

SOLVING QCD EVOLUTION EQUATIONS IN RAPIDITY SPACE WITH MARKOVIAN MONTE CARLO*

K. GOLEC-BIERNAT^{a,b}, S. JADACH^{a,c}, W. PŁACZEK^d, M. SKRZYPEK^{a,c}

^aThe H. Niewodniczański Institute of Nuclear Physics PAN
Radzikowskiego 152, 31-342 Kraków, Poland

^bInstitute of Physics, University of Rzeszów
Rejtana 16A, 35-959 Rzeszów, Poland

^cCERN, PH Department, 1211 Geneva 23, Switzerland

^dMarian Smoluchowski Institute of Physics, Jagellonian University
Reymonta 4, 30-059 Kraków, Poland

(Received November 17, 2008)

This work covers methodology of solving QCD evolution equation of the parton distribution using Markovian Monte Carlo (MMC) algorithms in a class of models ranging from DGLAP to CCFM. One of the purposes of the above MMCs is to test the other more sophisticated Monte Carlo programs, the so-called Constrained Monte Carlo (CMC) programs, which will be used as a building block in the parton shower MC. This is why the mapping of the evolution variables (eikonal variable and evolution time) into four-momenta is also defined and tested. The evolution time is identified with the rapidity variable of the emitted parton. The presented MMCs are tested independently, with $\sim 0.1\%$ precision, against the non-MC program `APChEb` especially devised for this purpose.

PACS numbers: 12.38.-t, 12.38.Bx, 12.38.Cy

1. Introduction

The problem of solving numerically the so-called evolution equations of the parton distribution functions (PDFs) in quantum Chromodynamics (QCD) is revisited again and again in all effort of providing more precise perturbative QCD predictions for the experiments in the Large Hadron Collider (LHC) and other hadron colliders (*e.g.* Tevatron). In this work we intend

* This work is partly supported by the EU grant MTKD-CT-2004-510126 in partnership with the CERN Physics Department and by the Polish Ministry of Scientific Research and Information Technology the grant No 620/E-77/6.PR UE/DIE 188/2005–2008.

to present a methodology of solving QCD evolution equations using Monte Carlo techniques for several types of the evolutions, the resulting numerical results, including the comparisons with other non-MC numerical methods.

Two decades ago, when first attempts of solving numerically and precisely the evolution time dependence of the parton distribution functions (PDFs) according to the DGLAP [1] equations were made, it was unthinkable that the Monte Carlo techniques could be used for this purpose. It was simply because the computers were too slow by several orders of the magnitude. Instead, various faster techniques were developed, based mainly on dividing the evolution time into short periods and using discrete grid in x -space — they are presently still widely used. Nowadays, with much faster computers, it is perfectly feasible to solving numerically the QCD evolution equations with 3–4 digit precision for DGLAP and other types of evolutions, albeit it is still much slower than with other techniques.

One may, therefore, ask the following question: does the MC technique of solving QCD evolution equations have some advantages over other techniques which makes it worth to pursue in spite of its slowness? In our opinion the MC technique offers certain unique advantages. Let us mention the most important ones: Although numerical statistical error is usually bigger than for other methods, this error is very stable and robust, not prone to any effects related to finite grid or time slicing. Another advantage of the MC method is that for many types of partons one may solve the evolution equations for all parton types simultaneously, without the need of diagonalizing kernels, that is using PDFs in the basis of gluon, singlet quark and several types of the non-singlet quark components, and then recombining that back. Finally, the biggest potential advantage is that in the MC method one can devise mapping of the evolution time and other variables into four-momenta, hence to set-up the starting point for constructing a more realistic treatment of the multiparton emission shower, that is the so-called parton shower MC. Also, the extensions from orthodox DGLAP towards more complicated kernels/evolutions featuring small x resummations, such as CCFM [2], can be treated with the MC techniques more easily than with other methods.

It should be stressed that this work is closely related with another work of Ref. [3]. In fact the MC programs of this work are exploited in Ref. [3] to test more complicated MC techniques of solving evolution equations. The main difference between this work and Ref. [3] is that here we concentrate on the Markovian class of MC solutions, while Ref. [3] elaborates on the class of non-Markovian techniques, in which the parton energy fraction x and its type f are constrained (predefined). The Markovian MC is better suited for the final-state parton cascade while the constrained MC of Ref. [3] is better for the initial state cascade, for instance in hadron colliders (W/Z boson production).

Our paper is organized as follows: In Section 2 we present general form of evolution equations and their iterative solutions. In Section 3 we describe in detail three Markovian algorithms for solving these equations. Section 4 contains details on evolution kernels and form-factors. In Section 5 we give some remarks on Monte Carlo implementations of the above algorithms. Section 6 is devoted to the Chebyshev polynomials method of solving the evolution equations. In Section 7 we present our numerical results. Finally, Section 8 summerizes the paper.

2. General evolution equations

In this work we shall cover several types of the QCD evolution equations ranging from DGLAP [1] to CCFM [2] and their extensions. The generic evolution equation covering all types of QCD evolution of our interest reads

$$\partial_t D_f(t, x) = \sum_{f'} \int_x^1 du \mathcal{K}_{ff'}(t, x, u) D_{f'}(t, u). \quad (2.1)$$

The parton distribution function (PDF) is $D_j(t, u)$, with x being the fraction of the hadron momentum¹ carried by the parton and j being the type (flavour) of the parton. The so called evolution time $t = \ln Q$ represents in QCD logarithm of the energy scale $Q = \mu$ determined by hard scattering process probing PDF. The case of the LL DGLAP case [1] is recovered with the following identification

$$\mathcal{K}_{ff'}(t, x, u) = \frac{1}{u} \mathcal{P}_{ff'} \left(t, \frac{x}{u} \right) = \frac{\alpha_S(t)}{2\pi} \frac{2}{u} P_{ff'}^{(0)} \left(t, \frac{x}{u} \right), \quad (2.2)$$

where $P_{ff'}^{(0)}(z)$ is the lowest order DGLAP kernel.

In the compact operator (matrix) notation Eq. (2.1) reads

$$\partial_t \mathbf{D}(t) = \mathbf{K}(t) \mathbf{D}(t). \quad (2.3)$$

Given a known $\mathbf{D}(t_0)$ at the initial time t_0 , the formal solution at any later time $t \geq t_0$ is provided by the time ordered exponential

$$\mathbf{D}(t) = \exp \left(\int_{t_0}^t \mathbf{K}(t') dt' \right)_{\text{T.O.}} \mathbf{D}(t_0) = \mathbf{G}_{\mathbf{K}}(t, t_0) \mathbf{D}(t_0). \quad (2.4)$$

¹ Or, equivalently, the fraction of the eikonal “plus” variable.

The *time-ordered exponential* evolution operator reads²

$$\begin{aligned} \mathbf{G}_{\mathbf{K}}(t, t_0) &= \mathbf{G}(\mathbf{K}; t, t_0) = \exp \left(\int_{t_0}^t \mathbf{K}(t') dt' \right)_{\text{T.O.}} \\ &= \mathbf{I} + \sum_{n=1}^{\infty} \prod_{i=1}^n \int_{t_0}^t dt_i \theta_{t_i > t_{i-1}} \mathbf{K}(t_i), \end{aligned} \quad (2.5)$$

where $(\mathbf{I})_{f_2, f_1}(x_2, x_1) \equiv \delta_{f_2 f_1} \delta_{x_2 = x_1}$ and the multiplication of the operators is defined as follows

$$(\mathbf{K}(t_2) \mathbf{K}(t_1))_{f_2, f_1}(x_2, x_1) = \sum_{f'} \int_{x_2}^{x_1} dx' \mathcal{K}_{f_2 f'}(t_2, x_2, x') \mathcal{K}_{f' f_1}(t_1, x', x_1). \quad (2.6)$$

From now on we adopt the following notation³:

$$\delta_{x=y} = \delta(x-y), \quad \theta_{y < x} = 1 \quad \text{for } y < x \quad \text{and} \quad \theta_{y < x} = 0 \quad \text{for } y \geq x.$$

In the case of the kernel split into two components, $\mathbf{K}(t) = \mathbf{K}^A(t) + \mathbf{K}^B(t)$, the solution of Eq. (2.4) can be reorganized as follows⁴

$$\begin{aligned} \mathbf{D}(t) &= \mathbf{G}_{\mathbf{K}^B}(t, t_0) \mathbf{D}(t_0) + \sum_{n=1}^{\infty} \left[\prod_{i=1}^n \int_{t_{i-1}}^t dt_i \right] \mathbf{G}_{\mathbf{K}^B}(t, t_n) \\ &\quad \times \left[\prod_{i=1}^n \mathbf{K}^A(t_i) \mathbf{G}_{\mathbf{K}^B}(t_i, t_{i-1}) \right] \mathbf{D}(t_0), \\ \mathbf{G}_{\mathbf{K}}(t, t_0) &= \mathbf{G}_{\mathbf{K}^B}(t, t_0) + \sum_{n=1}^{\infty} \left[\prod_{i=1}^n \int_{t_{i-1}}^t dt_i \right] \mathbf{G}_{\mathbf{K}^B}(t, t_n) \\ &\quad \times \left[\prod_{i=1}^n \mathbf{K}^A(t_i) \mathbf{G}_{\mathbf{K}^B}(t_i, t_{i-1}) \right], \end{aligned} \quad (2.7)$$

² Here and in the following we adopt the following conventions $\prod_{i=1}^n A_i \equiv A_n A_{n-1} \dots A_2 A_1$ and $\prod_{i=1}^n \int dt_i \equiv \int dt_n \int dt_{n-1} \dots \int dt_2 \int dt_1$. The inverse ordering will be similarly denoted with $\overleftarrow{\prod}_{i=1}^n$.

³ Similarly, we define $\theta_{z < y < x} = \theta_{z < y} \theta_{y < x}$.

⁴ The scope of the index i in \prod_i ceases at the closing bracket, but validity scope of indexed variables, like t_i , extends until the formula's end. The use of Eq. (2.6) is understood accordingly.

where $\mathbf{G}_{\mathbf{K}^B}$ is the evolution operator of Eq. (2.5) of the evolution with the kernel \mathbf{K}^B . Formal proof of identities in Eq. (2.7) can be found in Ref. [4].

2.1. Resuming virtual corrections

Monte Carlo method cannot efficiently deal with the non-positive distributions, hence resummation of negative virtual part in the evolution kernel is a necessary preparatory step. It will be done with help of identity of Eq. (2.7). We are going to resum (negative) diagonal virtual part $\mathbf{K}^V = \mathbf{K}^B$ in the kernel

$$\begin{aligned}\mathcal{K}_{ff'}(t, x, u) &= \mathcal{K}_{ff'}^V(t, x, u) + \mathcal{K}_{ff'}^R(t, x, u), \\ \mathcal{K}_{ff'}^V(t, x, u) &= -\delta_{ff'}\delta_{x=u}\mathcal{K}_{ff}^v(t, x).\end{aligned}\quad (2.8)$$

At this point we do not need to be very specific about $\mathcal{K}_{ff'}^R(t, x, u)$ — we only remark that due to infrared (IR) singularity at $x = u$ and $f = f'$ it includes IR cut-off, typically $u - x > \Delta(x, u, t)$, causing \mathcal{K}^v to be also Δ -dependent.

Thanks to diagonality of the kernel \mathbf{K}^V , the corresponding time-ordered exponential is easily calculable

$$\begin{aligned}\{\mathbf{G}_{\mathbf{K}^V}(t, t')\}_{ff'}(x, u) &= \delta_{ff'}\delta_{x=u} e^{-\Phi_f(t, t'|x)}, \\ \Phi_f(t, t'|x) &= \int_{t'}^t dt'' \mathcal{K}_{ff}^v(t'', x).\end{aligned}\quad (2.9)$$

Inserting the above in Eq. (2.7) we obtain

$$\mathbf{D}(t) = \sum_{n=0}^{\infty} \left[\prod_{i=1}^n \int_{t_{i-1}}^t dt_i \right] \mathbf{G}_{\mathbf{K}^V}(t, t_n) \left[\prod_{i=1}^n \mathbf{K}^R(t_i) \mathbf{G}_{\mathbf{K}^V}(t_i, t_{i-1}) \right] \mathbf{D}(t_0). \quad (2.10)$$

More compact notation is obtained with the prescription $\prod_{i=k}^{k-1} \mathbf{A}_i \equiv \mathbf{I}$ and $\prod_{i=k}^{k-1} \int dt_i \equiv 1$.

2.2. Momentum sum rule

Evolution equations and their time ordered solutions do not require any assumptions about the normalization of PDFs and kernels. However, Markovian Monte Carlo methods are inherently based on the unitary normalization of the probability distributions (for the forward step). Hence, we concentrate on the evolution equations which are supplemented with some conservation

rule, providing time-independent normalization condition. For DGLAP it is the momentum sum rule which is obeyed exactly and is exploited to this end (it can also be used for the CCFM class models). It will be also formulated in terms of the compact operator formalism. Let us define operator (vector) $\bar{\mathbf{E}}$ acting from the left side

$$\bar{\mathbf{E}} \mathbf{D}(t) \equiv \int_0^1 dx \sum_f x D_f(t, x). \quad (2.11)$$

The momentum sum rule can be stated as the following time conservation law:

$$\partial_t \bar{\mathbf{E}} \mathbf{D}(t) = 0. \quad (2.12)$$

Inserting evolution equation one obtains immediately

$$\partial_t \bar{\mathbf{E}} \mathbf{D}(t) = \bar{\mathbf{E}} \mathbf{K} \mathbf{D}(t) = 0. \quad (2.13)$$

The sufficient condition for the above to be true is the following property of the kernel

$$\bar{\mathbf{E}} \mathbf{K} = \bar{\mathbf{0}}, \quad (\bar{\mathbf{E}} \mathbf{K})_f(u) = \sum_{f'} \int_0^1 dx x \mathcal{K}_{f'f}(t, x, u) = 0, \quad (2.14)$$

for any u and f . In particular we have $\bar{\mathbf{E}} \mathbf{K}^V + \bar{\mathbf{E}} \mathbf{K}^R = \bar{\mathbf{0}}$, from which we can derive immediately the virtual part of the kernel

$$- (\bar{\mathbf{E}} \mathbf{K}^V)_f(u) = u \mathcal{K}_{ff}^v(t, u) = \sum_{f'} \int_0^u dx x \mathcal{K}_{f'f}^R(t, x, u) = (\bar{\mathbf{E}} \mathbf{K}^R)_f(u). \quad (2.15)$$

From $\bar{\mathbf{E}} \mathbf{K} = \bar{\mathbf{0}}$ also follows the following useful identity

$$\bar{\mathbf{E}} \mathbf{G}_{\mathbf{K}}(t, t_0) = \bar{\mathbf{E}}, \quad (2.16)$$

which provides immediately $\bar{\mathbf{E}} \mathbf{D}(t) = \bar{\mathbf{E}} \mathbf{D}(t_0)$.

2.3. Markovianization

The aim is now to transform Eq. (2.10) into a form better suited for the Monte Carlo evaluation, using Markovian algorithm. The basic problem is to show how to change the integration order from $\int_{t_0}^t dt_n \dots \int_{t_0}^{t_3} dt_2 \int_{t_0}^{t_2} dt_1$ to $\int_{t_0}^t dt_1 \int_{t_0}^{t_1} dt_2 \dots \int_{t_0}^{t_{n-1}} dt_n$, taking into account non-commutative character of the product of the kernels in the time ordered exponentials.

It is convenient not only to change the order of the t -integration but also to transpose simultaneously (temporarily) both sides of Eq. (2.10)

$$\bar{D}(t) = \bar{D}(t_0) \sum_{n=0}^{\infty} \left[\overline{\prod}_{i=1}^n \int_{t_0}^t dt_i \theta_{t_i > t_{i-1}} \bar{\mathbf{G}}_{\mathbf{K}^V}(t_i, t_{i-1}) \bar{\mathbf{K}}^R(t_i) \right] \bar{\mathbf{G}}_{\mathbf{K}^V}(t, t_n). \quad (2.17)$$

In the next step we isolate the integration over t_1 , the outermost one,

$$\begin{aligned} \bar{D}(t) = \bar{D}(t_0) & \left\{ \bar{\mathbf{G}}_{\mathbf{K}^V}(t, t_0) + \int_{t_0}^t dt_1 \bar{\mathbf{G}}_{\mathbf{K}^V}(t_1, t_0) \bar{\mathbf{K}}^R(t_1) \right. \\ & \left. \times \sum_{n=1}^{\infty} \left[\overline{\prod}_{i=2}^n \int_{t_1}^t dt_i \theta_{t_i > t_{i-1}} \bar{\mathbf{G}}_{\mathbf{K}^V}(t_i, t_{i-1}) \bar{\mathbf{K}}^R(t_i) \right] \bar{\mathbf{G}}_{\mathbf{K}^V}(t, t_n) \right\}. \end{aligned} \quad (2.18)$$

Closer look into second line in the above equation reveals⁵ that it represents again the time ordered evolution operator $\bar{\mathbf{G}}_{\mathbf{K}}(t, t_1)$ (with $t_0 \rightarrow t_1$). We obtain therefore

$$\bar{D}(t) = \bar{D}(t_0) \left\{ \bar{\mathbf{G}}_{\mathbf{K}^V}(t, t_0) + \int_{t_0}^t dt_1 \bar{\mathbf{G}}_{\mathbf{K}^V}(t_1, t_0) \bar{\mathbf{K}}^R(t_1) \bar{\mathbf{G}}_{\mathbf{K}}(t, t_1) \right\}. \quad (2.19)$$

Transposition can be now removed and the integral over t_1 is pulled out

$$D(t) = \int_{t_0}^t dt_1 \left\{ \mathbf{G}_{\mathbf{K}}(t, t_1) \mathbf{K}^R(t_1) \mathbf{G}_{\mathbf{K}^V}(t_1, t_0) + \mathbf{G}_{\mathbf{K}^V}(t, t_0) \delta_{t_1=t} \right\} D(t_0) \quad (2.20)$$

The above result can be also presented as an integral equation for the evolution operator

$$\mathbf{G}_{\mathbf{K}}(t, t_0) = \int_{t_0}^t dt_1 \left\{ \mathbf{G}_{\mathbf{K}}(t, t_1) \mathbf{K}^R(t_1) \mathbf{G}_{\mathbf{K}^V}(t_1, t_0) + \mathbf{G}_{\mathbf{K}^V}(t, t_0) \delta_{t_1=t} \right\}. \quad (2.21)$$

⁵ After renaming $t_i \rightarrow t_{i-1}$ and shifting indices i and n by one.

This can be inserted back into Eq. (2.19) many times. The following example shows three levels of the nesting

$$\begin{aligned}
 \mathbf{D}(t) = \int_{t_0}^t dt_1 \left(\int_{t_1}^t dt_2 \left[\int_{t_2}^t dt_3 \left\{ \mathbf{G}_{\mathbf{K}}(t, t_3) \mathbf{K}^{\mathbf{R}}(t_3) \mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t_3, t_2) + \mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t, t_2) \delta_{t_3=t} \right\} \right. \right. \\
 \left. \left. \times \mathbf{K}^{\mathbf{R}}(t_2) \mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t_2, t_1) + \mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t, t_1) \delta_{t_2=t} \right] \right. \\
 \left. \times \mathbf{K}^{\mathbf{R}}(t_1) \mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t_1, t_0) + \mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t, t_0) \delta_{t_1=t} \right) \mathbf{D}(t_0). \tag{2.22}
 \end{aligned}$$

It should be stressed that integration over t_1 is now the external one and in the MC it will be generated as a first one.

If the above nesting is continued to the level $N + 1$, then one may argue that the contribution from the term with $\mathbf{G}_{\mathbf{K}}(t, t_{N+1})$ for large N decreases like $1/N!$, hence in the Markovian MC we may use the following formula “truncated” at large fixed N playing a role of a dummy technical parameter:

$$\begin{aligned}
 \mathbf{D}(t) = \int_{t_0}^t dt_1 \left(\int_{t_1}^t dt_2 \left[\int_{t_2}^t dt_3 \left\{ \dots \right. \right. \right. \\
 \dots \int_{t_{N-1}}^t dt_N \left\{ \mathbf{K}^{\mathbf{R}}(t_N) \mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t_N, t_{N-1}) + \mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t, t_{N-1}) \delta_{t_N=t} \right\} \\
 \vdots \\
 \left. \times \mathbf{K}^{\mathbf{R}}(t_2) \mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t_2, t_1) + \mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t, t_1) \delta_{t_2=t} \right] \\
 \left. \times \mathbf{K}^{\mathbf{R}}(t_1) \mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t_1, t_0) + \mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t, t_0) \delta_{t_1=t} \right) \mathbf{D}(t_0), \tag{2.23}
 \end{aligned}$$

where the integration over t_{N+1} was consumed by $\delta_{t_{N+1}=t}$. The above identity will be instrumental in constructing MMC algorithm in the following section.

3. Markovian MC algorithms

For the Monte Carlo method one needs a (sum of) scalar multi-dimensional integral. For the straightforward Markovian algorithm we shall take the following multi-integral

$$C = \bar{\mathbf{E}}\mathbf{D}(t) = \bar{\mathbf{E}}\mathbf{G}_{\mathbf{K}}(t, t_0)\mathbf{D}(t_0). \tag{3.1}$$

The aim is to generate with the MC method all internal integration variables in the above equation. Then, the histogram of the variable $x = x_n$ and flavour type $f = f_n$ is evaluated in the high statistic MC run. Such a histogram is defined by means of inserting Dirac delta functions in the above multi-integral:

$$D_f(x) = \sum_{n=0}^{\infty} \sum_{f_n, f_0} \int dx_n dx_0 \left(\mathbf{G}_{\mathbf{K}}(t, t_0) \right)_{f_n, f_0}^{(n)}(x_n, x_0) \delta_{x=x_n} \delta_{f=f_n} D_{f_0}(t_0, x_0), \quad (3.2)$$

where n is the dimensionality of the integral in $\mathbf{G}_{\mathbf{K}}$.

3.1. Basic formalism

As a warm-up exercise let us insert $\mathbf{D}(t)$ of Eq. (2.20) into $\bar{\mathbf{E}}\mathbf{D}(t)$ and check how the identity $\bar{\mathbf{E}}\mathbf{D}(t) = \bar{\mathbf{E}}\mathbf{D}(t_0)$ is recovered through explicit integration over t_1

$$\begin{aligned} \bar{\mathbf{E}}\mathbf{D}(t) &= \int_{t_0}^t dt_1 \left\{ \bar{\mathbf{E}}\mathbf{G}_{\mathbf{K}}(t, t_1) \mathbf{K}^{\mathbf{R}}(t_1) \mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t_1, t_0) + \bar{\mathbf{E}}\mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t, t_0) \delta_{t_1=t} \right\} \mathbf{D}(t_0) \\ &= \int_{t_0}^t dt_1 \left\{ -\bar{\mathbf{E}}\mathbf{K}^{\mathbf{V}}(t_1) \mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t_1, t_0) + \bar{\mathbf{E}}\mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t, t_0) \delta_{t_1=t} \right\} \mathbf{D}(t_0) \\ &= \int_{t_0}^t dt_1 \left\{ -\bar{\mathbf{E}} \partial_{t_1} \mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t_1, t_0) + \bar{\mathbf{E}}\mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t, t_0) \delta_{t_1=t} \right\} \mathbf{D}(t_0) \\ &= \left\{ -\bar{\mathbf{E}}\mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t_1, t_0) \Big|_{t_1=t_0}^{t_1=t} + \bar{\mathbf{E}}\mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t, t_0) \right\} \mathbf{D}(t_0) \\ &= \bar{\mathbf{E}}\mathbf{D}(t_0). \end{aligned} \quad (3.3)$$

In the above the most essential was the use of $\bar{\mathbf{E}}\mathbf{G}_{\mathbf{K}}(t, t_1) = \bar{\mathbf{E}}$ in the first step, because it has allowed to decouple t_1 -integration from the integrations inside $\mathbf{G}_{\mathbf{K}}(t, t_1)$. Next, $\bar{\mathbf{E}}\mathbf{K}^{\mathbf{R}}(t_1) = -\bar{\mathbf{E}}\mathbf{K}^{\mathbf{V}}(t_1)$ was employed, then the evolution equation for $\mathbf{G}_{\mathbf{K}^{\mathbf{V}}}$ and finally $\mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t_0, t_0) = \mathbf{I}$ was also used. The decoupled inner integrations are explicitly present in the following iterative formula

$$\begin{aligned}
\bar{\mathbf{E}}\mathbf{D}(t) = & \int_{t_0}^t dt_1 \left(\int_{t_1}^t dt_2 \left[\int_{t_2}^t dt_3 \left\{ \dots \right. \right. \right. \\
& \dots \int_{t_{N-1}}^t dt_N \left\{ \bar{\mathbf{E}}\mathbf{K}^{\mathbf{R}}(t_N)\mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t_N, t_{N-1}) + \bar{\mathbf{E}}\mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t, t_{N-1})\delta_{t_N=t} \right\} \\
& \vdots \\
& \left. \times \mathbf{K}^{\mathbf{R}}(t_2)\mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t_2, t_1) + \bar{\mathbf{E}}\mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t, t_1)\delta_{t_2=t} \right] \\
& \left. \times \mathbf{K}^{\mathbf{R}}(t_1)\mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t_1, t_0) + \bar{\mathbf{E}}\mathbf{G}_{\mathbf{K}^{\mathbf{V}}}(t, t_0)\delta_{t_1=t} \right) \mathbf{D}(t_0). \quad (3.4)
\end{aligned}$$

Again, we would like to stress that the order of the integration starting from t_1 and ending with t_N is exactly the one which will be realized in the Markovian Monte Carlo algorithm.

3.2. Straightforward Markovian algorithm

In the Markovian MC we are going to generate t_i , one after another, starting from t_1 until for certain n , $0 \leq n \leq N$, $t = t_{n+1}$ is reached⁶. For this to be feasible in the Markovian MC, we have to show with the same algebra as in Eq. (3.4), that all integrals over t_i are properly normalized to momentum fraction⁷ x_{i-1} , starting with the innermost $\int_{t_{N-1}}^t dt_N$ and finishing with outermost $\int_{t_0}^t dt_1$. Following the above warm-up example one can show that the integration over t_1 decouples completely from all inner integrations over t_2, \dots, t_N and, therefore, can be generated independently as a first variable in the MC algorithm.

In the MC generation, whenever $\delta_{t_{n+1}=t}$ term is encountered for the first time, the real parton emission chain is terminated. More precisely, for all $k > n$ one may formally define $t_k = t$, but they are dummy (not used).

In Ref. [5] it was stated, that every standard (classic) MC algorithm can be reduced to a superposition of only three elementary methods: mapping of variables, weighting-rejecting and branching. As seen in Fig. 1, where the above basic MMC algorithm is depicted using graphical notation of Ref. [5], it is indeed a superposition of branching and mapping — every box $\boxed{f_i, x_i}$ typically includes more elementary methods (typically mappings and branchings).

⁶ Maximum number of steps N is large and fixed. Formally, $N \rightarrow \infty$ is understood.

⁷ Unitary normalization is obtained by means of applying $1/x_{i-1}$ normalization factor.

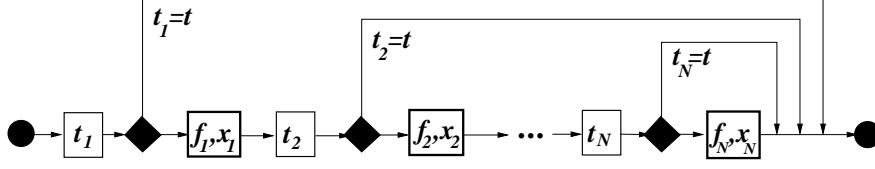


Fig. 1. Scheme of the standard Markovian Monte Carlo.

What still remains is to define in a more detail the distribution of all three variables of t_i, f_i, x_i of the single Markovian step after generating $t_{i-1}, f_{i-1}, x_{i-1}$ in the preceding step:

$$\begin{aligned}
 1 &= \frac{1}{x_{i-1}} \int_{t_{i-1}}^t dt_i \left\{ \bar{\mathbf{E}} \mathbf{K}^{\text{R}}(t_i) \mathbf{G}_{\mathbf{K}^{\text{V}}}(t_i, t_{i-1}) + \bar{\mathbf{E}} \mathbf{G}_{\mathbf{K}^{\text{V}}}(t, t_{i-1}) \delta_{t_i=t} \right\}_{f_{i-1}}(x_{i-1}) \\
 &= \frac{1}{x_{i-1}} \int_{t_{i-1}}^t dt_i \left\{ \left[\sum_{f_i} \int dx_i x_i \mathcal{K}_{f_i f_{i-1}}^{\text{R}}(t_i, x_i, x_{i-1}) e^{-\Phi_{f_{i-1}}(t_i, t_{i-1})} \right] \right. \\
 &\quad \left. + x_{i-1} \delta_{t_i=t} e^{-\Phi_{f_{i-1}}(t, t_{i-1})} \right\} \\
 &= \int_{t_{i-1}}^t dt_i \sum_{f_i} \int dx_i \omega(t_i, f_i, x_i | t_{i-1}, f_{i-1}, x_{i-1}). \tag{3.5}
 \end{aligned}$$

Let us also show the above distribution in a form immediately suitable for the MC generation

$$\begin{aligned}
 1 &= e^{-\Phi_{f_{i-1}}(t, t_{i-1})} + \int_{e^{-\Phi_{f_{i-1}}(t, t_{i-1})}}^1 d \left(e^{-\Phi_{f_{i-1}}(t_i, t_{i-1})} \right) \\
 &\quad \times \left[\sum_{f_i} \frac{\partial_{t_i} \Phi_{f_i f_{i-1}}(t_i, t_{i-1} | x_{i-1})}{\partial_{t_i} \Phi_{f_{i-1}}(t_i, t_{i-1} | x_{i-1})} \int dx_i \frac{1}{\partial_{t_i} \Phi_{f_i f_{i-1}}(t_i, t_{i-1} | x_{i-1})} \right. \\
 &\quad \left. \times \frac{x_i}{x_{i-1}} \mathcal{K}_{f_i f_{i-1}}^{\text{R}}(t_i, x_i, x_{i-1}) \right], \tag{3.6}
 \end{aligned}$$

where virtual form-factor is evaluated using real emission kernels and split into contributions from various transition channels according to

$$\begin{aligned}
\Phi_f(t_1, t_0|u) &= \int_{t_0}^{t_1} dt \mathcal{K}_{ff}^v(t, u) = \sum_{f'} \int_{t_0}^{t_1} dt \int_0^u \frac{dx}{u} x \mathcal{K}_{f'f}^R(t, x, u) \\
&= \sum_{f'} \Phi_{f'f}(t_1, t_0|u).
\end{aligned} \tag{3.7}$$

Given an uniform random number $r \in (0, 1)$, generation of t_i is done by means of solving the equation $r = U(t_i) = e^{-\Phi_{f_{i-1}}(t_i, t_{i-1})}$ for t_i , within the range $r \in [e^{-\Phi_{f_{i-1}}(t_i, t_{i-1})}, 1]$. The remaining range $r \in [0, e^{-\Phi_{f_{i-1}}(t_i, t_{i-1})}]$ is mapped into a single point $t_i = t$, that is the point where the distribution proportional to $\delta_{t=t_i}$ resides. Flavour index f_i is generated according to normalized discrete probability distribution $P_{f_i} = \partial_{t_i} \Phi_{f_i f_{i-1}}(t_i, t_{i-1}) / \partial_{t_i} \Phi_{f_i f_{i-1}}(t_i, t_{i-1})$. Finally, variable x_i is generated according to the normalized integrand of $\int dx_i$ in Eq. (3.6).

The above Markovian MC algorithm of Fig. 1 is completely standard and very well known. Practical problem is that the generation of t_i , for more complicated kernels than in DGLAP case requires numerical evaluation and inversion of the form-factor $\Phi_{f_i f_{i-1}}(t_i, t_{i-1})$. Generation of f_i is always rather trivial. On the other hand, generation of x_i can be also nontrivial. The above problems can be solved, at least partly, by more sophisticated versions of the Markovian MC, generally using MC weights, see next section.

3.3. Weighted Markovian MC algorithms

In the simplest Markovian MC method with weighted events, which will be referred to as an *internal loop MMC*, the real emission kernel in the distribution used in the generation of x_i is replaced by the simplified one $\mathcal{K}_{f_i f_{i-1}}^R(t_i, x_i, x_{i-1}) \rightarrow \bar{\mathcal{K}}_{f_i f_{i-1}}^R(t_i, x_i, x_{i-1})$, such that $\mathcal{K}^R \leq \bar{\mathcal{K}}^R$. Variables x_i are generated according to normalized distribution

$$\bar{P}(x_i) = \frac{1}{\partial_{t_i} \bar{\Phi}_{f_i f_{i-1}}(t_i, t_{i-1}|x_{i-1})} \frac{x_i}{x_{i-1}} \bar{\mathcal{K}}_{f_i f_{i-1}}^R(t_i, x_i, x_{i-1}), \tag{3.8}$$

where

$$\bar{\Phi}_{f'f}(t_1, t_0|u) = \int_{t_0}^{t_1} dt \int_0^u \frac{dx}{u} x \bar{\mathcal{K}}_{f'f}^R(t, x, u) \tag{3.9}$$

is also simpler than Φ . The above simplification is corrected by the MC weight

$$w_i^z = \frac{\mathcal{K}_{f_i f_{i-1}}^R(t_i, x_i, x_{i-1})}{\bar{\mathcal{K}}_{f_i f_{i-1}}^R(t_i, x_i, x_{i-1})} \leq 1, \tag{3.10}$$

which is used in the local rejection loop, for every forward step separately, using uniform random number r : if $r > w_i^z$ then generation of x_i is repeated. In this method generation of t_i is still done using the exact Sudakov form-factor $\bar{\Phi}_{f_{i-1}}(t_i, t_{i-1}|x_{i-1})$. This type of MMC algorithm is shown schematically in Fig. 2 and it is essentially a particular realization of the basic algorithm of Fig. 1. In the second method, which will be referred to as a *global loop MMC* the approximate form-factor $\bar{\Phi}_{f_{i-1}}(t_i, t_{i-1}|x_{i-1}) = \sum_{f_i} \bar{\Phi}_{f_i f_{i-1}}(t_i, t_{i-1}|x_{i-1})$ is used for generation of both t_i , f_i and x_i . Global correcting weight w is applied at the very end of the Markovian chain. However, the weight is not just $\prod w_i^z$, but it can be deduced as follows. According to Eq. (3.6) the normalized probability of the forward step $(i-1) \rightarrow i$ reads

$$\begin{aligned} \frac{dP_{f_i}}{dx_i dt_i}(t_{i-1}, f_{i-1}, x_{i-1}) &= \omega_{(i-1) \rightarrow i} = \omega_{(i-1) \rightarrow i}^R + \omega_{(i-1) \rightarrow i}^\delta \\ &= \theta_{t_{i-1} \leq t_i < t} \frac{x_i}{x_{i-1}} \mathcal{K}_{f_i f_{i-1}}^R(t_i, x_i, x_{i-1}) e^{-\bar{\Phi}_{f_{i-1}}(t_i, t_{i-1})} \\ &\quad + \delta_{t_i=t} \delta_{f_i f_{i-1}} \delta_{x_i=x_{i-1}} e^{-\bar{\Phi}_{f_{i-1}}(t, t_{i-1})}. \end{aligned} \quad (3.11)$$

The desired distribution of all variables in MMC event with n emission is

$$\omega^{(n)} = \omega_{n \rightarrow n+1}^\delta \prod_{i=1}^n \omega_{(i-1) \rightarrow i}^R. \quad (3.12)$$

However, in the actual global loop MMC method the distribution of these variables (before applying correcting MC weight) is the following

$$\bar{\omega}^{(n)} = \bar{\omega}_{n \rightarrow n+1}^\delta \prod_{i=1}^n \bar{\omega}_{(i-1) \rightarrow i}^R, \quad (3.13)$$

where barring means substitution of exact kernels and form-factors with the approximate ones: $\mathcal{K} \rightarrow \bar{\mathcal{K}}$, $\Phi \rightarrow \bar{\Phi}$. Global correcting MC weight is, therefore, just the usual ratio of the exact and approximate distributions

$$w^{(n)} = \frac{\omega^{(n)}}{\bar{\omega}^{(n)}} = e^{\bar{\Phi}_{f_n}(t, t_n) - \Phi_{f_n}(t, t_n)} \left(\prod_{i=1}^n w_i^z e^{\bar{\Phi}_{f_{i-1}}(t_i, t_{i-1}) - \Phi_{f_{i-1}}(t_i, t_{i-1})} \right). \quad (3.14)$$

The above weight is tested against the random number after the entire MC event generation is completed, see the external return loop in Fig. 3. Note that, although approximate form-factor $\bar{\Phi}_{f_{i-1}}(t_i, t_{i-1})$ and its inverse is used here for generation of t_i , the exact form-factor is still needed to calculate the global weight⁸.

⁸ Note that in our older papers describing this method we were denoting $T_f = \bar{\Phi}_f$ and $\Delta_f = \bar{\Phi}_f - \Phi_f$.

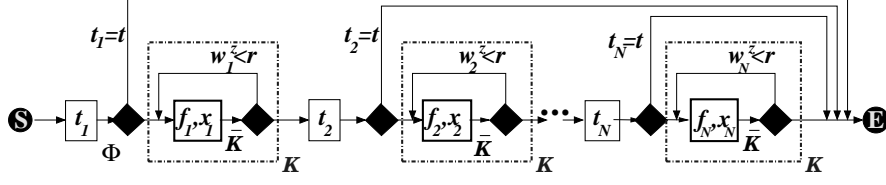


Fig. 2. Scheme of Markovian Monte Carlo with the internal rejection loop.

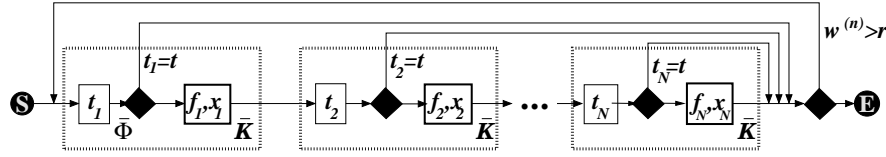


Fig. 3. Scheme of Markovian Monte Carlo with the global rejection loop.

Finally, we are going to derive the third method which will be referred to as *MMC with pseudo-emissions*. This method is also known in the literature under the name of the *Markovian MC algorithm with veto* or shortly *veto algorithm*. In this case we do the following modification of the evolution kernel

$$\begin{aligned}\tilde{\mathcal{K}}_{ff'}^V(t, x, u) &= \mathcal{K}_{ff'}^V(t, x, u) - \delta_{ff'} \delta_{x=u} \mathcal{K}_{ff}^S(t, x), \\ \tilde{\mathcal{K}}_{ff'}^R(t, x, u) &= \mathcal{K}_{ff'}^R(t, x, u) + \delta_{ff'} \delta_{x=u} \mathcal{K}_{ff}^S(t, x),\end{aligned}\quad (3.15)$$

where $\mathcal{K}_{ff}^S(t, x)$ is positive and its magnitude is judiciously chosen as the integral difference of the exact kernel \mathcal{K}^R and the approximate kernel $\bar{\mathcal{K}}^R \geq \mathcal{K}^R$ (typically the same as in the previous methods)

$$\mathcal{K}_{ff}^S(t, u) = \sum_{f'} \int_0^1 dx \frac{x}{u} (\bar{\mathcal{K}}_{ff'}^R(t, x, u) - \mathcal{K}_{ff'}^R(t, x, u)). \quad (3.16)$$

In this way we are artificially adding to the real emission kernel finite positive contributions, which represents real emission of a gluon with exactly zero momentum! This extra real emission is compensated immediately and exactly by enlarging negative virtual correction. Since the total evolution kernel remains unchanged,

$$\mathcal{K}_{ff'}(t, x, u) = \tilde{\mathcal{K}}_{ff'}^V(t, x, u) + \tilde{\mathcal{K}}_{ff'}^R(t, x, u), \quad (3.17)$$

the same time-ordered exponential solution remains valid, $\mathbf{D}(t) = \mathbf{G}_K(t, t_0) \mathbf{D}(t_0)$. However, the difference will occur when resumming virtual negative

corrections, because we are now resumming the enlarged $\tilde{\mathcal{K}}^V$. The basic solution used as a starting point for MMC now reads

$$\begin{aligned}
D(t) &= \sum_{n=0}^{\infty} \left[\prod_{i=1}^n \int_{t_{i-1}}^t dt_i \right] \mathbf{G}_{\tilde{\mathcal{K}}^V}(t, t_n) \left[\prod_{i=1}^n \tilde{\mathcal{K}}^R(t_i) \mathbf{G}_{\tilde{\mathcal{K}}^V}(t_i, t_{i-1}) \right] D(t_0), \\
\{\mathbf{G}_{\tilde{\mathcal{K}}^V}(t, t')\}_{ff'}(x, u) &= \delta_{ff'} \delta_{x=u} e^{-\tilde{\Phi}_f(t, t'|x)}, \\
\tilde{\Phi}_f(t, t'|x) &= \int_{t'}^t dt'' \tilde{\mathcal{K}}_{ff'}^v(t'', x). \tag{3.18}
\end{aligned}$$

The momentum sum rule still holds and can be used to evaluate modified form-factor

$$\begin{aligned}
\tilde{\Phi}_f(t_1, t_0|u) &= \int_{t_0}^{t_1} dt \tilde{\mathcal{K}}_{ff}^v(t, u) = \int_{t_0}^{t_1} dt \sum_{f'} \int_0^u \frac{dx}{u} x \tilde{\mathcal{K}}_{f'f}^R(t, x, u) \\
&= \int_{t_0}^{t_1} dt \left(\sum_{f'} \int_0^u \frac{dx}{u} x \mathcal{K}_{f'f}^R(t, x, u) + \mathcal{K}_{ff}^S(t, u) \right) \\
&= \int_{t_0}^{t_1} dt \sum_{f'} \int_0^u \frac{dx}{u} x \tilde{\mathcal{K}}_{f'f}^R(t, x, u) = \bar{\Phi}_f(t_1, t_0|u). \tag{3.19}
\end{aligned}$$

Obviously, \mathcal{K}^S was adjusted such that $\tilde{\Phi}_f = \bar{\Phi}_f$ holds. The immediate important gain is that simplified form-factor $\bar{\Phi}_f$ is used to generate t_i , instead of more complicated Φ_f .

However, there is one more possible gain from $\tilde{\Phi}_f = \bar{\Phi}_f$ in the algorithm of generating f_i and x_i . Due to $\mathcal{K} \rightarrow \tilde{\mathcal{K}}$, the probability of choosing f_i should be

$$\tilde{P}_{f_i} = \frac{\partial_{t_i} \tilde{\Phi}_{f_i f_{i-1}}(t_i, t_{i-1}|x_{i-1})}{\partial_{t_i} \tilde{\Phi}_{f_{i-1}}(t_i, t_{i-1}|x_{i-1})} = \frac{\partial_{t_i} \tilde{\Phi}_{f_i f_{i-1}}(t_i, t_{i-1}|x_{i-1})}{\partial_{t_i} \bar{\Phi}_{f_{i-1}}(t_i, t_{i-1}|x_{i-1})} \tag{3.20}$$

The next x_i should be generated according to $\tilde{\mathcal{K}}_{f_i f_{i-1}}^R(t_i, x_i, x_{i-1})$, including singular part proportional to $\delta_{x_i=x_{i-1}} \delta_{ff'}$. However, generating x_i and f_i according to this distribution can be inconvenient and the following clever trick may be helpful. Let us consider for a moment the internal loop MMC algorithm with $\bar{P}_{f_i} = \partial_{t_i} \bar{\Phi}_{f_i f_{i-1}} / \partial_{t_i} \bar{\Phi}_{f_{i-1}}$ for which x_i is generated according to $\bar{\mathcal{K}}(x_i, \dots)$. Give uniform random number r , the fraction of MC events obeying $r > w_i^z$ will be $(\partial_{t_i} \bar{\Phi}_{f_{i-1}} - \partial_{t_i} \Phi_{f_{i-1}}) / \partial_{t_i} \bar{\Phi}_{f_{i-1}}$. Now, due to

$\partial_{t_i} \tilde{\Phi}_f = \partial_{t_i} \bar{\Phi}_f$ this fraction happens to be exactly the same as the fraction of events $(\partial_{t_i} \tilde{\Phi}_{f_{i-1}} - \partial_{t_i} \bar{\Phi}_{f_{i-1}}) / \partial_{t_i} \bar{\Phi}_{f_{i-1}}$ located in the $\delta_{x_i=x_{i-1}} \delta_{f f'}$ term!

One can therefore proceed almost exactly as in the internal loop MMC algorithm, that is generate f_i according to \bar{P}_i and x_i according to kernel $\bar{\mathcal{K}}$, and next, for events with $r > w_i^z$, instead of repeating generation of f_i and x_i for the same t_i , one sets $f_i = f_{i-1}$ and $x_i = x_{i-1}$ (zero momentum real gluon!) and proceeds to generation of the next t_{i+1} . This completes description and derivation of the algorithm of MMC with *pseudo-emissions*. The advantage of this algorithm is that the numerical evaluation and inversion of the possibly complicated exact form-factor $\Phi_{ff'}(t, t'|u)$ is not required — only the simplified version $\bar{\Phi}_{ff'}(t, t'|u)$ is used. This type of MMC algorithm with pseudo-emissions is shown schematically in Fig. 4.

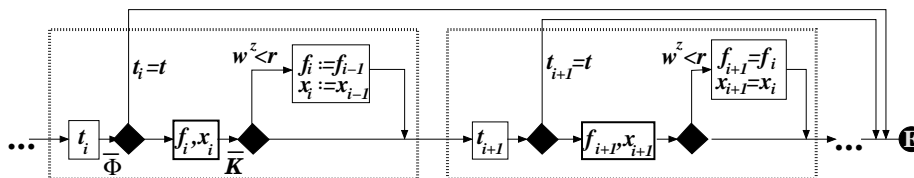


Fig. 4. Scheme of Markovian Monte Carlo with pseudo-emissions (veto).

Comparing to other derivations of the veto MMC, in our derivation we reduce veto MMC to the standard MMC without the need of repetition of the the explicit resummation of the contributions form \bar{G} s (which is typically done in the derivations of veto MMC in the literature). We believe that the proof presented here is both simpler and more rigorous.

Finally let us comment on one purely technical point. One may get false impression that the above algorithm with pseudo-emissions visualized in Fig. 4 cannot be reduced to a superposition of the three elementary methods of Ref. [5]. In fact it can be done rather easily — the above algorithm is just a variant of the basic algorithm of Fig. 1, in which the branch with $W^z < r$ representing emission of another type of real gluon \bar{G} with exactly zero momentum is present.

4. Kernels and form-factors

Our main interest is in the CCFM-like evolution with the evolution time being rapidity and running coupling constant α_S dependent on the transverse momentum of the emitted gluon. The LL DGLAP will be shown as a reference case, while another with rapidity ordering and z -dependent α_S will be also discussed. as a useful intermediate case between CCFM and

DGLAP. Running coupling constant

$$\alpha_S(q) = \alpha_S^{(0)}(q) = \frac{2\pi}{\beta_0} \frac{1}{\ln q - \ln A_0} \quad (4.1)$$

is taken in the LL approximation. All three types of evolution in this work are essentially the same as in Ref. [3], so we shall reduce to a minimum presentation of the corresponding three kernels and form-factors.

4.1. Kinematics

As already stressed we define explicit mapping of the evolution variables to four-momenta, because of possible applications in the parton shower MCs. It will be the same as in Ref. [3] and is basically that of CCFM model [2]. We define k_i^μ to be the momenta of emitted partons, whereas q_i^μ denote the virtual partons along the emission tree. The initial hadron carries $q_h^+ = 2E_h$. For each emitted parton we define

$$\begin{aligned} k_i^+ &= q_{i-1}^+ - q_i^+ = 2E_h(x_{i-1} - x_i) = 2E_h x_{i-1}(1 - z_i), \\ \eta_i &= \frac{1}{2} \ln \frac{k_i^+}{k_i^-}. \end{aligned} \quad (4.2)$$

Consequently, the transverse momentum of emitted massless parton reads

$$k_i^\perp = \sqrt{k_i^+ k_i^-} = k_i^+ e^{-\eta_i} = x_{i-1}(1 - z_i) 2E_h e^{-\eta_i}. \quad (4.3)$$

This suggests the convenient definition of the rapidity-based evolution time as

$$t_i = -\eta_i + \ln(2E_h). \quad (4.4)$$

Now, the transverse momentum of the emitted parton (in units of 1 GeV) becomes:

$$k_i^\perp = e^{t_i} x_i \frac{(1 - z_i)}{z_i} = e^{t_i} x_{i-1}(1 - z_i) = e^{t_i}(x_{i-1} - x_i). \quad (4.5)$$

4.2. Three types of kernels

In the following we are going to define matrix elements of the kernels

$$(\mathbf{K})_{ff'}(x, u) = \mathcal{K}_{ff'}(t, x, u) = \mathcal{K}_{ff'}^V(t, x, u) + \mathcal{K}_{ff'}^R(t, x, u), \quad (4.6)$$

starting with the real emission part $\mathcal{K}_{ff'}^R(t, x, u)$. It includes implicitly IR cut-off $u - x > \Delta(x, u)$. The virtual part $\mathcal{K}_{ff'}^V(t, x, u)$ will be determined

unambiguously by imposing momentum sum rule. It includes implicitly $\delta_{ff'}\delta_{x=u}$. We will use as a basic building block the real emission part of the LL DGLAP kernel. In order to facilitate numerical calculation it is decomposed as follows

$$zP_{f'f}^{(0)}(z) = \delta_{f'f} \left(\frac{A_{ff}}{1-z} + F_{ff}(z) \right) + (1 - \delta_{f'f})F_{f'f}(z), \quad (4.7)$$

($z = x/u$), with the coefficients A_{ff} and functions $F_{f'f}(z)$ defined in Ref. [6]. Let us start with pure bremsstrahlung case, real emission part.

Case (A): DGLAP LL is introduced here as a reference case:

$$\mathcal{K}_{ff}^{\text{R(A)}}(t, x, u) = \frac{\alpha_S(Q_0 e^t)}{\pi} \frac{1}{u} P_{ff}^{(0)}(x/u) \theta_{u-x \geq u\epsilon}, \quad (4.8)$$

where ϵ is infinitesimally small and $z = x/u$.

Case (B): The argument in α_S is $(1-z)q = (1-z)e^t = k^T/u$; as advocated in Ref. [7]. For the IR cut-off we use $\Delta(t, u) = \lambda u e^{-t}$:

$$\mathcal{K}_{ff}^{\text{R(B)}}(t, x, u) = \frac{\alpha_S^{(0)}((1-x/u)e^t)}{\pi} \frac{1}{u} P_{ff}^{(0)}(x/u) \theta_{u-x \geq u\lambda e^{-t}}. \quad (4.9)$$

Case (C): The coupling constant α_S depends on the transverse momentum $k^T = (u-x)e^t$, while for an IR cut-off we choose $\Delta(t, u) = \Delta(t) = \lambda e^{-t}$. The kernel reads:

$$\mathcal{K}_{ff}^{\text{R(C)}}(t, x, u) = \frac{\alpha_S^{(0)}((u-x)e^t)}{\pi} \frac{1}{u} P_{ff}^{(0)}(x/u) \theta_{u-x \geq \lambda e^{-t}}. \quad (4.10)$$

The generalized kernels beyond the case of the pure bremsstrahlung, for the quark–gluon transitions, valid for all three cases $X = A, B, C$, we define as follows

$$x\mathcal{K}_{f'f}^{\text{R(X)}}(t, x, u) = \delta_{f'f} x\mathcal{K}_{f'f}^{\text{R(X)}}(t, x, u) + (1 - \delta_{f'f}) \frac{\alpha_S(e^t)}{\pi} F_{f'f}(z) \theta_{u-x > \Delta^{(X)}(u)}, \quad (4.11)$$

where α_S in the flavour changing elements have no z - or k^T -dependence and the IR cut-off $\Delta^{(X)}$ is the same as in the bremsstrahlung case.

Note that the case (C) is fully compatible with the CCFM evolution [2], except that for the gluon gluon transitions (bremsstrahlung) the non-Sudakov form-factor assuring the compatibility with BFKL [8] is not shown (although it is already present in the MC program)⁹.

⁹ The original CCFM was formulated for pure gluonstrahlung, without quark–gluon transitions.

As in Ref. [3], for cases (B) and (C), we also introduce slightly modified version of the quark–gluon changing kernels elements:

$$\begin{aligned}
x\mathcal{K}_{f'f}^{\text{R(B)'}}(t, x, u) &= \delta_{f'f} x\mathcal{K}_{f'f}^{\text{R(B)}}(t, x, u) \\
&\quad + (1 - \delta_{f'f}) \frac{\alpha_S((1-z)e^t)}{\pi} F_{f'f}(z) \theta_{1-z > \lambda e^{-t}}, \\
x\mathcal{K}_{f'f}^{\text{R(C)'}}(t, x, u) &= \delta_{f'f} x\mathcal{K}_{f'f}^{\text{R(C)}}(t, x, u) \\
&\quad + (1 - \delta_{f'f}) \frac{\alpha_S(u(1-z)e^t)}{\pi} F_{f'f}(z) \theta_{u-x > \lambda e^{-t}}, \quad (4.12)
\end{aligned}$$

with the same arguments of α_S and IR cut-off as for gluonstrahlung. New variants are referred to as cases (B') and (C'). One can go back from cases (B') and (C') to (B) and (C) by means of applying well behaving MC weight.

4.3. Form-factors

Sudakov form-factor resulting from resummation of the virtual part in the kernel was defined in Eq. (2.9). The virtual part of the kernel is determined through momentum sum rule, see Eq. (2.15), leading to the following expression

$$\begin{aligned}
\Phi_f(t_1, t_0|u) &= \sum_{f'} \int_{t_0}^{t_1} dt \int_0^u \frac{dx}{u} x \mathcal{K}_{f'f}^{\text{R}}(t, x, u) \\
&= \sum_{f'} \int_{t_0}^{t_1} dt \int_0^u \frac{dy}{u} (u-y) \mathcal{K}_{f'f}^{\text{R}}(t, u-y, u) \\
&= \sum_{f'} \int_{t_0}^{t_1} dt \int_0^1 dz uz \mathcal{K}_{f'f}^{\text{R}}(t, uz, u), \quad (4.13)
\end{aligned}$$

where $z \equiv x/u$ and $y \equiv u-x = (1-z)u$, see also Eq. (3.7).

Following decomposition of the LL kernel into three parts

$$zP_{f'f}^{(0)}(z) = \delta_{f'f} \frac{A_{ff}}{1-z} + \delta_{f'f} F_{ff}(z) + (1 - \delta_{f'f}) F_{f'f}(z), \quad (4.14)$$

the Sudakov form-factor for practical reasons is split into three corresponding parts:

$$\Phi_f(t_1, t_0|u) = \Phi_f(t_1, t_0|u) + \Phi_f^b(t_1, t_0|u) + \Phi_f^c(t_1, t_0|u). \quad (4.15)$$

We show in the following explicit expressions for the above form-factor components for most complicated case (C), referring the reader to Ref. [3] for simpler cases (A) and (B):

$$\begin{aligned}
\Phi_f(t_1, t_0|u) &= \int_{t_0}^{t_1} dt \int_0^1 dz \frac{\alpha_S((1-z)ue^t)}{\pi} \frac{A_{ff}}{1-z} \theta_{(1-z)u > \lambda e^{-t}} \\
&= A_{ff} \frac{2}{\beta_0} \varrho_2(\bar{t}_0 + \ln u, \bar{t}_1 + \ln u; \bar{t}_\lambda), \\
\Phi_f^b(t_1, t_0|u) &= \int_{t_0}^{t_1} dt \int_0^1 dz \frac{\alpha_S((1-z)ue^t)}{\pi} F_{ff}(z) \theta_{(1-z)u > \lambda e^{-t}}, \\
\Phi_f^c(t_1, t_0|u) &= \int_{t_0}^{t_1} dt \frac{\alpha_S(e^t)}{\pi} \sum_{f' \neq f} \int_0^1 dz F_{f'f}(z) \theta_{(1-z)u > \lambda e^{-t}}, \quad (4.16)
\end{aligned}$$

where $\bar{t}_i \equiv t_i - \ln A_0$, $\bar{t}_\lambda \equiv t_\lambda - \ln A_0$, $t_\lambda \equiv \ln \lambda$, while function ϱ_2 is defined in Appendix of Ref. [3] in terms of log functions. Two other components Φ_f^b and Φ_f^c are evaluated numerically for every MC event. This is feasible, provided one integration is performed analytically (typically that over $v = \ln(1-z)$) and second integration is done numerically, see Ref. [3] for the details.

4.4. Discussion

In all three cases (A–C) the distributions of the single forward step (parton emission) are relatively simple — they are built out of LL DGLAP kernels and α_S depending on t_i , z_i or k_T . The same distributions enter into form-factor of Eq. (4.13). Practical problems in the MC implementations are not so much in the distribution shapes as in the kinematic limits. We shall therefore concentrate in the following on this subject. For this purpose we will draw the limits of the available phase space in the emission of several gluons in the two-dimensional Sudakov logarithmic plane parametrized with variables (k^+, k^-) and $(\eta, \ln k^T)$ simultaneously. The same integration limits are used in the calculation of the form-factors. The translation from evolution times and lightcone variables, t_i, x_i , to rapidities and transverse momenta, $(\eta_i, \ln k_i^T)$, will be done using mapping of Section. (4.1) in all three cases (A–C)¹⁰.

In in Fig. 5 we start with case (C). The total emission phase space has triangular shape and is limited by maximum rapidity (from right) minimum

¹⁰ This mapping is primarily adequate for (C). In principle it could be different for (A) and (B).

k^T (from below) and conservation of lightcone plus variable, $k_i^+ < 2E_h x_{i-1}$. Within the above phase space, momenta of three emitted gluons $k_i^\mu, i = 1, 2, 3$ are represented by the black numbered circles. They are ordered in rapidity. The integration domains for the four consecutive form-factors $\Phi_{f_i}(t_i|t_{i-1})$ in the forward step distributions in Eqs. (3.11–3.12) are also shown in Fig. 5 as a triangle and three trapezoids.

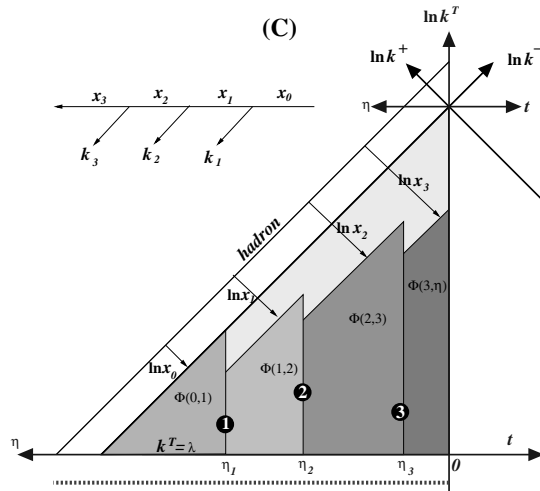


Fig. 5. Sudakov plane parametrized two sets of variables (k^+, k^-) and $(\eta, \ln k^T)$. Emission of three gluons. Their momenta $k_i^\mu, i = 1, 2, 3$ are marked as black numbered circles. Position of the Landau pole marked as dashed red line at $k^T = A_0$. Phase space limits as in case (C), that is CCFM evolution.

It is now interesting to compare the phase-space limits in the Sudakov plane between the case (C) and the two other cases (A) and (B). The corresponding plots are shown in Fig. 6. The main difference is in the shape of the lower infrared (IR) boundary of the emission phase space. In the case (A) of DGLAP it is at the same distance $\ln(1/\varepsilon)$ from the upper limit, hence rhomboid shapes with the variable widths and constant heights. In case (B) the IR limit in k^T is lowered by the factor x_{i-1} which grows after every emission, hence we see the trapezoids with the lower boundary descending deeper and deeper into smaller k^T . The above illustrates also why the construction of the MMC programs evolution type (B) served the role of an intermediate step on the way from DGLAP to CCFM.

Last not least, let us show kinematic limits in the extreme case of one $z_i \rightarrow 0$. This limit is treated in CCFM evolution better than in DGLAP, because CCFM in this limit coincides with the BFKL evolution [8]. Such a case is illustrated in Fig. 7, where the second emitted gluon is very hard,

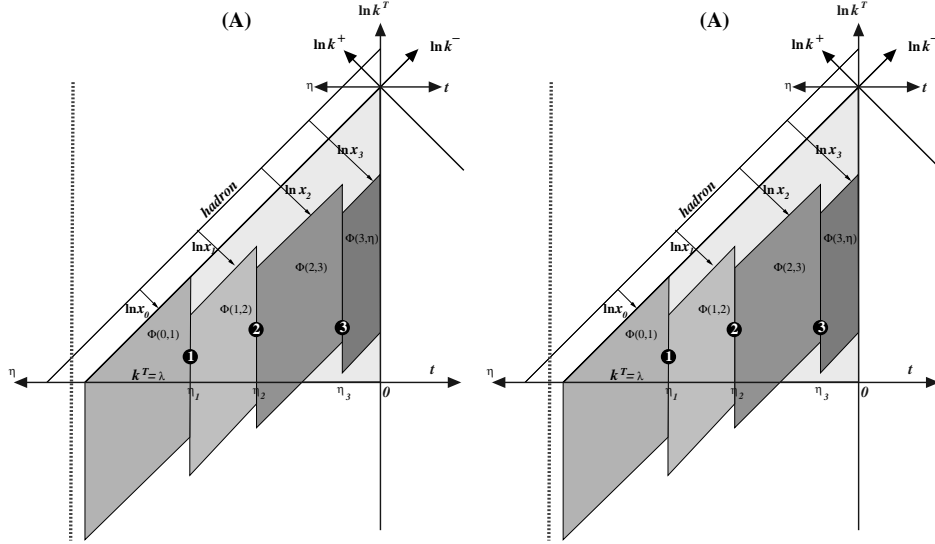


Fig. 6. Sudakov plane parametrized with (k^+, k^-) and $(\eta, \ln k^T)$. Emission of three gluons. Their momenta k_i^μ , $i = 1, 2, 3$ are marked as black numbered circles. Position of the Landau pole marked as dashed red line at (A) $Q = \Lambda_0$ or (B) $k^T = \Lambda_0$. Phase space limits as in case (A) and (B).

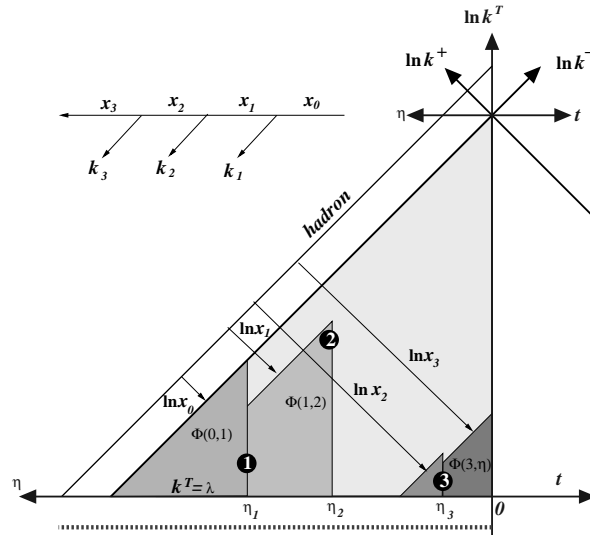


Fig. 7. Sudakov plane with emission of three gluons, case (C). For $z_2 \sim 0$ the second gluon has large k^T .

that is with high k^T , In fact larger than the scale of the hard process. (In this part of the phase space the non-Sudakov form-factor plays significant role.) The above kinematic region is properly included in the MMC case (C) and also in the CMC of Ref. [3].

5. Monte Carlo implementations

Studies of the DGLAP evolution, case (A), using the Markovian MCs were already covered in Refs. [9–11] in particular NLO case was extensively studied in Ref. [6]. The main aim of these papers was to show that MC method, although slower, is equally precise and more versatile as compared to older non-MC techniques, for example grid method based `QCDnum16` [12]. These MMCs were also used to test first examples of the constrained MCs [13, 14] for DGLAP-type evolution. The main advantage of MC method turns out to be very good and stable estimator of the error. The slowness of MMCs is mainly the problem in any attempt of fitting deep-inelastic ep data. Here, special pretabulation procedures are necessary, see Refs. [15]. The above studies of the evolution type (A) using MMCs were fairly complete, hence there is no need to repeat them here.

As already said, we do not show/repeat in this work tests of MMC type (A) and we will limit numerical results to comparisons of MMC *versus* non-MC program `APCheb` [16] for evolutions class (B) and (C). It should be stressed that `APCheb` was originally working only for DGLAP and was upgraded to evolutions type (B) and (C) for the purpose of the tests with MMCs. Comparisons of MMC and CMC programs for evolutions type (B) and (C) were also done and have been presented in Ref. [3]. In this way we have in our disposal three completely different programs (sometimes even four) which solve numerically evolution equations of all three types (A), (B) and (C) and provide identical results within precision of 0.2%!

5.1. Reusing MMC type (B) as type (C)

Historically, the MMC for evolution type (B) with $\alpha_S(e^t(1-z))$ and IR cutoff $1-z > \lambda e^{-t}$ was developed first, before CCFM-like scenario (C). While testing first versions of MMC type (C) the following observation was helpful. Examining carefully the probability distributions of the single forward step $\omega_{(i-1) \rightarrow i}$ of Eqs. (3.11,3.6) one may notice that the whole additional dependence on the x_{i-1} variable in $\omega_{(i-1) \rightarrow i}^{(C)}$ can be absorbed into λ and A_0 :

$$\omega_{(i-1) \rightarrow i}^{(C)}(\lambda, A_0) = \omega_{(i-1) \rightarrow i}^{(B)}(\lambda/x_{i-1}, A_0/x_{i-1}). \quad (5.1)$$

Of course, this is the consequence of the relations $\alpha_S(k_i^T) = \alpha_S(e^{t_i}(1-z_i) x_{i-1})$ and $k_i^T = e^{t_i}(1-z_i)x_{i-1} > \lambda$. As a results, we could in the tests of MMC

class (C) *reuse* the MMC for $\alpha_S(e^t(1-z))$ by means of resetting $\lambda \rightarrow \lambda/x_{i-1}$ and $\Lambda \rightarrow \Lambda/x_{i-1}$, before generating each single forward step. The above trick was quite helpful in testing MMC class (C), for pure bremsstrahlung.

6. Solving evolution equations with Chebyshev polynomials

In the previous section the Monte Carlo method for solving the evolution equations was presented. For the sake of the comparison, we are going to present an alternative method based on the expansion in the Chebyshev polynomials.

We start from the general form (2.1) of the evolution equations

$$\partial_t D_f(t, x) = \sum_{f'} \int_0^1 du \mathcal{K}_{ff'}(t, x, u) D_{f'}(t, u) \quad (6.1)$$

with the kernel (2.8). The momentum sum rule (2.11) imposed on the parton distributions allows to determine the virtual part of the kernel (2.8) from the condition (2.14). As a result, we arrive at the most general form of the evolution equations

$$\begin{aligned} \partial_t(xD_f(t, x)) &= \sum_{f'} \int_0^1 du x \mathcal{K}_{ff'}^R(t, x, u) D_{f'}(t, u) \\ &\quad - D_f(t, x) \sum_{f'} \int_0^1 du u \mathcal{K}_{f'f}^R(t, u, x). \end{aligned} \quad (6.2)$$

As an illustration, we consider in detail the evolution equations for the case (C) from Section 4.2. The evolution parameter t in this case is related to the rapidity $y = \eta$ of the emitted real parton by the relation (4.3), which now reads

$$k_T = 2E_h e^{-y}(u-x) = e^t(u-x). \quad (6.3)$$

Here u and x are the longitudinal momentum fraction before and after the emission. In the leading logarithmic approximation the real emission kernel takes the form

$$x \mathcal{K}_{ff'}^R(t, x, u) = \frac{\alpha_S(k_T)}{\pi} \frac{x}{u} P_{ff'}^{(0)}\left(\frac{x}{u}\right) \theta(u-x), \quad (6.4)$$

where $P_{ff'}^{(0)}$ are the leading order splitting functions. In order to avoid the Landau pole in α_S , we assume that the transverse momenta of the emitted partons are bounded from below,

$$k_T \geq \lambda \gg \Lambda_{\text{QCD}}. \quad (6.5)$$

This further restricts the momentum fractions u for the real emission (see the theta function in Eq. (4.10)):

$$u \geq x + \lambda e^{-t}. \quad (6.6)$$

Changing the integration variable, $z = x/u$, we obtain for the real emission part of the evolution equations (6.2)

$$\sum_{f'} \int_x^{z^{\text{R}}(x,t)} dz \frac{\alpha_S(e^t x(1-z)/z)}{\pi} P_{ff'}^{(0)}(z) \frac{x}{z} D_{f'}\left(t, \frac{x}{z}\right), \quad (6.7)$$

with the upper integration limit given by

$$z^{\text{R}}(x,t) = \frac{1}{1 + \lambda e^{-t}} > x. \quad (6.8)$$

For the virtual part of the evolution equations we interchange $u \leftrightarrow x$ in the kernel (6.4). Now, the conditions which restrict the u -values read

$$u < x, \quad k_{\text{T}} = e^t(x-u) \geq \lambda. \quad (6.9)$$

Changing the integration variable, $z = u/x$, in the second integral of Eqs. (6.2) we obtain for the virtual term

$$-x D_f(t, x) \sum_{f'} \int_0^{z^{\text{V}}(x,t)} dz \frac{\alpha_S(e^t x(1-z))}{\pi} z P_{f'f}^{(0)}(z), \quad (6.10)$$

where now

$$z^{\text{V}}(x,t) = 1 - \lambda e^{-t} > 0. \quad (6.11)$$

In summary, we find the following evolution equations

$$\begin{aligned} \partial_t(x D_f(t, x)) &= \sum_{f'} \int_x^{z^{\text{R}}(x,t)} dz \frac{\alpha_S(e^t x(1-z)/z)}{\pi} P_{ff'}^{(0)}(z) \frac{x}{z} D_{f'}\left(t, \frac{x}{z}\right) \\ &\quad - x D_f(t, x) \sum_{f'} \int_0^{z^{\text{V}}(x,t)} dz \frac{\alpha_S(e^t x(1-z))}{\pi} z P_{f'f}^{(0)}(z). \end{aligned} \quad (6.12)$$

These equations are complicated enough to be solved only numerically. In the next section we will present the method based on the expansion in the Chebyshev polynomials.

6.1. Chebyshev polynomial method

In this method we use the Chebyshev polynomials defined by

$$T_k(y) = \cos(k \arccos(y)), \quad y \in [-1, 1]. \quad (6.13)$$

The index $k = 0, 1, 2, \dots$ denotes the polynomial order. Fixing the order, $k = N$, we consider the equation $T_N(y) = 0$. It has N roots (nodes) given by

$$y_i = \cos \frac{\pi}{N}(i - 1/2), \quad i = 1, 2, \dots, N. \quad (6.14)$$

These roots allow to define the following discrete orthogonality relation for the set of the Chebyshev polynomials $\{T_0, T_1, \dots, T_{N-1}\}$:

$$\sum_{i=1}^N T_j(y_i) T_k(y_i) = C_j \delta_{jk}, \quad (6.15)$$

where $j, k = 0, 1, \dots, (N - 1)$. The coefficients $C_0 = N$ and $C_{j \geq 1} = N/2$.

A function $f(x)$ with $x \in [a, b]$ can be approximated with the help of the specified set of Chebyshev polynomials in following way

$$f(x) \approx \sum_{n=1}^N v_n c_n T_{n-1}(y(x)), \quad (6.16)$$

where $v_1 = 1/2$, $v_{n \geq 1} = 1$ and $y = y(x)$ is an arbitrary, invertible function which transforms $[a, b] \rightarrow [-1, 1]$. The coefficients c_n of the expansion can be calculated from the orthogonality relation (6.15),

$$c_n = \frac{2}{N} \sum_{i=1}^N f(x_i) T_{n-1}(y_i), \quad (6.17)$$

where $x_i = y^{-1}(y_i)$ are images of the roots (6.14) in the interval $[a, b]$. From relations (6.16) and (6.17) we see that one only needs the values $f(x_i)$ at the Chebyshev nodes to reconstruct the function at any other $x \in [a, b]$. This observation is a starting point of the method of the solution of the evolution equations (6.12). We simply solve them at the Chebyshev nodes $x = x_k$.

Therefore, writing Eqs. (6.12) in a prototype form,

$$\frac{dD(t, x_k)}{dt} = \int_{x_k}^{z(x_k, t)} dz P(t, z) D(t, x_k/z), \quad (6.18)$$

we consider the finite set of the first order differential equations for $k = 1, 2, \dots, N$. The integration on the r.h.s. needs the values of D at any point, thus we use the Chebyshev approximation

$$D(t, x_k/z) \approx \frac{2}{N} \sum_{n=1}^N \sum_{i=1}^N v_n D(t, x_i) T_{n-1}(y_i) T_{n-1}(y(x_k/z)). \quad (6.19)$$

Substituting into (6.18), we find the following set of equations

$$\frac{dD(t, x_k)}{dt} = \sum_{i=1}^N \mathcal{A}_{ki}(t) D(t, x_i) \quad (6.20)$$

which can easily be solved numerically [16]. The matrix $\mathcal{A}_{ki}(t)$ in these equations,

$$\mathcal{A}_{ki}(t) = \frac{2}{N} \sum_{n=1}^N v_n T_{n-1}(y_i) \int_{x_k}^{z(x_k,t)} dz P(t, z) T_{n-1}(y(x_k/z)), \quad (6.21)$$

is computed numerically in the process of finding the solution of Eqs. (6.20).

The differential equations which we consider need initial conditions at some initial scale $D(t = t_0, x)$. They are usually specified analytically such that the initial values $D(t_0, x_k)$ at the Chebyshev nodes are easily calculated.

The results of the comparison of the solutions of the evolution equations obtained using the Monte Carlo and Chebyshev methods are discussed in the next section. In general, a very good agreement between the results of these two methods is found.

7. Numerical results

Although our MMC program was systematically tested against non-MC programs `APCheb` and `QCDnum16` for all evolution types (A–C), we shall show examples of the numerical results for the more sophisticated and difficult evolution types (B) and (C).

Fig. 8 demonstrates distributions $x D_f(t, x)$ from MMC and `APCheb` [16] programs and the corresponding ratios `MMC/APCheb` for the evolution type (B), that is with $\alpha((1-z)Q)$. The four curves represent $x D_f(t, x)$ for $Q = e^t = 1, 10, 10^2, 10^3 \text{ GeV}$. The upper plots are for $f = G$, gluon while lower plots are for $f = q + \bar{q}$, quarks and antiquarks taken together. The starting quark and gluon distribution at $Q = e^t = 1 \text{ GeV}$ are defined exactly the same as in previous works of Refs. [9–11]. Results for all $Q = 1, 10, 10^2, 10^3 \text{ GeV}$ were obtained in the single MC run of $\sim 10^{10}$ MC events. As we see the distributions from two programs agree within the statistical MC error of about $\sim 0.2\%$.

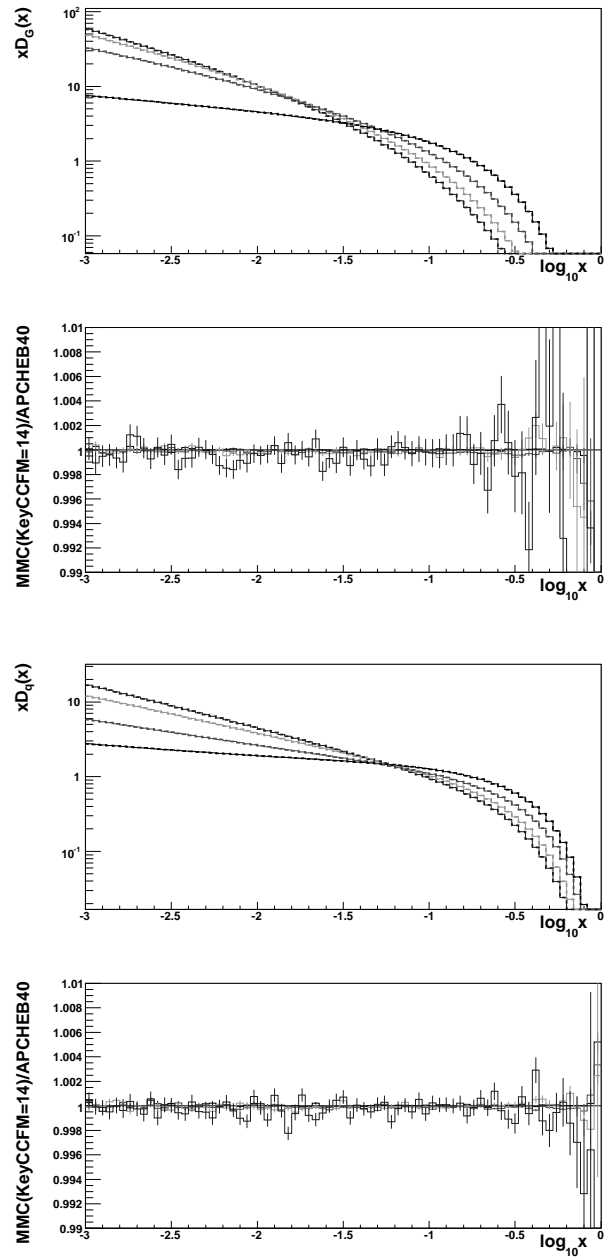


Fig. 8. For evolution type (B) with $\alpha((1-z)Q)$ plotted are distributions $x D_f(x)$ and their ratios MMC/APCheb for $f = \text{gluon}$ (upper) and $f = q + \bar{q}$ (lower plot).

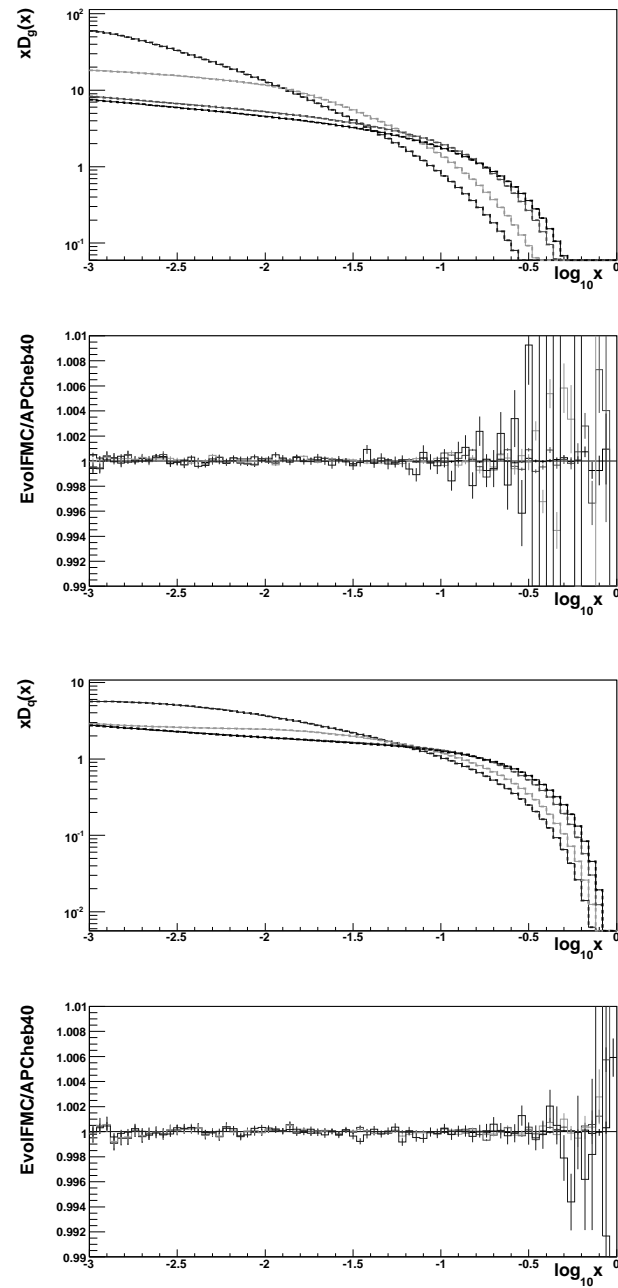


Fig. 9. For evolution type (C) with $\alpha(k^T)$ plotted are distributions $x D_f(x)$ and their ratios MMC/APCheb for $f = \text{gluon}$ (upper) and $f = q + \bar{q}$ (lower plot).

In Fig. 9 we show the same type of comparison of MMC and APCh**eb**, but for evolution type (C). Again precision agreement within the statistical MC error is reached.

For the LL DGLAP, case (A), we have reproduced results of Ref. [9] with smaller statistical errors and removing certain numerical biases which were seen in this paper in the gluon case, $f = G$. We do not show explicitly the corresponding numerical results.

8. Summary

We have developed and tested Markovian MC programs for two additional types of the QCD evolution equations, in addition to DGLAP. One of them is identical with the so-called all-loop CCFM (modulo non-Sudakov form-factor). The corresponding MC programs were tested to a high-precision level by means of comparison with the other non-MC program APCh**eb**. MMC of this work is also used to test another class of the constrained MCs in other independent works, for the same class to QCD evolutions. The aim of these exercises is to build basis for the new parton shower implementations. The mapping of the evolution variables into four-momenta was also introduced and tested.

We would like to thank A. Siódmok for useful discussions. We acknowledge the warm hospitality of the CERN Physics Department, where part of this work was done.

REFERENCES

- [1] L.N. Lipatov, *Sov. J. Nucl. Phys.* **20**, 95 (1975); V.N. Gribov, L.N. Lipatov, *Sov. J. Nucl. Phys.* **15**, 438 (1972); G. Altarelli, G. Parisi, *Nucl. Phys.* **126**, 298 (1977); Yu.L. Dokshitzer, *Sov. Phys. JETP* **46**, 64 (1977).
- [2] M. Ciafaloni, *Nucl. Phys.* **B296**, 49 (1988); S. Catani, F. Fiorani, G. Marchesini, *Phys. Lett.* **B234**, 339 (1990); *Nucl. Phys.* **B336**, 18 (1990); G. Marchesini, *Nucl. Phys.* **B445**, 49 (1995).
- [3] S. Jadach, W. Płaczek, M. Skrzypek, P. Stephens, Z. Waś, Constrained MC for QCD evolution with rapidity ordering and minimum kT, 2007, Report FJSPAN-IV-2007-3, CERN-PH-TH/2007-059 [hep-ph/0703281](#).
- [4] S. Jadach, M. Skrzypek, Z. Waś, [hep-ph/0701174](#).
- [5] S. Jadach, Practical guide to Monte Carlo, 1999, [physics/9906056](#), also available from <http://home.cern.ch/~jadach> (unpublished).
- [6] K. Golec-Biernat, S. Jadach, W. Płaczek, M. Skrzypek, *Acta Phys. Pol. B* **37**, 1785 (2006) [hep-ph/0603031](#).

- [7] D. Amati, A. Bassetto, M. Ciafaloni, G. Marchesini, G. Veneziano, *Nucl. Phys.* **B173**, 429 (1980).
- [8] L.N. Lipatov, *Sov. J. Nucl. Phys.* **23**, 338 (1976); E.A. Kuraev, L.N. Lipatov, V.S. Fadin, *Sov. Phys. JETP* **45**, 199 (1977); I.I. Balitsky, L.N. Lipatov, *Sov. J. Nucl. Phys.* **28**, 822 (1978); L.N. Lipatov, *Sov. Phys. JETP* **63**, 904 (1986).
- [9] S. Jadach, M. Skrzypek, *Acta Phys. Pol. B* **35**, 745 (2004) [hep-ph/0312355](#).
- [10] S. Jadach, M. Skrzypek, *Nucl. Phys. Proc. Suppl.* **157**, 241 (2006).
- [11] W. Płaczek, K. Golec-Biernat, S. Jadach, M. Skrzypek, [arXiv:0704.3344](#) [[hep-ph](#)].
- [12] M. Botje, QCDNUM16: A fast QCD evolution program, 1977, ZEUS Note 97-066, <http://www.nikhef.nl/h24/qcdcode/>.
- [13] S. Jadach, M. Skrzypek, *Comput. Phys. Commun.* **175**, 511 (2006) [hep-ph/0504263](#).
- [14] S. Jadach, M. Skrzypek, *Acta Phys. Pol. B* **36**, 2979 (2005) [hep-ph/0504205](#).
- [15] H. Jung, G.P. Salam, *Eur. Phys. J.* **C19**, 351 (2001) [hep-ph/0012143](#).
- [16] K. Golec-Biernat, [APChEb40](#), the Fortran code available on the request from the author, unpublished.