DESY 14-060
June 2014

The CCFM uPDF evolution
uPDFevolv
Version 1.0.00

F. Hautmann[1,2,3], H. Jung[4,5], S. Taheri Monfared[6]
[1]Dept. of Physics and Astronomy, University of Sussex, Brighton BN1 9QH
[2] Rutherford Appleton Laboratory, Chilton OX11 0QX
[3]Dept. of Theoretical Physics, University of Oxford, Oxford OX1 3NP
[4]DESY, Hamburg, FRG
[5]University of Antwerp, Antwerp, Belgium
[6]School of Particles and Accelerators, Institute for Research in Fundamental Sciences (IPM), P.O.Box 19395-5531, Tehran, Iran

Abstract

uPDFevolv is an evolution code for TMD parton densities using the CCFM evolution equation. A description of the underlying theoretical model and technical realisation is given together with a detailed program description, with emphasis on parameters the user may want to change.

PROGRAM SUMMARY

Title of Program: uPDFevolv 1.0.00
Computer for which the program is designed and others on which it is operable: any with standard Fortran 77 (gfortran) and C++, tested on Linux, MAC
Programming Language used: FORTRAN 77, C++
High-speed storage required: No
Separate documentation available: No
Keywords: QCD, small x, high-energy factorization, k_t-factorization, CCFM, unintegrated PDF (uPDF), transverse momentum dependent PDF (TMD)
Nature of physical problem: At high energies collisions of hadrons are described by parton densities dependent on the longitudinal momentum fraction x, the transverse momentum k_t and the evolution scale p (transverse momentum dependent (TMD) or unintegrated parton density functions (uPDF)). The evolution of the parton density with the scale p valid at both small and moderate x is given by the CCFM evolution equation
Method of solution: Since the CCFM evolution equation cannot be solved analytically, a Monte Carlo approach is applied, simulating at each step of the evolution the full four-momenta of the initial state partonic cascade.
Restrictions on the complexity of the problem: None

Other Program used: ROOT for plotting the result.
Download of the program: `http://www.desy.de/~jung/ccfm_uPDF`
Unusual features of the program: None

1 Theoretical Input

1.1 CCFM evolution equation and Transverse Momentum Dependent PDFs

QCD calculations of multiple-scale processes and complex final-states require in general transverse-momentum dependent (TMD), or unintegrated, parton density and parton decay functions [?, ?, ?, ?, ?, ?, ?, ?, ?, ?]. TMD factorization has been proven recently [?] for inclusive and semi-inclusive deep-inelastic scattering (DIS). For special processes in hadron-hadron scattering, like heavy flavor or heavy boson (including Higgs) production, TMD factorization holds in the high-energy limit (small x) [?, ?, ?].

In the framework of high-energy factorization [?, ?] the deep-inelastic scattering cross section can be written as a convolution in both longitudinal and transverse momenta of the TMD parton density function $\mathcal{A}\left(x, k_t, \mu\right)$ with off-shell partonic matrix elements, as follows

$$\sigma_j(x, Q^2) = \int_x^1 dz \int d^2k_t \; \hat{\sigma}_j(x, Q^2, z, k_t) \; \mathcal{A}\left(z, k_t, p\right), \tag{1}$$

with the DIS cross sections σ_j ($j = 2, L$) related to the structure functions F_2 and F_L by $\sigma_j = 4\pi^2 F_j/Q^2$. The hard-scattering kernels $\hat{\sigma}_j$ of Eq. (??) are k_t-dependent and the evolution of the transverse momentum dependent gluon density $\mathcal{A}$ is obtained by combining the resummation of small-x logarithmic contributions [?, ?, ?] with medium-x and large-x contributions to parton splitting [?, ?, ?] according to the CCFM evolution equation [?, ?, ?].

The factorization formula (??) allows one to resum logarithmically enhanced $x \to 0$ contributions to all orders in perturbation theory, both in the hard scattering coefficients and in the parton evolution, taking fully into account the dependence on the factorization scale p and on the factorization scheme [?, ?].

The CCFM evolution equation [?, ?, ?] is an exclusive equation for final state partons and includes finite-x contributions to parton splitting. It incorporates soft gluon coherence for any value of x.

1.1.1 Gluon distribution

The evolution equation for the TMD gluon density $\mathcal{A}(x, k_t, p)$, depending on x, k_t and the evolution variable p, is

$$\begin{aligned} \mathcal{A}(x, k_t, p) &= \mathcal{A}_0(x, k_t, p) + \int \frac{dz}{z} \int \frac{dq^2}{q^2} \, \Theta(p - zq) \\ &\times \; \Delta_s(p, zq) P(z, q, k_t) \; \mathcal{A}\left(\frac{x}{z}, k_t + (1-z)q, q\right) \quad , \end{aligned} \tag{2}$$

where the Θ-function specifies the ordering condition of the evolution.

The first term in the right hand side of Eq. (??) is the contribution of the non-resolvable branchings between the starting scale q_0 and the evolution scale p, and is given by

$$\mathcal{A}_0(x, k_t, p) = \mathcal{A}_0(x, k_t, q_0) \; \Delta_s(p, q_0), \tag{3}$$

where Δ_s is the Sudakov form factor, and $\mathcal{A}_0(x, k_t, q_0)$ is the starting distribution at scale q_0. The integral term in the right hand side of Eq. (**??**) gives the k_t-dependent branchings in terms of the Sudakov form factor Δ_s and unintegrated splitting function P. The Sudakov form factor Δ_s is given by

$$\Delta_s(p, q_0) = \exp\left(- \int_{q_0^2}^{p^2} \frac{dq^2}{q^2} \int_0^{1-q_0/q} dz \, \frac{\bar{\alpha}_s(q^2(1-z)^2)}{1-z} \right), \tag{4}$$

with $\overline{\alpha}_s = C_A \alpha_s/\pi = 3\alpha_s/\pi$.

For application in Monte Carlo event generators, like CASCADE [**?**, **?**], it is of advantage to write the CCFM evolution equation in differential form:

$$p^2 \frac{d}{dp^2} \frac{x\mathcal{A}(x, k_t, p)}{\Delta_s(p, q_0)} = \int dz \, \frac{d\phi}{2\pi} \, \frac{P(z, p/z, k_t)}{\Delta_s(p, q_0)} \, x'\mathcal{A}(x', k_t', p/z), \tag{5}$$

where the splitting variable x' is given by $x' = x/z$, $k_t{}' = q_t(1-z)/z + k_t$, and ϕ is the azimuthal angle of q_t.

For the evolution of the parton densities, however, a forward evolution approach, starting from the low scale q_0 towards the hard scale p, is used.

The splitting function $P_{gg}(z_i, q_i, k_{ti})$ for branching i is given by [**?**] (set by `Ipgg=1`, `ns=1` in uPDFevolv)

$$\begin{aligned} P_{gg}(z_i, q_i, k_{ti}) &= \bar{\alpha}_s(q_i^2(1-z_i)^2) \left(\frac{1}{1-z_i} - 1 + \frac{z_i(1-z_i)}{2} \right) \\ &+ \bar{\alpha}_s(k_{ti}^2) \left(\frac{1}{z_i} - 1 + \frac{z_i(1-z_i)}{2} \right) \Delta_{ns}(z_i, q_i^2, k_{ti}^2) \end{aligned} \tag{6}$$

where Δ_{ns} is the non-Sudakov form factor defined by

$$\log \Delta_{ns} = -\bar{\alpha}_s(k_{ti}^2) \int_0^1 dz' \left(\frac{1}{z'} - 1 + \frac{z'(1-z')}{2} \right) \int \frac{dq^2}{q^2} \Theta(k_{ti} - q)\Theta(q - z'q_{ti}). \tag{7}$$

In addition to the full splitting function, simplified versions are useful in applications and are made available. One uses only the singular parts of the splitting function (set by `Ipgg=0`, `ns=0` in uPDFevolv):

$$P_{gg}(z, q, k_t) = \frac{\bar{\alpha}_s(q^2)}{1-z} + \frac{\bar{\alpha}_s(k_t^2)}{z} \Delta_{ns}(z, q^2, k_t) \tag{8}$$

with

$$\log \Delta_{ns} = -\bar{\alpha}_s(k_{ti}^2) \int_0^1 \frac{dz'}{z'} \int \frac{dq^2}{q^2} \Theta(k_{ti} - q)\Theta(q - z'q_{ti}). \tag{9}$$

Another uses $\alpha_s(q^2)$ also for the small z part (set by `Ipgg=2`, `ns=2` in uPDFevolv):

$$P_{gg}(z, q, k_t) = \frac{\bar{\alpha}_s(q^2)}{1-z} + \frac{\bar{\alpha}_s(q^2)}{z} \Delta_{ns}(z, q^2, k_t) \tag{10}$$

with

$$\log \Delta_{ns} = -\int_0^1 \frac{dz'}{z'} \int \frac{dq^2}{q^2} \bar{\alpha}_s(q^2) \Theta(k_{ti} - q) \Theta(q - z' q_{ti}). \tag{11}$$

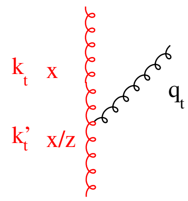

Figure 1: Gluon branching

In general a four-momentum **a** can be written in light-cone variables as $\mathbf{a} = (a^+, a^-, a_T)$ with a^+ and a^- being the light-cone components and a_T being the transverse component. The CCFM (as well as the BFKL) evolution depends only on one of the light-cone components. Assuming that the other one can be neglected, this leads to the condition that the virtuality of the parton propagator $a^2 = 2a^+a^- - a_T^2$ should be dominated by the transverse component, while the contribution from the longitudinal components is required to be small. The condition that $a^+a^- = 0$ leads to the so-called consistency constraint (see Fig. **??**), which has been implemented in different forms (set by `Ikincut=1,2,3` in uPDFevolv)

$$q_t^2 < \frac{k_t^2}{z} \quad \text{LDC [?, ?]} \tag{12}$$

$$q_t^2 < \frac{(1-z)k_t^2}{z} \quad \text{[?]} \tag{13}$$

$$k_t'^2 < \frac{k_t^2}{z} \quad \text{BFKL [?]} \tag{14}$$

1.1.2 Valence Quarks

Using the method of [**?**, **?**] valence quarks are included in the branching evolution at the transverse-momentum dependent level according to

$$\begin{aligned} xQ_v(x, k_t, p) &= xQ_{v0}(x, k_t, p) + \int \frac{dz}{z} \int \frac{dq^2}{q^2} \Theta(p - zq) \\ &\times \Delta_s(p, zq) P_{qq}(z, q, k_t)\, xQ_v\left(\frac{x}{z}, k_t + (1-z)q, q\right) \quad , \end{aligned} \tag{15}$$

where p is the evolution scale. The quark splitting function P_{qq} is given by

$$P_{qq}(z, q, k_t) = \frac{C_F}{2\pi} \alpha_s\left(q^2(1-z)^2\right) \frac{1+z^2}{1-z} \quad . \tag{16}$$

In Eqs. (**??**),(**??**) the non-Sudakov form factor is not included, unlike the CCFM kernel given in the appendix B of [**?**], because we only associate this factor with $1/z$ terms. The term xQ_{v0}

in Eq. (**??**) is the contribution of the non-resolvable branchings between starting scale q_0 and evolution scale p, given by

$$xQ_{v0}(x,k_t,p) = xQ_{v0}(x,k_t,q_0)\Delta_s(p,q_0) \quad , \tag{17}$$

where Δ_s is the Sudakov form factor.

1.1.3 Sea quarks

For a complete description of the final states also the contribution from sea-quarks needs to be included. We include splitting functions P_{ab} according to

$$\begin{aligned}
P_{gg}(z) &= \bar{\alpha}_s \left(\frac{1}{1-z_i} - 1 + \frac{z_i(1-z_i)}{2} \right) + \bar{\alpha}_s \left(\frac{1}{z_i} - 1 + \frac{z_i(1-z_i)}{2} \right) \Delta_{ns} & (18)\\
P_{qg}(z) &= \bar{\alpha}_s \frac{1}{4C_A} \left(z^2 + (1-z)^2 \right) & (19)\\
P_{gq}(z) &= \bar{\alpha}_s \frac{C_F}{2C_A} \left(\frac{1+(1-z)^2}{z} \right) & (20)\\
P_{qq}(z) &= \bar{\alpha}_s \frac{C_F}{2C_A} \left(\frac{1+z^2}{1-z} \right) & (21)
\end{aligned}$$

with $\bar{\alpha}_s = C_A \alpha_s/\pi$, $C_A = 3$ and $C_F = 4/3$.

The $g \to q\bar{q}$ splitting has been calculated in a k_t- factorized form in [**?**],

$$P_{qg}(z,\tilde{q},k_t) = \bar{\alpha}_s \frac{1}{4C_A} \left[\frac{\tilde{q}^2}{\tilde{q}^2 + z(1-z)\, k_t^2} \right]^2 \left(z^2 + (1-z)^2 + 4z^2(1-z)^2 \, \frac{k_t^2}{\tilde{q}^2} \right) \tag{22}$$

with $\tilde{q} = q - zk_t$, and $q(k_t)$ being the transverse momentum of the quark (gluon).

The evolution equation for the TMD sea-quark density $\mathcal{S}(x,k_t,p)$, depending on x, k_t and the evolution variable p, is

$$\begin{aligned}
\mathcal{S}(x,k_t,p) &= \mathcal{S}_0(x,k_t,p) \\
&+ \int \frac{dz}{z} \int \frac{dq^2}{q^2}\, \Theta(p-zq)\, \Delta_s(p,zq) P_{qg}(z,q,k_t)\, \mathcal{A}\left(\frac{x}{z}, k_t + (1-z)q, q \right) \\
&+ \int \frac{dz}{z} \int \frac{dq^2}{q^2}\, \Theta(p-zq) \Delta_s(p,zq) P_{qq}(z,q,k_t)\, \mathcal{S}\left(\frac{x}{z}, k_t + (1-z)q, q \right), \quad (23)
\end{aligned}$$

where $\mathcal{S}_0(x,k_t,p)$ is the non-resolvable branching probability similar to Eqs. (**??**),(**??**).

The evolution of the TMD gluon density including the contribution from quarks is given by

$$\begin{aligned}
\mathcal{A}(x,k_t,p) &= \mathcal{A}_0(x,k_t,p) \\
&+ \int \frac{dz}{z} \int \frac{dq^2}{q^2}\, \Theta(p-zq) \Delta_s(p,zq) P_{gg}(z,q,k_t)\, \mathcal{A}\left(\frac{x}{z}, k_t + (1-z)q, q \right) \\
&+ \int \frac{dz}{z} \int \frac{dq^2}{q^2}\, \Theta(p-zq) \Delta_s(p,zq) P_{gq}(z,q,k_t)\, \mathcal{S}\left(\frac{x}{z}, k_t + (1-z)q, q \right). \quad (24)
\end{aligned}$$

1.1.4 Monte Carlo solution of the CCFM evolution equations

The evolution equations Eqs.(**??**,**??**) are integral equations of the Fredholm type

$$f(x) = f_0(x) + \lambda \int_a^b K(x,y) f(y) dy$$

and can be solved by iteration as a Neumann series

$$\begin{aligned} f_1(x) &= f_0(x) + \lambda \int_a^b K(x,y) f_0(y) dy \\ f_2(x) &= f_0(x) + \lambda \int_a^b K(x,y_1) f_0(y_1) dy_1 + \lambda^2 \int_a^b \int_a^b K(x,y_1) K(y_1,y_2) f_0(y_2) dy_2 dy_1 \\ &\dots \end{aligned} \tag{25}$$

using the kernel $K(x,y)$, with the solution

$$f(x) = \lim_{n\to\infty} \sum_{i=0}^{n} f_i(x). \tag{26}$$

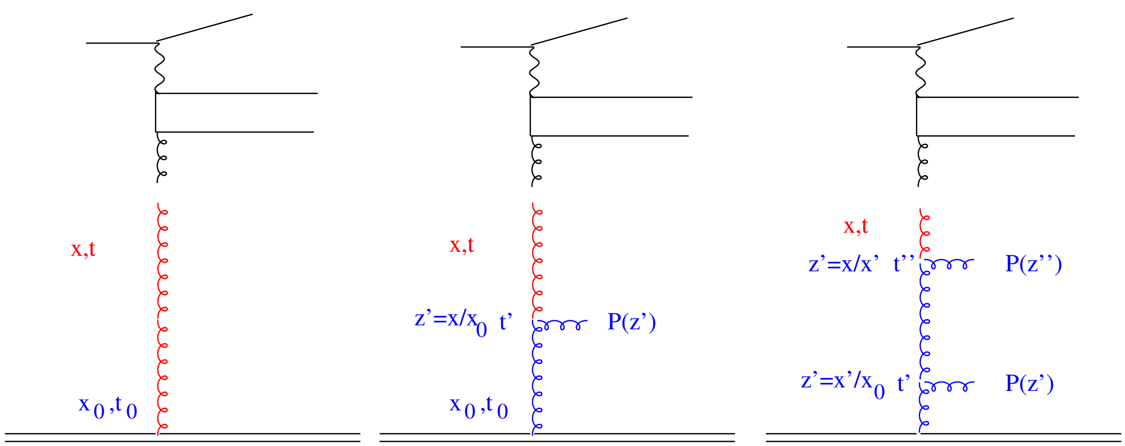

Figure 2: Evolution by iteration

Applying this to the evolution equations Eqs.(**??**,**??**), we identify f_0 with the first term in eqs.(**??**), where we use for simplicity here and in the following $\Delta_s(p) = \Delta_s(p, q_0)$:

$$\mathcal{A}_0(x, k_t, p) = \mathcal{A}_0(x, k_t) \Delta_s(p). \tag{27}$$

The first iteration involves one branching:

$$\mathcal{A}_1(x,k_t,p) = \mathcal{A}_0(x,k_t)\Delta_s(p) + \int_x^1 \frac{dz'}{z'}\int_{q_0}^p \frac{dq'^2}{q'^2}\Theta(p-z'q')\frac{\Delta_s(p)}{\Delta_s(zq')}\tilde{P}(z')\mathcal{A}_0(x/z',k_t',q'). \quad (28)$$

The second iteration involves two branchings,

$$\begin{aligned}
\mathcal{A}_2(x,k_t,p) &= \mathcal{A}_0(x,k_t)\Delta(p) \\
&+ \int_x^1 \frac{dz'}{z'}\int_{q_0}^p \frac{dq'^2}{q'^2}\Theta(p-z'q')\frac{\Delta(p)}{\Delta(q')}\tilde{P}(z')\mathcal{A}_1(x/z',k_t',q') \\
&= \mathcal{A}_0(x,k_t)\Delta(p) + \frac{\alpha_s}{2\pi}\int_x^1 \frac{dz'}{z'}\int_{q_0}^p \frac{dq'^2}{q'^2}\Theta(p-z'q')\frac{\Delta_s(p)}{\Delta_s(zq')}\tilde{P}(z')\mathcal{A}_0(x/z',k_t',q') \\
&+ \left(\frac{\alpha_s}{2\pi}\right)^2 \int_x^1 \frac{dz'}{z'}\int_{q_0}^p \frac{dq'^2}{q'^2}\Theta(p-z'q')\frac{\Delta_s(p)}{\Delta_s(z'q')}\tilde{P}(z') \\
&\times \int_x^1 \frac{dz''}{z''}\int_{q_0}^p \frac{dq''}{q''}\Theta(p-z''q'')\frac{\Delta_s(p)}{\Delta_s(z''q'')}\tilde{P}(z'')\mathcal{A}_0(z''/z',k_t'',q''), \quad (29) \\
\mathcal{A}_3(x,k_t,p) &= \cdots \\
&\vdots
\end{aligned}$$

In a Monte Carlo (MC) solution [?, ?] we evolve from q_0 to a value q' obtained from the Sudakov factor $\Delta_s(q',q_0)$ (for a schematic visualisation of the evolution see fig. **??**). Note that the Sudakov factor $\Delta_s(q',q_0)$ gives the probability for evolving from q_0 to q' without resolvable branching. The value q' is obtained from solving for q':

$$R = \Delta_s(q',q_0), \quad (30)$$

for a random number R in $[0,1]$.

If $q' > p$ then the scale p is reached and the evolution is stopped, and we are left with just the first term without any resolvable branching. If $q' < p$ then we generate a branching at q' according to the splitting function $\tilde{P}(z')$, as described below, and continue the evolution using the Sudakov factor $\Delta_s(q'',q')$. If $q'' > p$ the evolution is stopped and we are left with just one resolvable branching at q'. If $q'' < p$ we continue the evolution as described above. This procedure is repeated until we generate $q > p$. By this procedure we sum all kinematically allowed contributions in the series $\sum f_i(x,p)$ and obtain an MC estimate of the parton distribution function.

With the Sudakov factor Δ_s and using

$$\frac{\partial}{\partial q'^2}\Delta_s(p,zq') = \frac{\partial}{\partial q'^2}\frac{\Delta_s(p)}{\Delta_s(zq')} = \frac{\Delta_s(p)}{\Delta_s(zq')}\left[\frac{1}{q'^2}\right]\int^{z_{max}} dz\tilde{P}(z),$$

we can write the first iteration of the evolution equation as

$$\begin{aligned}\mathcal{A}_1(x,k_t,p) &= \mathcal{A}_0(x,k_t,p) \\ &+ \int_x^1 \frac{dz'}{z'}\int_{q_0}^{p} d\Delta_s(p,z'q')\tilde{P}(z')\mathcal{A}_0(x/z',k_t',q')\left[\int^{z_{max}} dz\tilde{P}(z)\right]^{-1}. \end{aligned} \quad (31)$$

The integrals can be solved by a Monte Carlo method [**?**]: z is generated from

$$\int_{z_{min}}^{z} dz'\tilde{P}(z') = R_1\int_{z_{min}}^{z_{max}} dz'\tilde{P}(z'), \quad (32)$$

with R_1 being a random number in $[0,1]$, and q' is generated from

$$\begin{aligned} R_2 &= \int_{-\infty}^{x} f(x')dx' = F(x) \\ &= \int_{zq}^{p} \frac{\partial}{\partial q'^2}\left(\frac{\Delta_s(p)}{\Delta_s(zq')}\right) dq'^2 \\ &= \Delta_s(p,zq') \end{aligned} \quad (33)$$

solving for q', using z from above and another random number R_2 in [0,1].

This completes the calculation on the first splitting. This procedure is repeated until $q' > p$ and the evolution is stopped.

With z' and q' selected according to the above the first iteration of the evolution equation yields

$$\begin{aligned} x\mathcal{A}_1(x,k_t,p) &= x\mathcal{A}_0(x,k_t)\Delta_s(p) \\ &+ \sum_i \tilde{P}(z_i')x_i'\mathcal{A}_0(x_i',k_{t1}',q_i')\left[\int^{z_{max}} dz\tilde{P}(z)\right]^{-1}, \end{aligned} \quad (34)$$

with $x_i' = x/z_i$.

1.1.5 Normalisation of gluon and quark distributions

The valence quark densities are normalised so that they fulfil for every p the flavor sum rule.

The gluon and sea quark densities are normalised so that for every p

$$\int_0^1 dx\int_0^\infty dk_t^2 x\mathcal{A}(x,k_t,q_0) = \int_0^1 dx\int_0^\infty dk_t^2\left(x\mathcal{A}(x,k_t,p) + x\mathcal{S}(x,k_t,p)\right). \quad (35)$$

1.2 Computational Techniques: CCFM Grid

When using the CCFM evolution in a fit program to determine the starting distribution $\mathcal{A}_0(x)$, a full MC solution [?, ?] is no longer suitable, since it is time consuming and suffers from numerical fluctuations. Instead a convolution method introduced in [?, ?] is used. The kernel $\tilde{\mathcal{A}}\left(x'', k_t, p\right)$ is determined once from the Monte Carlo solution of the CCFM evolution equation, and then folded with the non-perturbative starting distribution $\mathcal{A}_0(x)$,

$$\begin{aligned} x\mathcal{A}(x, k_t, p) &= x \int dx' \int dx'' \mathcal{A}_0(x') \tilde{\mathcal{A}}\left(x'', k_t, p\right) \delta(x' x'' - x) \\ &= \int dx' \mathcal{A}_0(x') \cdot \frac{x}{x'}\, \tilde{\mathcal{A}}\left(\frac{x}{x'}, k_t, p\right). \end{aligned} \quad (36)$$

The kernel $\tilde{\mathcal{A}}$ incorporates all of the dynamics of the evolution, including Sudakov form factors and splitting functions. It is determined on a grid of $50 \otimes 50 \otimes 50$ bins in x, k_t, p. The binning in the grid is logarithmic, except for the longitudinal variable x where we use 40 bins in logarithmic spacing below 0.1, and 10 bins in linear spacing above 0.1.

Using this method, the complete coupled evolution of gluon and sea quarks is more complicated, since it is no longer a simple convolution of the kernel with the starting distribution. To simplify the approach, here we allow only for one species of partons at the starting scale, either gluons or sea-quarks. During evolution the other species will be generated. This approach, while convenient for QCD fits, has the feature that sea-quarks, in the case of gluons only at q_0, are generated with perturbative transverse momenta ($k_t > k_{t\ cut}$), without contribution from the soft (non-perturbative) region.

1.3 Functional Forms for starting distribution

1.3.1 Standard parametrisation

For the starting distribution $\mathcal{A}_0$, at the starting scale q_0, the following form is used:

$$x\mathcal{A}_0(x, k_t, q_0) = A_1 x^{-A_2} \cdot (1-x)^{A_3} \left(1 - A_4 x + A_5 \sqrt{x} + A_6 x^2\right) \exp[-k_t^2/\sigma^2] \ , \quad (37)$$

with $\sigma^2 = q_0^2/2$ and free parameters $A_1, \ldots, A_6$.

Valence quarks are treated using the method of [?, ?, ?] with starting distributions at scale q_0 parameterized using standard collinear pdfs (set by `Ipdf` in uPDFevolv) as

$$xQ_{v0}(x, k_t, q_0) = xQ_{v\text{coll.pdf}}(x, q_0)\ \exp[-k_t^2/\sigma^2] \ . \quad (38)$$

with $\sigma^2 = q_0^2/2$. At every scale p the flavor sum rule is fulfilled for valence quarks.

1.3.2 Saturation ansatz

A saturation ansatz for the starting distribution $\mathcal{A}_0$ at scale q_0 is available, following the parameterisation of the saturation model by Eq.(18) of [?],

$$x\mathcal{A}_{sat} = \frac{1}{\alpha_s} \frac{3\sigma_0}{4\pi^2} R_0^2(x) k_t^2 \exp\left(-R_0^2(x) k_t^2\right), \quad (39)$$

with $R_0^2(x) = (x/x_0)^\lambda$. The free parameters are $\sigma_0 = A_2$, $\lambda = A_3$, $x_0 = A_4$ and $\alpha_s = A_5$. In order to be able to use this type of parameterisation over the full x range, an additional factor of $(1-x)^{A_6}$ (see [**?**]) is applied.

1.4 Plotting TMDs

A simple plot program is included in the package. For a graphical web interface use TMD-PLOTTER [**?**].

1.5 Application

The evolution of the TMD gluon density has been used to perform fits to the DIS precision data [**?**, **?**], as described in detail in [**?**].

2 Description of the program components

2.1 Program history

```
*______________________________________________________________________
uPDFevolv
*           Version  10000
*           first public release
*______________________________________________________________________
```

2.2 Subroutines and functions

The source code of uPDFevolv and this manual can be found under:
`http://www.desy.de/~jung/uPDFevolv/`

`sminit`	to initialise
`sminfn`	to generate starting distributions in x and k_t
`smbran`	to simulate perturbative branchings
`splittgg`	to generate $g \to gg$ splitting via P_{gg}
`splittgq`	to generate $g \to qq$ splitting via P_{gq}
`splittqg`	to generate $q \to gq$ splitting via P_{qq}
`splittqq`	to generate $q \to qg$ splitting via P_{qq}
`szvalnew`	to calculate z values for $g \to gg$ splitting
`smqtem`	to generate t from the corresponding Sudakov factor
`updfgrid`	to build, fill and normalise the updf grid.
`asbmy(kt)`	to calculate $\frac{C_A}{\pi}\alpha_s(k_t)$

Utility routines:

`evolve tmd`	Main routine to perform CCFM evolution
`updfread`	example program to read and plot the results
`gadap`	1-dimensional Gauss integration routine
`gadap2`	2-dimensional Gauss integration routine

`divdif`	linear interpolation routine (CERNLIB)
`ranlux`	Random number generator `RANLUX` (CERNLIB)

2.3 Parameter in steering files

`'updf-grid.dat'`	name of the grid file
`oneLoop = 0`	to select *all loop* CCFM or *one loop* DGLAP type evolution
`saturation = 0`	to select standard or saturated initial condition
`Ipdf = 60500`	LHApdf set name for collinear valence quark starting distribution
`Itarget = 2212`	hadron target ID (2212=proton)
`Iglu = 1`	for gluon only evolution
`Ipgg = 1`	parameter for P_{gg} splitting function
`ns = 1`	parameter for treatment of non-sudakov form factor
`ikincut = 2`	flag for consistency constraint
`Qg = 2.2`	starting value q_0 for perturbative evolution
`QCDlam = 0.20`	value for Λ_{qcd}
`A1,..., A6`	values for starting distribution; meaning depends on whether standard or saturation ansatz is used.

3 Example Program

```
      Program ccfm_uPDF

      Include 'SMallx.inc'
      Integer Iev
C--- event common block
      Integer NMXHEP,NEVHEP,NHEP,ISTHEP,IDHEP,JMOHEP,JDAHEP
      Double Precision PHEP,VHEP,EVWGT
C---Event weight
      COMMON/HEPWGT/EVWGT
      Integer nobran,ikincut
      Common/myvar/nobran,ikincut
      Double Precision Qbarmy,Qbar_min,Qbar_max
      Common/mglubran/Qbarmy,Qbar_min,Qbar_max
      Integer neve
      Common/myevt/neve
      Integer nloop
      Common/myloop/nloop
      Double Precision x3lmin,x3lmax,x3ldif
      Integer Nbp
      Parameter (Nbp=50)
      Double Precision X3(0:Nbp+1)
      Double Precision X3M(0:Nbp)
      Double Precision x3b(0:Nbp)
      Common/gridtt/x3m

      Integer ng_max
      Integer nrglu
      Common/mynrglu/nrglu
      Integer nmax,i,nx3,kev,ic
      Integer Ipgg,ns_sel
      Double Precision scal
      Common/Pggsel/Ipgg,ns_sel,scal
      Double Precision Qgmin
      Character *72 TXT
      CHARACTER   FILNAME*132,testNAME*132
      Common/gludatf/filname
      Double Precision Xnorm
      Common/smnorm/ Xnorm
      Logical pdflib,quark,gluon,photon,saturation
```

```
      Common /SMbran2/pdflib,quark,gluon,photon,saturation
      Integer Ioneloop,Itarget,Iglu,Isaturation
      Integer Ipdf
      Common/pdf/Ipdf
      Integer iparton
      Common /SMquark/iparton
      Character *15 char
      Double Precision BB
      Common /splitting/ BB
      Double precision ininorm(-6:6)
      Common/smininorm/ininorm

      Integer IRR
      Couble precision au
      Logical first
      Common/f2fit/au(50),first
*
      Read(5,*) filname
      Write(6,*) ' output file ',filname
      xnorm = 1.
      Read(5,101) TXT
      Read(txt,1005) char,Ioneloop
      Write(6,*) txt,char,Ioneloop
1005  format(a10,I8)
      Read(5,101) TXT
      Read(txt,1010) char,Isaturation
1010  format(a14,I8)
      Write(6,*) txt,char,Isaturation
      Read(5,101) TXT
      Read(txt,1006) char,Ipdf
      Write(6,*) txt,char,Ipdf
1006  format(a7,I8)
      Read(5,101) TXT
      Read(txt,1007) char,Itarget
      Write(6,*) txt,char,Itarget
1007  format(a10,I8)
      Read(5,101) TXT
      Read(txt,1008) char,Iglu
      Write(6,*) txt,char,Iglu
1008  format(a7,I8)
      Read(5,101) TXT
  101 Format(A72)
      Read(txt,1000) char,Ipgg
      Write(6,*) txt,char,Ipgg
1000  format(a7,I8)
      Read(5,101) TXT
      Read(txt,1001) char,ns_sel
      Write(6,*) txt,char,ns_sel
1001  format(a5,I8)
      Read(5,101) TXT
      Read(txt,1011) char,ikincut
      Write(6,*) txt,char,ikincut
1011  format(a10,I8)
      Read(5,101) TXT
      Read(txt,1002) char,Qg
      Write(6,*) txt,char,Qg
1002  format(a5,F16.8)
      Read(5,101) TXT
      Read(txt,1003) char,Qs
      Write(6,*) txt,char,Qs
1003  format(a5,F16.8)
      Read(5,101) TXT
      Read(txt,1018) char,QCDlam
      Write(6,*) txt,char,QCDlam
1018  format(a9,F16.8)
      Read(5,101) TXT
      Read(txt,1003) char,AU(1)
      Read(5,101) TXT
      Read(txt,1003) char,AU(2)
      Read(5,101) TXT
      Read(txt,1003) char,AU(3)
      Read(5,101) TXT
      Read(txt,1003) char,AU(4)
      Read(5,101) TXT
      Read(txt,1003) char,AU(5)
      Read(5,101) TXT
      Read(txt,1003) char,AU(6)

      If(iglu.ne.0) then
         gluon = .true.
```

```
          else
          gluon = .false.
        Endif
      If(Ioneloop.eq.1) then
          onel = .true.
          else
          onel=.false.
      Endif
      If(Isaturation.eq.1) then
          saturation = .true.
          else
          saturation=.false.
      Endif
      If(iglu.eq.0) then
          Read(50,101) TXT
          Read(txt,1009) char,Iparton
          Write(6,*) txt,char,Iparton
          Write(6,*) txt
1009  Format(a9,I8)
      Endif
      Close(50)

C---Initialize run
      Call SMinit
      neve = 0
      nmax =nev
      Xini = 0.
      Write(6,*) ' output file ',filname
      Write(6,*) ' selection Ipgg = ',Ipgg,' ns_sel = ',ns_sel
      Write(6,*) ' Qg = ',Qg,' Qs = ',Qs,' Xnorm = ',Xnorm
      Write(6,*) ' LHAPDFLIB for val quark Ipdf = ',Ipdf
      Write(6,*) ' Itarget = ',Itarget
      Write(6,*) ' BB = ',BB

      Qgmin = max(Qg-0.5d0,QCDlam)
      Qgmin = max(Qg,QCDlam)
      x3lmin = log(Qgmin)
      x3lmax = log(qmax)
      x3ldif = (x3lmax-x3lmin)/Real(Nbp)
      Do I=0,Nbp+1
          x3(I) = exp(x3lmin + x3ldif*Real(I))
      Enddo
      Do I=0,Nbp
          x3m(i) = (x3(i) + x3(i+1))/2.
          x3b(i) = x3(i+1) - x3(i)
      Enddo
      Nx3 = -1
C---Initialize analysis
      Xini=0.
      Xfin=0.
      Call updfgrid(1)
      Nx3 = 49
600  Nx3 = Nx3 + 1
      write(6,*) ' ng_max = ',ng_max,' at nx3-1 ',nx3-1
      IF(Nx3.gt.Nbp) Then
          write(6,*) ' evolve_tmd: Nx3 gt Npb -> Program stopped'
          stop
      Endif
      Qbarmy = x3m(Nx3)
      nloop = Nx3
      ng_max = 0

      Write(6,*) ' Qbar_my  ',Qbarmy

      Do Iparton=0,2
           Write(6,*) ' evolving parton ID: ',Iparton
           If(iparton.eq.0) then
                gluon=.true.
                else
               gluon=.false.
           Endif
          Xini = 0
          Do I=1,nev
C---Initialize event
              Call SMinfn

C---gluon branching process
              Call smbran
              Xini = Xini + x0
              If(ng_max.le.nrglu) ng_max=nrglu
```

```
              If(wt.NE.0.0) then
                  kev=i
                  if (kev.gt.0) ic=100000
                  if (mod(kev,ic).eq.0) write(6,*) ' event ',kev,nev,' loop P_max ',nloop,Qbarmy
              Endif
          Enddo
      Enddo
C---Terminate analysis
      call updfgrid(3)
      Stop
80  Write(6,*) ' steering file ccfm_updf not found '
      stop
      End
```

4 Program Installation

uPDFevolv follows the standard AUTOMAKE convention. To install the program, do the following

```
1) Get the source

tar xvfz uPDFevolv-XXXX.tar.gz
cd uPDFevolv-XXXX

2) Generate the Makefiles (do not use shared libraries)
./configure

3) Compile the binary
make

4) Install the executable
make install

4) The executable is in bin

run it with:
bin/updf_evolve < steer_gluon-JH-2013-set2

plot the result with:

bin/updfread
```

5 Acknowledgments

We are very grateful to Bryan Webber for careful reading of the manuscript and clarifying comments.

References

[1] J. Collins, *Foundations of perturbative QCD*, Vol. 32. Cambridge monographs on particle physics, nuclear physics and cosmology., 2011.

[2] S. M. Aybat and T. C. Rogers, Phys.Rev. **D83**, 114042 (2011). 1101.5057.

[3] M. Buffing, P. Mulders, and A. Mukherjee, Int.J.Mod.Phys.Conf.Ser. **25**, 1460003 (2014). 1309.2472.

[4] M. Buffing, A. Mukherjee, and P. Mulders, Phys.Rev. **D88**, 054027 (2013). 1306.5897.

[5] M. Buffing, A. Mukherjee, and P. Mulders, Phys.Rev. **D86**, 074030 (2012). 1207.3221.

[6] P. Mulders, Pramana **72**, 83 (2009). 0806.1134.

[7] S. Jadach and M. Skrzypek, Acta Phys.Polon. **B40**, 2071 (2009). 0905.1399.

[8] F. Hautmann, Acta Phys.Polon. **B40**, 2139 (2009).

[9] F. Hautmann, M. Hentschinski, and H. Jung (2012). 1205.6358.

[10] F. Hautmann and H. Jung, Nucl.Phys.Proc.Suppl. **184**, 64 (2008). 0712.0568.

[11] S. Catani, M. Ciafaloni, and F. Hautmann, Phys. Lett. **B242**, 97 (1990).

[12] J. C. Collins and R. K. Ellis, Nucl. Phys. **B360**, 3 (1991).

[13] F. Hautmann, H. Jung, and V. Pandis, AIP Conf.Proc. **1350**, 263 (2011). 1011.6157.

[14] S. Catani, M. Ciafaloni, and F. Hautmann, Nucl. Phys. **B366**, 135 (1991).

[15] S. Catani, M. Ciafaloni, and F. Hautmann, Phys. Lett. **B307**, 147 (1993).

[16] L. Lipatov, Phys.Rept. **286**, 131 (1997). hep-ph/9610276.

[17] V. S. Fadin, E. Kuraev, and L. Lipatov, Phys.Lett. **B60**, 50 (1975).

[18] I. I. Balitsky and L. N. Lipatov, Sov. J. Nucl. Phys. **28**, 822 (1978).

[19] V. N. Gribov and L. N. Lipatov, Sov. J. Nucl. Phys. **15**, 438 (1972).

[20] G. Altarelli and G. Parisi, Nucl. Phys. **B126**, 298 (1977).

[21] Y. L. Dokshitzer, Sov. Phys. JETP **46**, 641 (1977).

[22] M. Ciafaloni, Nucl. Phys. **B296**, 49 (1988).

[23] S. Catani, F. Fiorani, and G. Marchesini, Nucl. Phys. **B336**, 18 (1990).

[24] G. Marchesini, Nucl. Phys. **B445**, 49 (1995). hep-ph/9412327.

[25] S. Catani and F. Hautmann, Nucl. Phys. **B427**, 475 (1994). hep-ph/9405388.

[26] S. Catani and F. Hautmann, Phys.Lett. **B315**, 157 (1993).

[27] H. Jung and G. P. Salam, Eur. Phys. J. **C19**, 351 (2001). hep-ph/0012143.

[28] H. Jung, S. Baranov, M. Deak, A. Grebenyuk, F. Hautmann, *et al.*, Eur.Phys.J. **C70**, 1237 (2010). 1008.0152.

[29] M. Hansson and H. Jung (2003). hep-ph/0309009.

[30] B. Andersson, G. Gustafson, and J. Samuelsson, Nucl.Phys. **B467**, 443 (1996). Revised version.

[31] J. Kwiecinski, A. D. Martin, and P. Sutton, Z.Phys. **C71**, 585 (1996). hep-ph/9602320.

[32] M. Deak, F. Hautmann, H. Jung, and K. Kutak, *Forward-Central Jet Correlations at the Large Hadron Collider*, 2010. 1012.6037.

[33] G. Marchesini and B. Webber, Nucl. Phys. **B 349**, 617 (1991).

[34] G. Marchesini and B. Webber, Nucl. Phys. **B 386**, 215 (1992).

[35] H. Jung and F. Hautmann (2012). 1206.1796.

[36] F. Hautmann and H. Jung, Nuclear Physics B **883**, 1 (2014). 1312.7875.

[37] K. Golec-Biernat and M. Wusthoff, Phys. Rev. **D 60**, 114023 (1999). hep-ph/9903358.

[38] A. Grinyuk, A. Lipatov, G. Lykasov, and N. Zotov, Phys.Rev. **D87**, 074017 (2013). 1301.4545.

[39] H. Jung et al. , *TMDlib and TMDplotter*. DESY-14-059.