we will need to fit those parameters later on, we limit ourselves to four parameters for each function, leaving us with a total of twenty parameters which will probably need to be reduced even further. Each function will then look like (with n fixed, usually $n = 2$):

$$p_1 \cdot x^{p_2} \cdot (1-x)^{p_3} \cdot \left(1 + p_4\, x^{1/n}\right)$$

11.2 An .LHpdf file

The LHAPDF interface can work with two types of files, .LHgrid files and .LHpdf files. Files of the first type contain precalculated tables for the PDFs, while files of the second type contain only information on the grid, the parametrizations and the type of evolution, calculations happen at runtime. In our case it suffices to build an .LHpdf file. All details on how to compose such a file can be found at [26], but we will summarize here what we did.

After some preamble, the first choice that needs to be made is how to evolve and how the grid looks. We use the 'QCDNUM' package for evolution and allow the grid to extend from $x = 10^{-6}$ to 10^{0} in 400 steps and from $Q^2 = 10^{-2}$ to 10^{6} in 100 steps:

```
'Evolution:'
'nlo',1.0,1.0
'QCDNUM'
'pythia6.grid',400,1d-6,1d0,100,1d-2,1d6
```

Secondly the handling of α_s is set: variable, next-to-leading order and dependent on the Z-mass and the heavy quark masses:

```
'Alphas:'
'Variable','nlo','MRSTalfa'
1,91.118,1.43,4.3,180.0
```

Now the real parametrization section can start. As stated we parametrize five functions, and have a total of twenty parameters. Since $\alpha_s(M_Z)$ also forms a parameter, the function parameters are numbered from 2 to 21.

```
'Parametrization:'
'MRST',-1,5
'uv','x-taylor'
2.0,4
2,3,4,5
'dv','x-taylor'
2.0,4
6,7,8,9
'S','x-taylor'
2.0,4
10,11,12,13
'g','x-taylor'
2.0,4
14,15,16,17
'Delta','x-taylor'
1.0,4
18,19,20,21
```

Once we have the functions, the composition of the parton density functions with respect to them needs to be defined. To begin with the t- and $\bar{t}$-quarks are left out and the b-, $\bar{b}$-, c- and $\bar{c}$-quark densities are defined with a mass treshold. The remaining quark densities and the gluon density are fixed in terms of the five functions:

```
'Compose:'
'tbar','none'
'bbar','treshold'
4.3
'cbar','treshold'
1.43
'sbar','composite'
0.0,0.0,0.1,0.0,0.0
'ubar','composite'
0.0,0.0,0.2,0.0,-0.5
'dbar','composite'
0.0,0.0,0.2,0.0,0.5
'g','composite'
0.0,0.0,0.0,1.0,0.0
'd','composite'
0.0,1.0,0.2,0.0,0.5
'u','composite'
1.0,0.0,0.2,0.0,-0.5
's','composite'
0.0,0.0,0.1,0.0,0.0
'c','treshold'
1.43
'b','treshold'
4.3
't','none'
```

Finally the file has a list with the parameter values. In our case these will be subject of the PROFESSOR 1.2.1 fit later on, we borrowed the initial values from existing LHAPDF parametrizations. Both lines of parameters (here split) are identical. The first line indicates the default values, the second and possible following lines indicate the different sets included in the file (we have only one set):

```
'Parameterlist:'
'list',1,21
0.118 0.6051 0.4089 3.395 2.078 0.05811 0.2882 3.874 34.69 0.2001
      -0.2712 7.808 2.283 64.57 0.9171 6.587 -3.168 1.29 1.183 9.808 9.987
0.118 0.6051 0.4089 3.395 2.078 0.05811 0.2882 3.874 34.69 0.2001
      -0.2712 7.808 2.283 64.57 0.9171 6.587 -3.168 1.29 1.183 9.808 9.987
'End:'
```

12 Monte Carlo Event Generation using PYTHIA 6.4

As mentioned before, generating DIS events in PYTHIA 8.1 is impossible, which is why we use PYTHIA 6.4. This e^+p event generation with PYTHIA 6.4 can be done standalone, running a fortran routine entirely similar to the C++ code we described for pp event generation with PYTHIA 8.1 in part II, but in order to simplify the analysis of the Monte Carlo data with RIVET 1.5.0, we make use of the AGILE package [16]. This package, more in particular its function `agile-runmc`, allows to start a Monte Carlo run (with a fortran generator) based on a parameter file, to background it, and to directly perform a RIVET analysis on the output (while it is being created). Important herein is also the use of the UNIX .fifo file structure which allows to pipe the AGILE output data to the RIVET input channels.

We won't go in detail about the use of AGILe, but we show the necessary parameter settings to generate DIS events with Pythia 6.4. In the end it doesn't make any difference whether these parameters are used directly or through AGILe.

Pythia 6.4 parameter	value set	explanation
MSEL	0	user selected process
MSUB(10)	1	DIS events
MSTP(11)	0	no electron radiation
MSTP(21)	4	only charged current interactions
MSTP(52)	2	user selected external pdf
MSTP(51)	29000	number of external pdf
MSTP(61)	2	initial state radiation
MSTP(71)	0	no final state radiation
MSTP(81)	0	no multiple parton-parton interactions
MSTU(101)	0	fixed α_{em}
CKIN(35)	0.04	minimal Q^2
CKIN(36)	32000	maximal Q^2

Note that external pdf number 29000 actually belongs to the MRST98 LHAPDF PDFset. Because we couldn't figure out how to interface the program with an entirely user defined set, we borrowed this MRST98 file (which we didn't need) to contain our parametrization.

13 A Rivet Analysis

The Rivet package is meant to simplify the analysis of events. In general it uses a initialize-analyze-finalize structure. In the initialize section, the user can define which parts of the data are going to be used and book histograms which will be filled later on. In the analyze section, the data becomes available per event (in some sort of automatic event loop) and can be used for calculations or directly filled in histograms. Then in the finalize section, the created histograms can be normalized, or calculations depending on global information such as the total cross section or number of events can be performed. In the following sections we will provide some more detail on the analysis we wrote to process the DIS events created with Pythia 6.4 and then also on how an analysis is compiled, run and interpreted technically.

13.1 Writing the analysis

The Rivet system makes use of projections to easily provide access to certain parts of the event data. In the case of DIS events, the projection `<DISkinematics>` is the most important one. Without any fuss it provides DIS variables x, y, Q^2, W^2 and s (per event). These can then be used to construct histograms. In our case, the histograms of interest are of structure function F_2. As we have the HERA data as a reference (with $F_2(x)$ for several Q^2 values), we want to construct histograms for our Monte Carlo data that represent the same information.

The system allows initialization of histograms directly in the AIDA format (wanted for use with Professor), but it is important to watch out with defaults and for example to define logarithmic binning for histograms in x and Q^2. Also, on paper, the Rivet package has options for 2D histograms. These functions are however not very willing to adapt to the logarithmic binning we want. Therefore, it is easier to split the 2D histogram into an array of 1D histograms which can be fixed with the necessary logarithmic options. The definition of projections and histograms makes up the initialize section of the analysis:

```
private:
  vector<AIDA::IHistogram1D*> _histn, _histF2;
  double qedg2[51], xedg2[51];

public:
  void init() {
      const DISKinematics dk;
      addProjection(dk,"Kinematics");

      const BinEdges xedg = logBinEdges(50,1E-7,1.0);
      const double qedg[51] = { ... };
      _histn   = vector<AIDA::IHistogram1D *>(50);
      _histF2  = vector<AIDA::IHistogram1D *>(50);
      for(int i=0; i<50; i++) {
          qedg2[i] = pow(10.,qedg[i]);
          xedg2[i] = xedg[i];
          sprintf(hnamen,"n_%02i",i+1);
          sprintf(hnamesig,"F2_%02i",i+1);
          _histn[i]   = bookHistogram1D(hnamen,xedg,hnamen,"x","n");
          _histF2[i]  = bookHistogram1D(hnamesig,xedg,hnamesig,"x","F2");
      }
      qedg2[51] = pow(10.,qedg[51]);
      xedg2[51] = xedg[51];
  }
```

The next section of the analysis, the analyze section, contains the majority of calculations needed in the processing of the events as well as the filling of the previously constructed histograms. Before we show how it can be implemented, we need to understand what we want to extract from the event data. The `<DISKinematics>` projection provides the standard DIS variables for each event, allowing us to create a histogram $N(x, Q^2)$ in the event loop. We are however interested in structure function F_2 and not merely a number of events. We start with the discretization of the differential cross section in which we see luminosity L as a scale factor and bin widths Δx and ΔQ^2:

$$\frac{d^2\sigma}{dxdQ^2} = \left[N_{bin} \cdot \left(\frac{\sigma_{tot}}{N_{tot}} \right)_{=\frac{1}{L}} \right]_{=\Delta\sigma} \cdot \frac{1}{\Delta x \Delta Q^2}$$

Then continuing towards an expression for structure function F_2, we work with equation (1) on page 5:

$$\begin{aligned} & \left(1 + (1-y)^2\right) F_1 + \frac{1-y}{x} \left(F_2 - 2xF_1\right) \\ = \;& y^2 F_1 + \frac{1-y}{x} F_2 \\ = \;& \frac{1}{2x} \left[y^2 \left(2xF_1 - F_2\right) + 2(1-y)F_2 + y^2 F_2 \right] \\ = \;& \frac{1}{2x} \left[\left(1 + (1-y)^2\right) F_2 - y^2 F_L \right] \end{aligned}$$

to find:

$$\frac{d^2\sigma}{dxdQ^2} = \frac{4\pi\alpha^2}{Q^4} \frac{1}{2x} \left[\left(1 + (1-y)^2\right) F_2 - y^2 F_L \right]$$

If we drop the negligible F_L (this is valid for $y < 0.6$), we come to an expression for F_2:

$$F_2 \approx \sigma_{red} = \frac{xQ^4}{2\pi\alpha^2} \frac{1}{1+(1-y)^2} \frac{d^2\sigma}{dxdQ^2}$$

and with the previously discretize differential cross section we have final form:

$$F_2 \approx \sigma_{red} = \frac{xQ^4}{2\pi\alpha^2} \frac{1}{1+(1-y)^2} \left[N_{bin} \cdot \left(\frac{\sigma_{tot}}{N_{tot}} \right) \right] \frac{1}{\Delta x \Delta Q^2}$$

Nearly all of these calculations have to be done in the event loop. Only the scaling with the luminosity factor has to wait until the finalize section, because the total cross section and number of events are only available at the end of the run. Below we show our analyze section implementation. To circumvent the problem with the 2D histograms, we have a number of 1D histograms in x which we bin with a manual selection on Q^2. Note that the RIVET histograms automatically add a factor $\frac{1}{\Delta x}$ to the weight, which is why we need an extra Δx in the weight for histograms $N(x, Q^2)$ and don't have Δx in the weight for histograms $F_2(x, Q^2)$. The event weight itself is always one in our case.

```
public:
  void analyze(const Event& event) {
      const double weight = event.weight(); // (always 1 here)
      double weightx = 1., weightq2 = 1., factor = 1.;

      const DISKinematics& dk = applyProjection<DISKinematics>(event,"Kinematics");
      double x  = dk.x();
      double y  = dk.y();
      double q2 = dk.Q2();
      factor = x*q2*q2/2/pi/alpha/alpha/(1.+(1.-y)*(1.-y));

      for(int i=0; i<50; i++) {
          weightq2 = qedg2[i+1] - qedg2[i];
          if(qedg2[i] < q2 && q2 <= qedg2[i+1]) {
              for(int j=0; j<50; j++) {
                  if(xedg2[j] < x && x <= xedg2[j+1]) {
                      weightx = xedg2[j+1]-xedg2[j];
                  }
              }
              _histn[i]->fill(x,weight*weightx);
              if(y <= 0.6) _histF2[i]->fill(x,weight/weightq2*factor);
              Q2sum[i] += q2;
              N[i] += 1;
          }
      }
  }
```

The last part of the analysis is the finalize section, where histograms are scaled with luminosity and dimensional factors. For the luminosity, we have the known $L = \frac{N_{tot}}{\sigma_{tot}}$. Next to that, we need to factor out the dimensions of the cross section (given by PYTHIA in pb) and Q^2 (given in GeV^2). In natural units this can be done with:

$$\begin{aligned} 1 \text{ pb} \cdot \text{GeV}^2 &= \left[\frac{1.054 \cdot 2.998}{1.602} \cdot 10^{-7} \frac{\text{m} \cdot \text{eV}}{\hbar \text{c}} \right]^{-2} \cdot 10^{-12} 10^{-28} \text{ m}^2 \cdot 10^{18} \text{ eV}^2 \\ &= 2.570 \cdot 10^{-9} \end{aligned}$$

As a last point we also calculate the mean Q^2 value per histogram, which will not be equal to the Q^2 range's centre. All this results in the following finalize section:

```
public:
  void finalize() {
      double normfacn=1./sumOfWeights();
      double normfacF2=crossSection()/sumOfWeights()*pbGeV2;
      for(int i=0; i<50; i++) {
          _histn[i]->scale(normfacn);
          _histF2[i]->scale(normfacF2);
          Q2mean[i] = Q2sum[i]/N[i];
      }
  }
```

The output of the RIVET analysis are several histograms $F_2(x)$ for some Q^2 value (range's mean). These can be compared to the histograms earlier obtained from the HERA data.

13.2 Building and running the analysis

To be able to run a user built RIVET analysis on HEPMC data (for example output of runs with the AGILE package), it needs to be packaged into a plugable library, a plugin. To get this done, the RIVET package has a function `rivet-buildplugin` which produces a .so-file, a library format well known to UNIX users. The plugin filename always has to start with 'Rivet'. To create `Rivet-ANALYSIS_2011_01.so` based on C++ file `ANALYSIS_2011_01.cc` one has:

```
rivet-buildplugin Rivet-ANALYSIS_2011_01.so ANALYSIS_2011_01.cc
```

Once the analysis is built, the process of running the event generator and simultaneously running the analysis can commence. Starting a PYTHIA 6.4 run through the AGILE package can be done with:

```
agile-runmc Pythia6:423 --beams=e+:27.6,p:920 -n 1000000 -s 2 \
    -P paramfile -o hepmc.fifo &
```

The first parameter indicates that PYTHIA 6.4 is being used. The –beams clause specifies e^+p-collisions and the particles' energies. Parameters n and s are the number of events and the random number seed, the rest of the parameters (for PYTHIA) are stored in a parameter file. We used the parameters from section 12. Finally -o indicates the output file, in this case through a fifo pipe which can be created with `mkfifo hepmc.fifo`. The ampersand sends the `agile-runmc` function to the background, making the command line free for the user to call a RIVET analysis.

Calling the analysis we described uses the command `rivet`. We run the analysis on the .fifo-file, directly performing the analysis of the PYTHIA events while they are being generated. Alternatively one could write a run.hepmc file with PYTHIA and than after the run is finished, perform the analysis on that .hepmc-file. With a .fifo-file the command is:

```
rivet --analysis=ANALYSIS_2011_01 hepmc.fifo
```

The output of the RIVET analysis comes in the form of a file `Rivet.aida`, containing the filled histograms in AIDA format.

13.3 Studying the output of the analysis

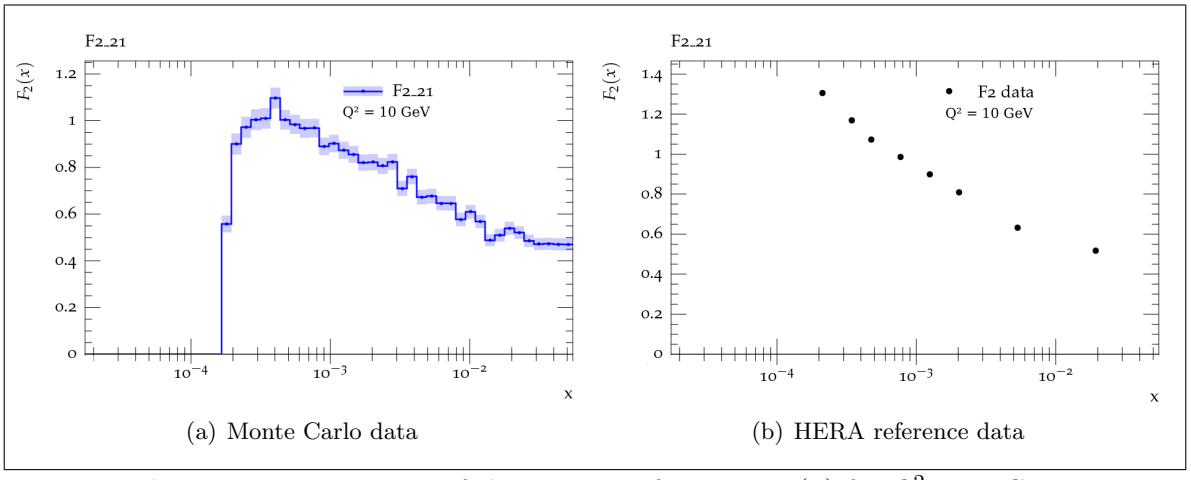
Another function in the RIVET system, `rivet-mkhtml`, allows to quickly generate plots of the AIDA histograms. It produces .dat-files and .png-images starting from the AIDA data. Plots can be generated seperately for either the Monte Carlo run data or the HERA reference data, or in a combined way to make a comparison between both. In the former case no further processing is necessary, the .png-images are there. In the latter case one would use the `compare-histos` and `make-plots` functions also included in RIVET to make the combined plots.

When using the RIVET functions to make plots, one has the option to include a .plot-file with the same name as the .aida-file. This file would include plotting options for a histogram with a certain name: axis names, axis limits, linear/logarithmic settings, ... As an example:

```
# BEGIN PLOT /ANALYSIS_2011_01/F2_01
Title=F2_01
LogX=1
LogY=0
XLabel=x
YLabel=$F_{2}(x)$
ErrorBars=1
ErrorBands=0
# END PLOT

# BEGIN PLOT /ANALYSIS_2011_01/F2_02
...
```

In figure 38 we show the $F_2(x)$ histograms (for $Q^2 \approx 10$ GeV) for both the Monte Carlo and the HERA reference data to illustrate the output of the RIVET analysis. In the next section we will show comparison plots between Monte Carlo and reference data, to check the RIVET analysis and the PYTHIA 6.4 runs.



(a) Monte Carlo data (b) HERA reference data

Figure 38: Histograms of the structure function $F_2(x)$ for $Q^2 \approx 10$ GeV

13.4 Check of the PYTHIA setup and the RIVET analysis

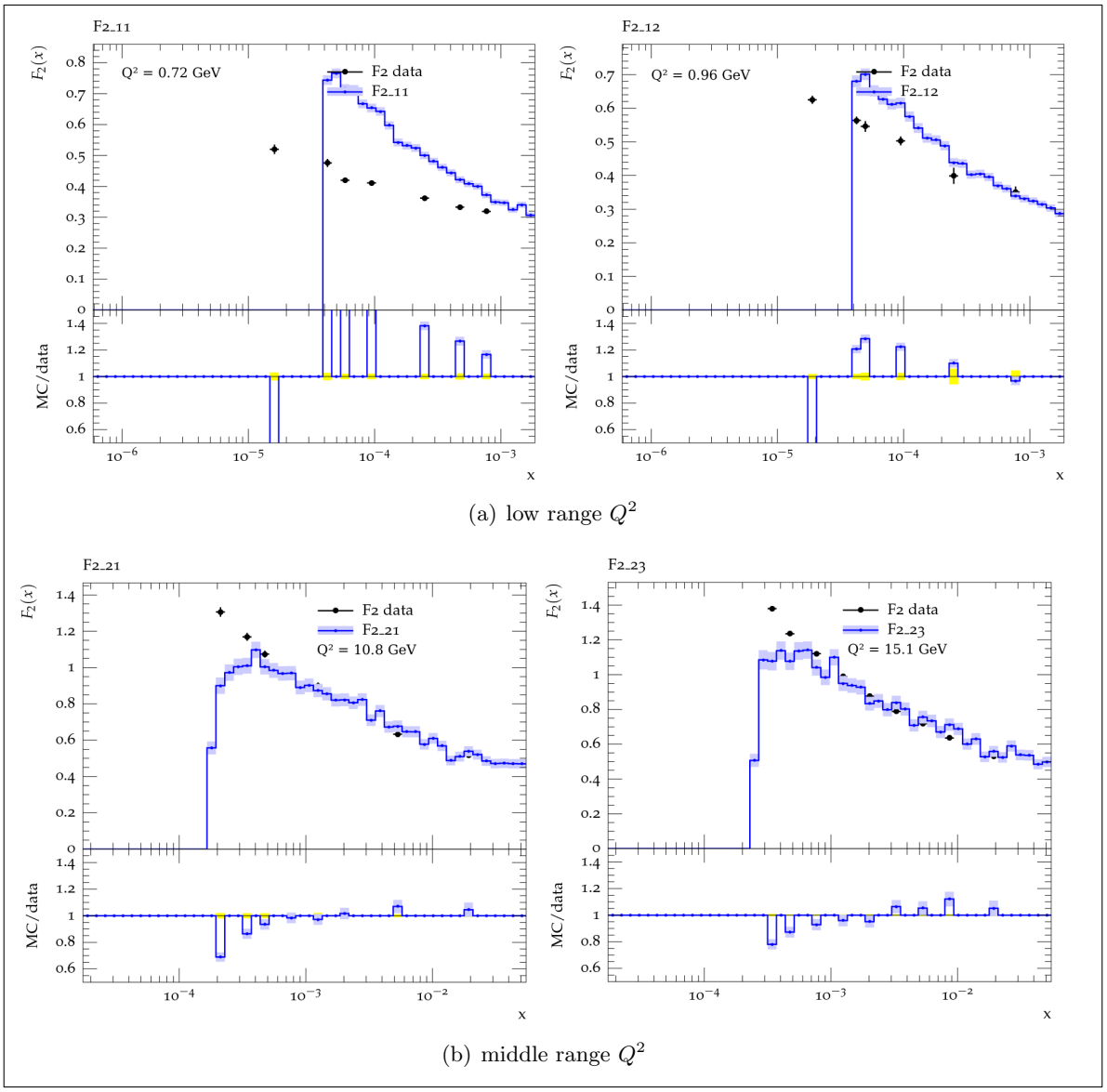
Because it is hard to judge the reliability of the PYTHIA runs and the correctness of a RIVET analysis thereof if we are working with a parametrization that has not been tuned yet, we will first produce a Monte Carlo data set with PYTHIA using the `HERAPDF01.LHpdf` LHAPDF PDFset and analyse that. This PDFset has been tuned to HERA data and therefore the Monte Carlo result using it should match the HERA data well. We generated events between $Q^2 = 0.5$ GeV and $Q^2 = 27$ GeV, in the first two histogram ranges. Making the same comparison for the third range would simply be a third repetition of the same CPU intensive process, with different histogram settings. Note that fixing the Q^2_{min} scale in PYTHIA on a value below 1 GeV requires changing a couple of extra PYTHIA switches. Going below 1 GeV however also brings us in a region that is not really perturbative any longer. We therefore don't expect the results for the lowest Q^2-values to be very good. We performed runs with 750.000 events, for our 50×14 grid.

Below he extra parameters that need to be set. Note that they were only fixed to force PYTHIA to produce test results in the lowest Q^2-range. For all other runs, in the middle Q^2-range, the parameters from section 12 suffice.

PYTHIA 6.4 parameter	value set	explanation
CKIN(35)	0.5	minimal Q^2
CKIN(3)	0.5	minimal p_T
CKIN(5)	0.5	minimal p_T override
PARP(15)	0.5	lower p_{cut}
PARP(62)	0.5	effective Q^2_{cut} or $p_{T,cut}$
MSTP(66)	0	lower Q^2_{cut} = PARP(62)

In figure 39 we show some results of the comparison between the Monte Carlo data (PYTHIA with `HERAPDF01.LHpdf`) and the HERA reference data, for two values in the lowest Q^2-range (39) and two values in the middle Q^2-range (39(a)). The $F_2(x)$ histograms are numbered with the Q^2-bin numbers also indicated on scheme 37.

From the plots in figure 39(a) we can clearly judge that indeed the reliability below $Q^2 = 1$ GeV is bad. This is why in the next section, where we look further at parametrization, we will neglect the first histogram range and focus on the second one. The other plots, in figure 39(b) show a better match between the Monte Carlo and the reference data, although not entirely overlapping. We shall have to see whether our other parametrizations achieves better results. This is discussed in section 14.



(a) low range Q^2

(b) middle range Q^2

Figure 39: Comparison plots for $F_2(x)$ from MC and HERA data

14 Tuning with Professor

14.1 Introduction

For the optimization of our parametrization, we will have to make some choices. As suspected before, it is not realistic within a reasonable time frame to use twenty parameters, nor to generate events in the entire (x, Q^2) range at once. Since we learned in the previous section that the very small Q^2 events cannot be generated reliably due to their non-perturbative character, it's not a farfetched step to dismiss the first Q^2 range as given in figure 37. From the other two ranges, we chose to work with the middle one (Q^2 = 1.2 - 27 GeV), which matches with smaller x-values. These smaller x-values allow for us to ignore valence quarks (which are important at higher x), and in that way simplify any parametrization.

For example for the parametrization explained in section 11.1 we could keep only three functions (S, g and Δ), and limit the parametrization to three parameters per function, leaving us with nine parameters. Within Professor there is the choice between quadratic and cubic interpolation. This matches with an absolute minimum number of runs of 55 and 220. We have a grid with 700 bins (50×14), and could reasonably generate 750.000 events per run to go with that. At approximately half an hour per run, this is still very time consuming. A choice could therefore be to stick with quadratic interpolation, for which one can then attempt to reach a better sampling ratio, defined as the total number of runs N_{tot} divided by the minimum number of runs N_{min}. A value of at least two would be good.

All this said, starting an optimization procedure with Professor for a parametrization with nine parameters, without any prior knowledge about the parameters and the sensitivity of the result to these parameters, would not really be a good idea and probably wouldn't produce conclusive results even if one made the effort of generating enough data to work with. Instead of working with an entirely new parametrization, with many parameters, we will start from an existing one, looking at the effect of changing just one or two of its parameters and testing whether or not it can be optimized. An important note here is that, as Pythia works at leading order, the best result will be gained if we also select a leading order parametrization to work with. We shall work with CTEQ6ll.LHpdf. This is not yet a tuning procedure for which the Professor package is really needed, but we might as well learn to work with it now. Before we show any results, we will comment on some options and function of the package.

14.2 Sampling parameters

To sample parameters in a large parameter space, which would be rather tricky to do manually, we can make use of Professor's function `prof-sampleparams`. This function takes as input the number of combinations to generate N, a file with parameter ranges and optionally a template file to write the resulting parameters to. The output is a file with a list of parameters and then optionally a copy of the template file with the parameters filled in in the correct places. Depending on the given number of combinations, several run folders with a number `xxx` are produced, each containing a parameter list and a filled in template file. The structure of the run folders matches the folder structure needed for a Professor fit with `prof-tune`, saving the user the worry of having to move around files and possibly mix them up.

In the case of tuning a PDF parametrization, the parameters that need to be sampled are the ones in the .LHpdf-file describing the parametrization. Because one needs to perform a Pythia run for each parameter combination and need an adapted .LHpdf-file for each of those runs, good use can be made of the template system.

Documentation on both the Rivet and Professor systems is rather sparse, but luckily the required file compositions for the range- and template files can be guessed by having a look at the code that is supposed to read them. For the range-file one needs a parameter name, a lower limit and un upper limit. The parameter names can be more complicated and informative, but a simple `Pxx` does just fine too:

```
P01     llim    ulim
P02     llim    ulim
...
```

Next for the template files, one can use two tags to indicate the run number or a parameter: `<__RUNID__>` and `<__PARNAME__>`. Note that there are double underscores each time and that RUNID needs to be capitalized. The parameter names need to be the same as in the range-file. With the .LHpdf-format in mind one could produce something like:

```
'Description:'
'.LHpdf-file for Pythia 6.4 run <__RUNID__>'
...
'Parameterlist:'
'list',1,6
<__P01__> <__P02__> <__P03__> <__P04__> <__P05__> <__P06__>
<__P01__> <__P02__> <__P03__> <__P04__> <__P05__> <__P06__>
'End:'
```

Finally the Professor function can be run. Parameters will be sampled randomly from the hypercube that makes up the parameter space, with limits according to the range-file. Output will be written to a given directory.

```
prof-sampleparams -o mc/ -T template.LHpdf ranges.txt
```

14.3 Studying reference data enclosure by the generated data

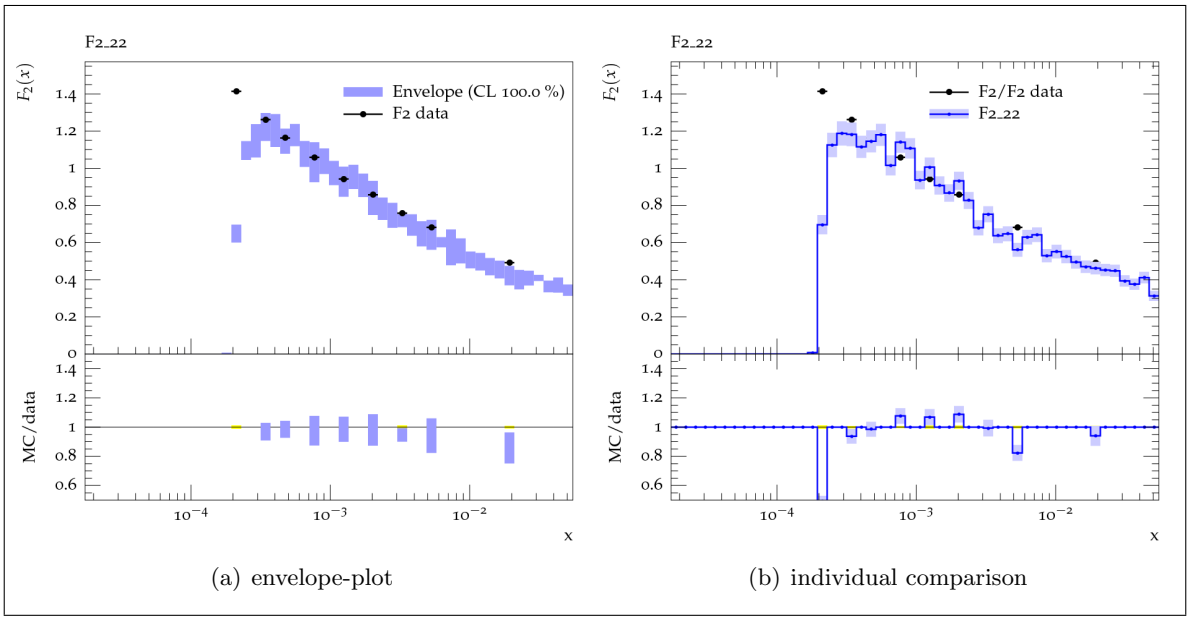
After choosing parameter ranges (once), sampling parameter points (N times) and running the event generator for each set, one is left with N Monte Carlo data sets. Each of those can be compared with the reference data, for example by running the previously described `compare-histos` and `make-plots`. However, such individual comparisons don't give a very good picture of how well the generated data encompasses the reference data. For some runs the results will be too high or too low, while in other cases the shape may be off. We want one plot which shows the extrema of the generated data around the reference data, to judge whether or not there seems to be a possibility for a fit. For example it may be that the initially chosen parameter ranges are too small, or wrong, and that the reference data falls (partially) outside the extrema of the generated data. If this were the case, it is usefull to see it before spending time on trying to optimize the parameters. The type of plots we want to have can be made with the Professor function `prof-envelopes`.

The function scans all the generated data and produces plots showing the reference data and an envelope (hopefully around it and not to the side) indicating the range encompassed by the generated data. A plot is made for each Rivet histogram available in both the reference and the generated data. In our case these would be the fourteen $F_2(x)$ histograms in the

middle Q^2 range. Note that the histograms don't need to be limited to one observable, one could use many different observables in one fit, as long as they are sensitive to the parameter(s) one wants to optimize. In this light it is also possible to attach a weight to a histogram, telling PROFESSOR that not all histograms are equally important. This may be the case if some observables are more important than others, or for example for our set of $F_2(x)$ histograms, if part of the Q^2-range is judged more important. The weights go in a file in the format ``histname weight''. The `prof-envelopes` function takes the following arguments:

```
prof-envelopes --datadir . --weights weights.txt --outdir ../envelopes
```

An example of an envelope plot around the reference data and a comparison of an individual run with the reference data are shown in figure 40. Both are taken from the test we did with the CTEQ6ll.LHpdf set, which will be described in section 14.6. It is already clear from this that the statistical fluctuation in the generated event data will be a problem for any attempt to optimize a parameter.



(a) envelope-plot (b) individual comparison

Figure 40: An envelope plot around HERA reference data next to a comparison of a single event with HERA reference data

14.4 Parametrizing the generator response

As mentioned in the introduction on the PROFESSOR package (section 4.1.4), the response of the PYTHIA 6.4 runs to changes in the input parameters will be interpolated in the process of optimizing the input parameters. To do this, the package has a function `prof-interpolate`. However, before we get to this interpolation, it is necessary to define so-called run combinations. We have a set of runs which can be used by the interpolation and tuning functions. Of course we could use all the runs at once, but the end result will be better if we can find a common result between several different run combinations. However, this means that we will run the interpolation on reduced sets of runs. As it is not interesting to use too few runs (small sample ratio), leaving out runs and making multiple run combinations is only interesting when there are enough runs to begin with. Run combination files can be produced using:

```
prof-runcombs --datadir . -o runcombs.txt -c 0:1 -c 15:24
```

This way, one file will be produced with 25 run combinations. The first one is simply the complete set, nothing left out, the other 24 are random combinations of the total set minus 15 runs. Once we have a run combinations file, we can run the interpolation. One also has to make the choice of quadratic or cubic interpolation here. The same option will need to be selected when doing actual tuning. The interpolation function will also take into account eventual previously defined weights for the observables. An interpolation will be produced for each of the run combinations in the given file. To start the interpolation(s) one has:

```
prof-interpolate --datadir . --ipol=quadratic --runs runcombs.txt
      --weights weights.txt --outdir .
```

The produced output (in the form of .pkl-files) is not interesting for us to look at, but it will be used by the tuning function which we describe in the next section.

14.5 Tuning to reference data

Actual tuning is done by the function `prof-tune`. Next to the location of the data, the user needs to specify the interpolation method used by `prof-interpolate`, which needs to have been succesfully run before any tuning can be performed. Also, there is the option of selecting a starting point for the optimalization, the initial parameter point for the built-in minimizing function. By default it will be the center of the hypercube representing the parameter space, but it can also be chosen randomly from the hypercube by Professor or given manually by the user. Sometimes running several of the options can provide extra information. Finally if there were multiple runcombinations or non-default weights, those too need to be specified. An example for the tuning function:

```
prof-tune --datadir . --ipol=quadratic --runs runcombs.txt
      --weights weights.txt --outdir . --spmethods=center,random
```

This will produce output in a `tunes` folder, with a subfolder for each runcombination. The `results.pkl` output file will have an entry for each found minimization result, with a matching χ^2-value. Again the .pkl-file itself isn't really clear information for the user, but luckily there is a function in the package to extract the relevant information from it and show it in standard text format:

```
prof-showminresults tunes/results.pkl
```

A last interesting function in the package is `prof-plotresultscatter`. If multiple run combinations were studied, or multiple starting points provided different minimization results, a plot can be made of the goodness of fit in function of the parameter value. From such a plot one could identify cases where the minimizer gets stuck in a local minimum, or see whether the result isn't too dependent on the runcombination. For a good minimization, the different results from different run combinations should cluster around the result obtained with all the runs. The scatter result will be bad if for example the observables aren't sensitive to the varied parameter, or if the parametrization doesn't properly describe the observables. Linked to that latter case, event data with too high statistical fluctuation would cause trouble too, as it also can't properly describe the observables.

```
prof-plotresultscatter tunes/results.pkl
```

14.6 Study with the CTEQ6ll.LHgrid parametrization

We again do this test with the CTEQ6ll.LHgrid parametrization in the middle grid region as defined in figure 37, producing 750.000 events per PYTHIA run for the 50×14 grid. To begin with we made a comparison of the results for PYTHIA run without initial state radiation, final state radiation and fragmentation, and the results with those three switched on. Our parameter file used:

PYTHIA 6.4 parameter	value set	explanation
MSEL	0	user selected process
MSUB(10)	1	DIS events
MSTP(11)	0	no electron radiation
MSTP(21)	4	only charged current interactions
MSTP(52)	2	user selected external pdf
MSTP(51)	10042	CTEQ6ll external pdf
MSTP(61)	0/1	initial state radiation
MSTP(71)	0/1	no final state radiation
MSTP(81)	0	no multiple parton-parton interactions
MSTP(91)	0	intrinsic $k_T = 0$
MSTU(101)	0	fixed α_{em}
MSTJ(1)	0/1	fragmentation
CKIN(35)	1.09	minimal Q^2
CKIN(36)	31.65	maximal Q^2

The results without ISR/FSR/fragmentation are plotted in the left hand column of figure 41, the results with ISR/FSR/fragmentation are plotted in the right hand column of the same figure. Two plots next to each other horizontally are always made for the same Q^2-range, with the F2_xx numbers again referring to the Q^2 bin numbers from figure 37.

Comparing both columns it is clear that there is a difference and that the case with ISR/FSR/ fragmentation produces the best results. For this case we next produced a series of runs with a variation in the first gluon parameter in the CTEQ6ll PDFset and attempt to tune this parameter with PROFESSOR. All steps can be taken, which we show below, but we can also expect that the result will not be conclusive, mainly due to bad statistics for our generated event data.

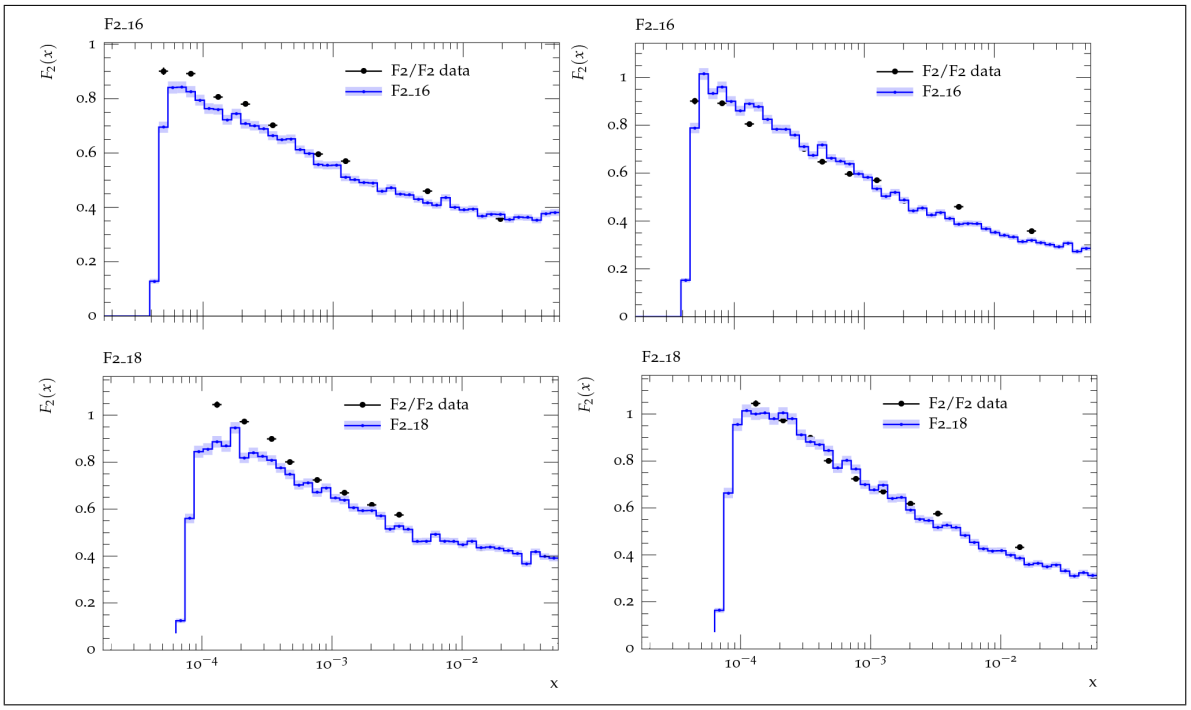
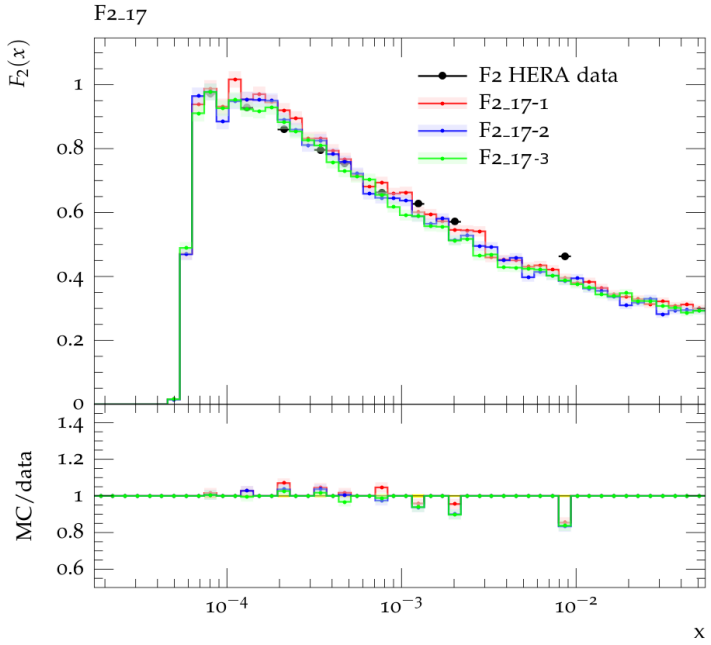


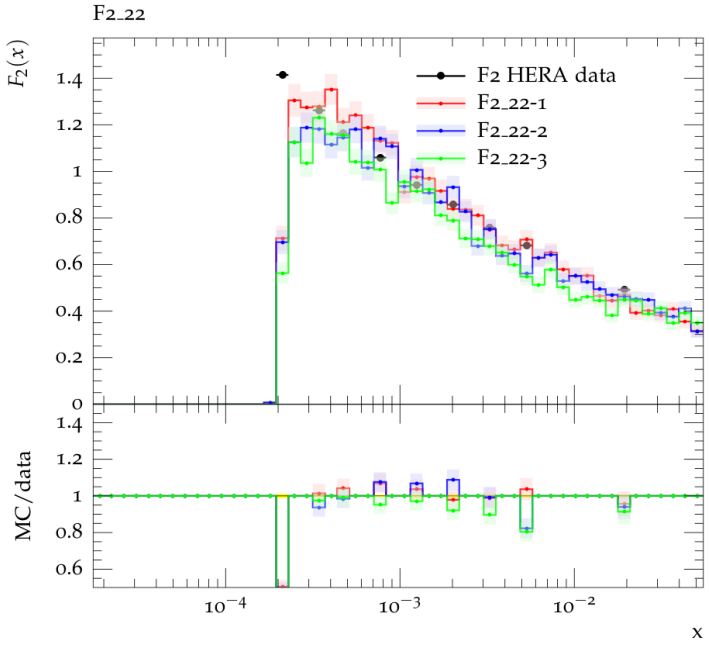
Figure 41: Comparison of PYTHIA results obtained with CTEQ6ll with the HERA reference data

The parameter we varied had an initial value of 2.445. We allowed values between 2.15 and 2.75. If we make a comparison plot for three runs, respectively G01=2.15 (red), 2.45 (blue) and 2.75 (green), we get figure 42. As remarked repeatedly, statistic really are too bad to state anything reliably, but still there's a tendency of green $<$ blue $<$ red. This indicates that the result is indeed sensitive to variations in parameter G01.

Finally we ran a PROFESSOR tuning procedure for a total of nine runs with G01 between 2.15 and 2.75. We used cubic interpolation, and made ten run combinations, one with all the runs and nine with eight of the nine runs. Minimizing twice per run (once from the center or the parameter space and once randomly), we get twenty minimization results and the `prof-showminresults` indicates average: G01$= 2.449 \pm 0.160$. The minimization result using all runs was G01$= 2.419$. While the results are not as good as one would like, they are not entirely nonsensical either. The found minimum matches with the actual CTEQ6ll value (G01$= 2.445$), which is not bad given the rather large variable range we allowed. In all, we can't conclude much more here, but it is clear that there are options for improvements, for example improving the used grid, the PYTHIA 6.4 statistics and looking at other/more variables.



(a) F2_17



(b) F2_22

Figure 42: Comparison of three PYTHIA results obtained with CTEQ6ll for different G01 values

Conclusion

We started this work on parton density functions by setting up event generation with Monte Carlo event generator Pythia 8.1 and looking at ways to extract both integrated and unintegrated parton density functions from those events. Helper functions were constructed to retrieve the parton transverse momenta from the event data. The attempt to get the unintegrated parton density functions failed because we ran into trouble with factorization and integrating. We learned that what we were looking at were actually parton luminosity functions, products of parton density functions. For the integrated distributions we could work around the difficulties and managed to extract actual parton density functions next to the luminosity functions.

After studying results from Pythia 8.1, we looked for alternative sources for the same functions. A first one was the LHAPDF interface, allowing us to both extract integrated parton density functions from a PDFset of our choice and to build parton luminosity functions with them. A second alternative approach was the Kimber-Martin-Ryskin approach, which supplied both integrated and unintegrated parton density functions. To get to them, we calculated Sudakov form factors and performed integrations as described in the Kimber-Martin-Ryskin paper, using the LHAPDF interface as a source of integated parton density functions in the calculations of unintegrated ones. The third and last alternative source of parton density functions, only for the gluon, was a CCFM dataset which we accessed using an interpolation function.

The first part ended with a comparison of all these different parton density and luminosity functions. All of them were produced taking care to use the same PDFset (CTEQ5l) and scales whenever possible. For the quark density functions we could put Pythia 8.1, LHAPDF and KMR results next to each other. They matched reasonably well, only the KMR result being a bit larger at large x. The cause of this difference could be found in the evaluation of α_s at a different scale in the KMR approach. For the gluon density function we did not have a Pythia 8.1 result, but with the added CCFM result there were still three entries in the comparison. Again there was the difference between the LHAPDF and KMR results, with the CCFM result matching the KMR one best. Finally for the parton luminosity functions we compared results from Pythia 8.1 and LHAPDF. The match was very good in all cases, with a too steep drop at the smallest x for the Pythia 8.1 gluon result as the only remarkable point. This could be explained by the simplification of a Breit-Wigner distribution with a delta-function in our treatment of the Pythia 8.1 events.

In the second part we looked at electron-proton events as opposed to the proton-proton events studied in the first part, learning early on that this required a switch from Pythia 8.1 in C++ to Pythia 6.4 in fortran, because the former event generator doesn't support electron-proton (Deep Inelastic Scattering) events. We used this event type with the idea to use the HERA combined results as reference data to perform Pdf4mc for Pythia; to optimize parton density functions for Pythia by constructing a parametrization and tuning to the reference data.

The first hurdle was to find a grid on which we could represent the HERA data without losing too much detail, but which wasn't too fine either, keeping in mind the number of Monte Carlo generated events needed tot get reasonable statistics. We made certain choices, working with different ranges, but this is a point where a lot of improvement is possible.

The next point was to get the data, both reference and generated, in a format which could be understood by the PROFESSOR tuning package. For the reference data we just performed a crude, unportable conversion from the data table in text format to the required AIDA format, for the generated data the AIDA data was acquired as the output of a RIVET analysis. Such an analysis tracks the generated event data as it is being produced, calculates the relevant observables and produces histograms for them according to the specified grid and in the correct format. In our case we extracted histograms of proton structure function F_2 from the data. Also in the second part we explained the methods of parametrizing parton density functions using an LHAPDF PDFset, and we tested our RIVET analysis using the HERAPDF01 PDFset.

Finally we studied results obtained from PYTHIA 6.4 using the leading order CTEQ6ll PDFset and running our RIVET analysis for F_2. We compared the match with the HERA reference data switching initial state radiation, final state radiation and fragmentation on and off and, concluding that the former option gives better results, performed a series of runs using those settings and varying one gluon PDF parameter. The effect of this variation was studied and the parameter was tuned using PROFESSOR. While the result was not as nice as one would like and looking at one parameter is not yet really an implementation of PDF4MC, there are enough indications that improvements are possible and that this system is not without merit.

An actual PDF4MC tuning of a PDF for the PYTHIA 6.4 Monte Carlo generator would require more work, looking at multiple parameters and optimizing them together, after checking the response to changes in each of them individually. Such a procedure would make use of all the elements we described: getting reference data, choosing a parametrization, generating events, building and performing RIVET analyses and tuning with the PROFESSOR package. We did not encounter fundamental problems in any of the steps, indicating that given more time, it should be entirely possible to perform full PDF4MC.

Samenvatting

Dit werk rond parton dichtheidsfuncties werd aangevat met het instellen van Monte Carlo event generator PYTHIA 8.1 voor het aanmaken van proton-proton events en het zoeken naar manieren om zowel geïntegreerde als ongeïntegreerde parton dichtheidsfuncties uit de event data af te leiden. Om de transversale impuls van de partonen, welke niet in the standaardinformatie vervat is, te bepalen, werden helper-functies gemaakt. Pogingen om ongeïntegreerde parton dichtheidsfuncties af te leiden waren onsuccusvol. Een probleem daarbij was het niet volledig factorizeren van de functies wat problemen oplevert bij benodigde integraties. Het bleek dat de bekomen distributies in feite overeenkwamen met parton luminositeitsfuncties in plaats van met dichtheidsfuncties. Deze luminositeitsfuncties zijn producten van dichtheidsfuncties. Voor de geïntegreerde parton dichtheidsfuncties vonden we de mogelijkheid om het factorizatieprobleem te omzeilen en toch de dichtheidsfuncties te bepalen, naast ook weer de luminositeitsfuncties.

Na het analyseren van PYTHIA 8.1 resultaten werd nagegaan welke alternatieve bronnen beschikbaar waren om dezelfde functies te bekomen. Een eerste alternatief is de LHAPDF interface welke ons toelaat enerzijds geïntegreerde parton dichtheidsfuncties te bekomen uit een PDFset naar keuze en deze anderzijds te gebruiken luminositeitsfuncties terug te vinden. Een tweede alternatief is de Kimber-Martin-Ryskin aanpak, waaruit zowel geïntegreerde als ongeïntegreerde parton dichtheidsfuncties volgen. Om deze te bekomen werden Sudakov vormfactoren berekend en integralen bepaald volgens de procedure beschreven in het Kimber-Martin-Ryskin artikel. Daarbij werd ook opnieuw de LHAPDF interface gebruikt als bron van geïntegreerde parton dichtheidsfuncties bij het bepalen van ongeïntegreerde. De derde en laatste alternatieve bron van parton dichtheidsfuncties, enkel voor het gluon, was een CCFM dataset, toegankelijk gemaakt door een interpolatiefunctie.

Het eerste gedeelte werd beëindigd met een vergelijking van alle bekomen parton dichtheids- en luminositeitsfuncties. Bij deze bepalingen werd trouwens ook telkens goed opgelet om indien mogelijk steeds dezelfde PDFset (CTEQ5l) en schalen te gebruiken. Voor de quark dichtheidsfuncties konden resultaten bekomen met PYTHIA 8.1, de LHAPDF interface en de KMR methode naast elkaar gezet worden. Ze bleken goed overeen te komen, al was er een kleine afwijking te zien voor het KMR resultaat bij grote x-waarden. Dit kon verklaard worden doordat in the KMR methode enkele evaluaties van α_s voorkomen bij andere schalen dan in de normale gevallen. Voor de gluon dichtheidsfunctie was er geen PYTHIA 8.1 resultaat, maar wel het extra CCFM resultaat, zodat er toch weer drie resultaten te vergelijken waren. Opnieuw was er een verschil tussen LHAPDF en KMR, om dezelfde reden, en CCFM kwam best overeen met het KMR resultaat. Tenslotte werden de parton luminositeitsfuncties bekomen via PYTHIA 8.1 en via de LHAPDF interface vergeleken. Voor alle partonen was de vergelijking goed, alleen voor het gluon is er bij de allerkleinste x-waarden een afwijking in het PYTHIA 8.1 resultaat, de functie daalt te snel in vergelijking met het LHAPDF resultaat. Dit laatste kan worden verklaard door het feit dat we een vereenvoudiging maakten in onze verwerking van de PYTHIA 8.1 events, er werd een Breit-Wigner distributie vervangen door een delta-functie.

In het tweede gedeelte werd gekeken naar elektron-proton botsingen in plaats van de proton-proton botsingen uit het eerste deel. Hierbij moest al snel worden vastgesteld dat we noodzakelijkerwijze moesten werken met PYTHIA 6.4 in fortran in plaats van met PYTHIA 8.1 in C++, aangezien deze laatste geen elektron-proton botsingen (diep inelastische verstrooiing) ondersteunt. We schakelden over op dit botsingsprocess met het idee om HERA data

als referentie te gebruiken en PDF4MC uit te voeren voor PYTHIA. Dit houdt in dat parton dichtheidsfuncties geoptimaliseerd worden voor PYTHIA door gebruik te maken van een parametrisatie en deze dan te tunen aan de referentiedata.

De eerste stap die uitgevoerd moest worden was het kiezen van een grid die geschikt was om de HERA data weer te geven zonder al te veel detail te verliezen, maar die tegelijkertijd ook niet te fijn was, aangezien rekening moet worden gehouden met het aantal events dat gegenereerd moet worden om voldoende statistiek te bekomen bij Monte Carlo runs. Er werden bepaalde keuzes gemaakt, onder ander het verdelen van de grid in een aantal bereiken, maar in elk geval is dit een onderdeel waar verbetering zeker mogelijk is.

Het volgende punt bestond erin zowel de referentiedata als de gegenereerde data om te kunnen zetten in een formaat dat bruikbaar is voor het PROFESSOR pakket voor optimalisatie. Voor de referentiedata werd de conversie naar het AIDA formaat eenmalig gemaakt op een nogal onverfijnde manier, voor de gegenereerde data wordt rechtstreeks AIDA data bekomen een een door ons geschreven RIVET analyse. Zo'n analyse onderzoekt de gegenereerde data meteen op het moment dat ze gegenereerd wordt, berekent relevante observabelen en creëert vervolgens histogrammen voor deze observabelen, in het juiste formaat en volgens de op voorhand gespecifieerde grid. In ons geval bestond de output uit histogrammen voor proton structuurfunctie F_2. Nog in het tweede gedeelte werd gekeken naar de wijze waarop parametrisaties voor parton dichtheidsfuncties gemaakt worden met behulp van een LHAPDF PDFset en werd onze RIVET analyse getest aan de hand van de HERAPDF01 PDFset.

Als laatste punt werden resultaten bekomen met PYTHIA 6.4 en de eerste orde CTEQ6ll PDFset bestudeerd, na het uitvoeren van onze RIVET analyse voor F_2. Eerst werden initial state radiation, final state radiation en fragmentie uitgeschakeld, vervolgens ingeschakeld. In beide gevallen werd de vergelijking gemaakt met de HERA referentiedata. Deze is duidelijk beter met de effecten ingeschakeld in de event generator. Met deze laatste instellingen werd verdergewerkt; er werd een aantal runs geproduceerd voor verschillende waarden van een gluonparameter uit de gebruikte PDFset. Het effect van die variatie werd geanalyseerd en de parameter werd gefit met het PROFESSOR pakket. Het resultaat was niet ideaal, en de uitgevoerde test met een parameter is ook nog niet echt een volledige implementatie van het PDF4MC idee, maar het is wel duidelijk dat nog op allerlei vlakken verbetering mogelijk is en dat dit idee niet onverdienstelijk is.

Effectief tunen van een PDF voor de PYTHIA 6.4 Monte Carlo generator vereist tenslotte meer werk dan wat hier werd uitgevoerd. Er moet gekeken worden naar meerdere parameters en deze dienen samen geoptimaliseerd te worden, nadat eerst voor elk afzonderlijk het effect van variatie is nagekeken. Zo'n procedure zou gebruik maken van alle onderdelen die we in dit werk beschreven hebben: referentiedata vinden, een parametrisatie kiezen, events genereren, RIVET analyses uitvoeren en tunen met PROFESSOR. We hebben bij geen enkele van de stappen fundamentele problemen ondervonden bij tests, wat aangeeft dat met meer (reken)tijd, volledige PDF4MC tuning perfect mogelijk moet zijn.

Bibliography

[1] V. Barone and E. Predazzi. *High-Energy Particle Diffraction*. Springer, 2002.

[2] J. Collins, M. Diehl, H. Jung, L. Lönnblad, M. Lublinsky, and T. Teubner. Unintegrated parton density functions. 2005.

[3] E. Eichten, I. Hinchliffe, K. Lane, and C. Quigg. Supercollider physics. *Review of Modern Physics*, 56(4):579, 1984.

[4] R.K. Ellis, W.J. Stirling, and B.R. Webber. *QCD and Collider Physics*. Cambridge University Press, 1996.

[5] W. Greiner and J. Reinhardt. *Quantum Electrodynamics*. Springer, 4th edition, 2009.

[6] D. Griffiths. *Introduction to elementary particles*. Wiley & Sons, 1987.

[7] H1 and ZEUS Collaboration. Combined measurement and qcd analysis of the inclusive e+-p scattering cross sections at hera. *arXiv*, hep-ex:09110884v2, 2010.

[8] F. Halzen and A.D. Martin. *Quarks and Leptons*. Wiley & Sons, 1984.

[9] M.A. Kimber, A.D. Martin, and M.G. Ryskin. Unintegrated parton distributions. *Physical Review D*, 63:114027, 2001.

[10] M. Kuhlen. *QCD and the Hadronic Final State in Deep Inelastic Scattering at HERA*. PhD thesis, Max-Planck-Institut für Physik, 1997.

[11] H.L. Lai, J. Huston, S. Mrenna, P. Nadolsky, D. Stump, W.-K. Tung, and C.P. Yuan. Parton distributions for event generators. *arXiv*, hep-ph:09104183v3, 2010.

[12] M. Lüscher. A portable high-quality random number generator for lattice field theory simulations. *Computer Physics Communications*, 79(1):100, 1994.

[13] A.D. Martin. Proton structure, partons, QCD, DGLAP and beyond. *arXiv*, hep-ph:08020161v1, 2008.

[14] F. von Samson-Himmelstjerna. *Determination of parton density functions using Monte Carlo event generators*. PhD thesis, Freie Universität Berlin, 2009.

[15] K. Wille. *The Physics of Particle Accelerators*. Oxford University Press, 2000.

[16] AGILe overview. http://projects.hepforge.org/agile/.

[17] AIDA xml format documentation. http://aida.freehep.org.

[18] CCFM datasets. https://www.desy.de/~jung/cascade/updf.html and http://arxiv.org/abs/hep-ph/0411287.

[19] A. Lipatov. CCFM dataset interpolation function nr_updf, 2008.

[20] CMS images. http://w3.iihe.ac.be/cms/data/Pictures.

[21] CMS overview. http://cms.web.cern.ch/cms/Detector/FullDetector/index.html.

[22] Desy homepage and HERA pages. www.desy.de, www-h1.desy.de, www-zeus.desy.de, hasylab.desy.de/images/content/e8/index_eng.html.

[23] M. Flechl. The underlying event in hadronic collisions. http://www.isv.uu.se/thep /courses/QCD/QCD_presentation_Martin.pdf.

[24] HERA Combined Results. https://www.desy.de/h1zeus/combined_results/index.php ?do=proton_structure.

[25] H. Jung. QCD and Monte Carlos Lecture Course. http://www.desy.de/∼jung/qcd_ and_mc_2010/QCD_and_MC_2010_

[26] Les Houches Accord. http://projects.hepforge.org/lhapdf/.

[27] LHC images. http://faculty.physics.tamu.edu/kamon/research/refColliders/LHC.

[28] OpenMP framework. http://openmp.org/wp.

[29] OpenMP framework. https://computing.llnl.gov/tutorials/openMP.

[30] Professor 1.2.1 overview. http://projects.hepforge.org/professor/.

[31] Pythia 6.4 manual. http://home.thep.lu.se/ torbjorn/pythia/lutp0613man2.pdf.

[32] Pythia 8.1 overview. http://home.thep.lu.se/ torbjorn/php8145/Welcome.php.

[33] Rivet 1.5.0 overview. http://projects.hepforge.org/rivet/.

[34] Root overview. http://root.cern.ch/drupal/.

[35] Root user's guide. http://root.cern.ch/root/doc/RootDoc.html.

[36] T. Sjöstrand. Various lectures on monte carlo and event generators. http://home.thep. lu.se/∼torbjorn/welcomeaux/talks.html.