The CCFM uPDF evolution uPDFevolv Version 1.0.00

F. Hautmann, H. Jung, S. Taheri Monfared

1 Department of Physics and Astronomy, University of Sussex, Brighton BN1 9QH, UK
2 Rutherford Appleton Laboratory, Chilton OX11 0QX, UK
3 Department of Theoretical Physics, University of Oxford, Oxford OX1 3NP, UK
4 DESY, Hamburg, Germany
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.

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 [1–10]. TMD factorization has been proven recently [1] 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) [11–13].

In the framework of high-energy factorization [11,14] the deep-inelastic scattering cross section can be written as a convolution in both longitudinal and transverse momenta of the TMD parton density function $A(x, k_t, \mu)$ with off-shell partonic matrix elements, as follows

$$\sigma_j(x, Q^2) = \int_1^x dz \int d^2k_t \hat{\sigma}_j(x, Q^2, z, k_t) A(z, k_t, p),$$

(1)

with the DIS cross sections $\sigma_j (j = 2, L)$ 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. (1) are $k_t$-dependent and the evolution of the transverse momentum dependent gluon density $A$ is obtained by combining the resummation of small-x logarithmic contributions [15–17] with medium-x and large-x contributions to parton splitting [18–20] according to the CCFM evolution equation [21–23].

The factorization formula (1) allows one to sum 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 $\mu$ and on the factorization scheme [24,25].

The CCFM evolution equation [21–23] 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 $A(x, k_t, p)$, depending on $x$, $k_t$ and the evolution variable $p$, is
\[ A(x, k_t, p) = A_0(x, k_t, p) + \int \frac{dz}{z} \int \frac{dq^2}{q^2} \Theta(p - q) \]
\[ \times \Delta_s(p, q) P(z, q, k_t) A\left(\frac{x}{z}, k_t + (1-z)q, q\right), \]
\[ (2) \]
where \( z \) is the longitudinal momentum fraction, \( q \) is the angular variable and the \( \Theta \) function specifies the ordering condition of the evolution [26].

The first term in the right hand side of Eq. (2) is the contribution of the non-resolvable branchings between the starting scale \( q_0 \) and the evolution scale \( p \), and is given by
\[ A_0(x, k_t, p) = A_0(x, k_t, q_0) \Delta_s(p, q_0), \]
\[ (3) \]
where \( \Delta_s \) is the Sudakov form factor, and \( A_0(x, k_t, q_0) \) is the starting distribution at scale \( q_0 \). The integral term in the right hand side of Eq. (2) gives the \( k_t \)-dependent branchings in terms of the Sudakov form factor \( \Delta_s \) and unintegrated splitting function \( P \).

The Sudakov form factor \( \Delta_s \) is given by
\[ \Delta_s(p, q_0) = \exp\left(-\int_{q_0^2}^{p^2} \frac{dq^2}{q^2} \int_0^{1-q_0/q} \frac{dz}{z} \frac{\tilde{\alpha}_s(q^2(1-z)^2)}{1-z}\right), \]
\[ (4) \]
with \( \tilde{\alpha}_s = C_A \alpha_s / \pi = 3.3 \alpha_s / \pi \).

For application in Monte Carlo event generators, like CASCADe [27,28], it is of advantage to write the CCFM evolution equation in differential form:
\[ \frac{d}{dp^2} x A(x, k_t, p) \]
\[ = \int dz \frac{dP(z, p/z, k_t)}{2\pi \Delta_s(p, q_0)} x' A(x', k'_t, p/z), \]
\[ (5) \]
where the splitting variable \( x' \) is given by \( x' = x/z, k'_t = q_t(1-z)/z + k_t, \) and \( \phi \) 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 \( P_{gg}(z_i, q_i, k_{ii}) \) for branching \( i \) is given by [29] (set by \( \text{Ipgg}=1, \text{ns}=1 \) in uPDFevol):
\[ P_{gg}(z_i, q_i, k_{ii}) = \tilde{\alpha}_s(q_i^2(1-z_i)^2) \left( \frac{1}{1-z_i} - 1 + \frac{z_i(1-z_i)}{2} \right) \Delta_{ns}(z_i, q_i^2, k_{ii}^2) \]
\[ + \tilde{\alpha}_s(k_{ii}^2) \left( \frac{1}{z_i} - 1 + \frac{z_i(1-z_i)}{2} \right) \Delta_{ns}(z_i, q_i^2, k_{ii}^2), \]
\[ (6) \]
where \( \Delta_{ns} \) is the non-Sudakov form factor defined by
\[ \log \Delta_{ns} = -\tilde{\alpha}_s(k_{ii}^2) \int_0^1 dz' \left( \frac{1}{z' - 1} + \frac{z'(1-z')}{2} \right) \int \frac{dq^2}{q^2} \Theta(k_{ii} - q) \Theta(q - z'q_{ii}). \]
\[ (7) \]

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 \( \text{Ipgg}=0, \text{ns}=0 \) in uPDFevol):
\[ P_{gg}(z, q, k_t) = \frac{\tilde{\alpha}_s(q^2)}{1-z} + \frac{\tilde{\alpha}_s(k_t^2)}{z} \Delta_{ns}(z, q^2, k_t) \]
\[ (8) \]
with
\[ \log \Delta_{ns} = -\tilde{\alpha}_s(k_{ii}^2) \int_0^1 dz' \int \frac{dq^2}{q^2} \Theta(k_{ii} - q) \Theta(q - z'q_{ii}). \]
\[ (9) \]

Another uses \( \alpha_s(q^2) \) also for the small \( z \) part (set by \( \text{Ipgg}=2, \text{ns}=2 \) in uPDFevol):
\[ P_{gg}(z, q, k_t) = \frac{\tilde{\alpha}_s(q^2)}{1-z} + \frac{\tilde{\alpha}_s(k_t^2)}{z} \Delta_{ns}(z, q^2, k_t) \]
\[ (10) \]
with
\[ \log \Delta_{ns} = -\int_0^1 dz' \int \frac{dq^2}{q^2} \alpha_s(q^2) \Theta(k_{ii} - q) \Theta(q - z'q_{ii}). \]
\[ (11) \]

In general a four-momentum \( a \) can be written in light-cone variables as \( a = (a^+, a^-, a_T) \) 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 \( a_T^2 = 2a^+a^- - a_T^2 \) 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. 1), which has been implemented in different forms (set by \( \text{Ikincut}=1,2,3 \).
The term \( xQ \) denoted level according to the branching evolution at the transverse-momentum distribution from sea-quarks needs to be included. We include for a complete description of the final states also the contribution from sea-quarks evolution according to 

\[
q_t^2 < \frac{k_t^2}{z} \quad \text{LDC [21, 30]}
\]

\[
q_t^2 < (1 - z)k_t^2 \quad [31]
\]

\[
k_t^2 < \frac{k_t^2}{z} \quad \text{BFKL [31]}
\]

### 1.1.3 Sea quarks

Using the method of [32, 33] valence quarks are included in the branching evolution at the transverse-momentum dependent level according to

\[
xQ_v(x, k_t, p) = xQ_v(0, k_t, p) + \int \frac{dz}{z} \int \frac{dq^2}{q^2} \theta(p - zq) \times \Delta_s(p, zq) P_{qq}(z, q, k_t) xQ_v \left( \frac{z}{z}, k_t + (1 - z)q, q \right),
\]

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

\[
P_{qq}(z, q, k_t) = \frac{C_F}{2\pi} \alpha_s(q^2(1 - z)^2)^{\frac{1 + z}{1 - z}}.
\]

In Eqs. (15), (16) the non-Sudakov form factor is not included, unlike the CCFM kernel given in the appendix B of [22], because we only associate this factor with \( 1/z \) terms. The term \( xQ_v \) in Eq. (15) is the contribution of the non-resolvable branchings between starting scale \( q_0 \) and evolution scale \( p \), given by

\[
xQ_v(0, k_t, p) = xQ_v(0, k_t, q_0) \Delta_s(p, q_0),
\]

where \( \Delta_s \) is the Sudakov form factor.

### 1.1.4 Monte Carlo solution of the CCFM evolution equations

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

\[
A(x, k_t, p) = A_0(x, k_t, p) + \int \frac{dz}{z} \int \frac{dq^2}{q^2} \theta(p - zq) \Delta_s(p, zq) P_{gg}(z, q, k_t) \times A \left( \frac{z}{z}, k_t + (1 - z)q, q \right)
\]

\[
+ \int \frac{dz}{z} \int \frac{dq^2}{q^2} \theta(p - zq) \Delta_s(p, zq) P_{pq}(z, q, k_t) \times S \left( \frac{z}{z}, k_t + (1 - z)q, q \right)
\]

where \( S \) is similar to Eqs. (3), (17).

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

\[
A(x, k_t, p) = A_0(x, k_t, p)
\]

\[
+ \int \frac{dz}{z} \int \frac{dq^2}{q^2} \theta(p - zq) \Delta_s(p, zq) P_{gg}(z, q, k_t)
\]

\[
+ \int \frac{dz}{z} \int \frac{dq^2}{q^2} \theta(p - zq) \Delta_s(p, zq) P_{pq}(z, q, k_t)
\]

\[
+ S \left( \frac{z}{z}, k_t + (1 - z)q, q \right).
\]

### 1.1.4 Monte Carlo solution of the CCFM evolution equations

The evolution equations (23), (24) are integral equations of the Fredholm type

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

where \( \lambda \) is the integral kernel and \( K(x, y) \) is the kernel of the Fredholm type integral equation.
and can be solved by iteration as a Neumann series

\[ f_1(x) = f_0(x) + \lambda \int_a^b K(x, y) f_0(y) dy \]

\[ f_2(x) = f_0(x) + \lambda \int_a^b K(x, y_1) f_0(y_1) dy_1 + \lambda^2 \int_a^b \int_a^b K(x, y_1) K(y_1, y_2) f_0(y_2) dy_2 dy_1 + \ldots \]  

(25)

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

\[ f(x) = \lim_{n \to \infty} \sum_{i=0}^n f_i(x). \]  

(26)

Applying this to the evolution equations Eqs. (23), (24), we identify \( f_0 \) with the first term in Eq. (24), where we use for simplicity here and in the following \( \Delta_s(p) = \Delta_s(p_0) \):

\[ A_0(x, k_t, p) = A_0(x, k_t) \Delta_s(p). \]  

(27)

The first iteration involves one branching:

\[ A_1(x, k_t, p) = A_0(x, k_t) \Delta_s(p) \]

\[ + \int_x^p \frac{dz'}{z'} \int_{q_0}^p \frac{dq'^2}{q'^2} \Theta(p - z' q') \]

\[ \times \Delta_s(p) \Delta_s(\Delta q') \tilde{P}(z') A_0(x/z', k_t', q'). \]  

(28)

The second iteration involves two branchings,

\[ A_2(x, k_t, p) = A_0(x, k_t) \Delta(p) \]

\[ + \int_x^p \frac{dz'}{z'} \int_{q_0}^p \frac{dq'^2}{q'^2} \Theta(p - z' q') \]

\[ \times \Delta_s(p) \Delta_s(\Delta q') \tilde{P}(z') A_0(x/z', k_t', q') \]

\[ + \left( \frac{\alpha_s}{2\pi} \right)^2 \int_x^p \int_{q_0}^p \left( \int_x^{z'} \frac{dz''}{z''} \int_{q_0}^{z''} \frac{dq''}{q''} \Theta(p - z'' q'') \right) \]

\[ \times \Delta_s(p) \Delta_s(\Delta q') \tilde{P}(z'') A_0(x/z'', k_t'', q''). \]  

(29)

In a Monte Carlo (MC) solution [34,35] 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. 2). 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 \( q' \):

\[ R = \Delta_s(q', q_0). \]  

(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 \( \Delta_s \) and using

\[ \frac{\partial}{\partial q'^2} \Delta_s(p, z q') = \frac{\partial}{\partial q'^2} \Delta_s(p) \]

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

we can write the first iteration of the evolution equation as

\[ A_1(x, k_t, p) = A_0(x, k_t, p) \]

\[ + \int_x^p \frac{dz'}{z'} \int_{q_0}^p d\Delta_s(p, z' q') \tilde{P}(z') \]

\[ \times A_0(x/z', k_t', q'). \]  

(31)

The integrals can be solved by a Monte Carlo method [36]: \( z \) is generated from

\[ \int_{z_{\text{min}}}^{\tilde{z}} dz' \tilde{P}(z') = R_1 \int_{z_{\text{min}}}^{z_{\text{max}}} dz' \tilde{P}(z'), \]  

(32)

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

\[ R_2 = \int_{-\infty}^{x} f(x') dx' = F(x) \]

\[ = \int_{q_0}^{p} \frac{\partial}{\partial q'^2} \left( \Delta_s(p, z q') \right) dq'^2 \]

\[ = \Delta_s(p, z q'). \]  

(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

$$x A_1(x, k_t, p) = x A_0(x, k_t) \Delta_1(p) + \sum_i \tilde{P} (z'_i) x'_i A_0(x'_i, k'_t, q'_i) \left[ \int_{z''_{\text{max}}}^{z_{\text{max}}} dz \tilde{P}(z) \right]^{-1}, \quad (34)$$

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 $p$

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

1.2 Computational techniques: CCFM grid

When using the CCFM evolution in a fit program to determine the starting distribution $A_0(x)$, a full MC solution [34,35] is no longer suitable, since it is time consuming and suffers from numerical fluctuations. Instead a convolution method introduced in [37,38] is used. The kernel $\tilde{A}(x'', k_t, p)$ is determined once from the Monte Carlo solution of the CCFM evolution equation, and then folded with the non-perturbative starting distribution $A_0(x)$.

$$x A(x, k_t, p) = x \int dx' \int dx'' A_0(x') \tilde{A}(x'', k_t, p) \delta(x'' - x) = \int dx' A_0(x') \frac{x}{x'} \tilde{A} \left( \frac{x}{x'}, k_t, p \right). \quad (36)$$

The kernel $\tilde{A}$ 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 ($k_t > k_{t\text{cut}}$), without contribution from the soft (non-perturbative) region.

1.3 Functional forms for starting distribution

1.3.1 Standard parametrisation

For the starting distribution $A_0$, at the starting scale $q_0$, the following form is used:

$$x A_0(x, k_t, q_0) = A_1 x^{-A_2} \cdot (1 - x)^{A_3} (1 - A_4 x + A_5 \sqrt{x} + A_6 x^2) \exp[-k_t^2/\sigma^2]. \quad (37)$$

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

Valence quarks are treated using the method of [32,33,38] with starting distributions at scale $q_0$ parameterized using standard collinear pdfs (set by Tpdf in upDFevolv) as

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

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 \( A_0 \) at scale \( q_0 \) is available, following the parameterisation of the saturation model by Eq. (18) of [39].

\[
x A_{\text{sat}} = \frac{1}{4\pi^2} \frac{3\sigma_0}{\alpha_s} R_0^2(x) k_t^2 \exp\left(-R_0^2(x) k_t^2\right),
\]

(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 [40]) is applied.

1.4 Plotting TMDs

A simple plot program is included in the package. For a graphical web interface use TMDplotter [41].

1.5 Application

The evolution of the TMD gluon density has been used to perform fits to the DIS precision data [42,43], as described in detail in [38].

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: https://updfdevolv.hepforge.org/

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

Utility routines:

- evolvetmd
- updforead
- gadap
- gadap2
- divdif
- ranlux

Main routine to perform CCFM evolution

Example program

Program ccfm_uPDF

Include 'SMallx.inc'

Integer lev

C--- common block

Integer jmohep,jevhep,imhep,icbhep,iwkhep,shep,ihhep/ihhhep,/dhhep

Double Precision Jhep,Jhep,jhep,ihhhep,iwkhep,shep,ihhep/ihhhep,/dhhep

C---event weight

COMONS/MN7/DMN7

Integer nobran,ikincut

Common/nobran,ikincut

Double Precision Qgmin, Qbar_min, Qbar_max

Common/qgmin/qbar_min/qbar_max

Integer ns

Common/ns

Integer ns

Common/loop/nloop

Double Precision x1min, x1max, x1dif

Integer ns

Double Precision X3[0:Nbp+1]

Double Precision X3[0:Nbp+1]

Double Precision X3[0:Nbp+1]

Double Precision X3[0:Nbp+1]

Double Precision S3[0:Nbp+1]

Integer ng_max

Integer ng_max

Integer ng_max

Common/loop/nloop

Double Precision x1min, x1max, x1dif

Integer ns

Double Precision X3[0:Nbp+1]

Double Precision X3[0:Nbp+1]

Double Precision X3[0:Nbp+1]

Double Precision X3[0:Nbp+1]

Double Precision S3[0:Nbp+1]

Character *3,J TST

Character *3,J TST

Character *3,J TST

Character *3,J TST

Double Precision Qmin

Double Precision Qmin

Double Precision Qmin

Double Precision Qmin

Double Precision Qmin

Common/pdflib,quark,gluon,photon,saturation

Common/pdflib,quark,gluon,photon,saturation

Common/pdflib,quark,gluon,photon,saturation

Common/pdflib,quark,gluon,photon,saturation

Common/pdflib,quark,gluon,photon,saturation

Integer jmohep,jevhep,imhep,icbhep,shep,ihhep/ihhhep,/dhhep

2.3 Parameter in steering files

- 'updf - grid.dat'
  - oneLoop = 0
  - saturation = 0
  - Tpdf = 60500
  - Itarget = 2212
  - Ipgg = 1
  - ns = 1
  - ikincut = 2
  - Qg = 2.2
  - QCDlam = 0.20
  - A1,...,A6

name of the grid file

to select all loop CCFM or one loop DGLAP type evolution

to select standard or saturated initial condition

ehadron target ID (2212 = proton)

for gluon only evolution parameter for \( P_{gg} \) splitting function

parameter for treatment of non-sudakov form factor

flag for consistency constraint

starting value \( q_0 \) for perturbative evolution

value for \( \Lambda_{QCD} \)

values for starting distribution; meaning depends on whether standard or saturation ansatz is used.
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
Write(6,*) txt,char,Isaturation
1010 format(a14,I8)
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
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)
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
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-.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
Endif
C---gluon branching process
Call embtran
Xini = Xini + x0
If(nmax.le.nrglu) nmax=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
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) Set 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_evolv < steer_gluon-JH-2013-set2
plot the result with:
bin/updfread
Acknowledgments We are very grateful to Bryan Webber for careful reading of the manuscript and clarifying comments.

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

Funded by SCOAP3 / License Version CC BY 4.0.

References

1. J. Collins, Foundations of Perturbative QCD, vol. 32. Cambridge Monographs on Particle Physics, Nuclear Physics and Cosmology (2011)
2. S.M. Aybat, T.C. Rogers, Phys. Rev. D 83, 114042 (2011). arXiv:1101.5057
3. M. Buffing, P. Mulders, A. Mukherjee, Int. J. Mod. Phys. Confl. Ser. 25, 1460003 (2014). arXiv:1309.2472
4. M. Buffing, A. Mukherjee, P. Mulders, Phys. Rev. D 88, 054027 (2013). arXiv:1306.5897
5. M. Buffing, A. Mukherjee, P. Mulders, Phys. Rev. D 86, 074030 (2012). arXiv:1207.3221
6. P. Mulders, Pramana 72, 83 (2009). arXiv:0806.1134
7. S. Jadach, M. Skrzypek, Acta Phys. Pol. B 40, 2071 (2009). arXiv:0905.1399
8. F. Hautmann, Acta Phys. Pol. B 40, 2139 (2009)
9. F. Hautmann, M. Hentschinski, H. Jung (2012). arXiv:1205.6358
10. F. Hautmann, H. Jung, Nucl. Phys. Proc. Suppl. 184, 64 (2008). arXiv:0712.0568
11. S. Catani, M. Ciafaloni, F. Hautmann, Nucl. Phys. B 366, 135 (1991)
12. S. Catani, M. Ciafaloni, F. Hautmann, Phys. Lett. B 307, 147 (1993)
13. F. Hautmann, Phys. Lett. B 535, 159 (2002). hep-ph/0203140
14. S. Catani, M. Ciafaloni, F. Hautmann, Phys. Lett. B 242, 97 (1990)
15. L. Lipatov, Phys. Rep. 286, 131 (1997). hep-ph/9610276
16. V.S. Fadin, E. Kuraev, L. Lipatov, Phys. Lett. B 60, 50 (1975)
17. L.I. Balitsky, L.N. Lipatov, Sov. J. Nucl. Phys. 28, 822 (1978)
18. V.N. Gribov, L.N. Lipatov, Sov. J. Nucl. Phys. 15, 438 (1972)
19. G. Altarelli, G. Parisi, Nucl. Phys. B 126, 298 (1977)
20. Y.L. Dokshitzer, Sov. Phys. JETP 46, 641 (1977)
21. M. Ciafaloni, Nucl. Phys. B 296, 49 (1988)
22. S. Catani, F. Fiorani, G. Marchesini, Nucl. Phys. B 336, 18 (1990)
23. G. Marchesini, Nucl. Phys. B 445, 49 (1995). hep-ph/9412327
24. S. Catani, F. Hautmann, Nucl. Phys. B 427, 475 (1994). hep-ph/9405388
25. S. Catani, F. Hautmann, Phys. Lett. B 315, 157 (1993)
26. F. Hautmann, H. Jung, JHEP 10, 113 (2008). arXiv:0805.1049
27. H. Jung, G.P. Salam, Eur. Phys. J. C 19, 351 (2001). hep-ph/0012143
28. H. Jung, S. Baranov, M. Deak, A. Grebenyuk, F. Hautmann et al., Eur. Phys. J. C 70, 1237 (2010). arXiv:1008.0152
29. M. Hansson, H. Jung (2003). arXiv:hep-ph/0309009
30. B. Andersson, G. Gustafson, J. Samuelsson, Nucl. Phys. B 467, 443 (1996) (revised version)
31. J. Kwiecinski, A.D. Martin, P. Sutton, Z. Phys. C 71, 585 (1996). hep-ph/9602320
32. M. Deak, F. Hautmann, H. Jung, K. Kutak, Forward-Central Jet Correlations at the Large Hadron Collider (2010). arXiv:1012.6037
33. M. Deak, F. Hautmann, H. Jung, K. Kutak, Eur. Phys. J. C 72, 1982 (2012). arXiv:1112.6354
34. G. Marchesini, B. Webber, Nucl. Phys. B 349, 617 (1991)
35. G. Marchesini, B. Webber, Nucl. Phys. B 386, 215 (1992)
36. F. James, Rep. Prog. Phys. 43, 1145 (1980)
37. H. Jung, F. Hautmann (2012). arXiv:1206.1796
38. F. Hautmann, H. Jung, Nucl. Phys. B 883, 1 (2014). arXiv:1312.7875
39. K. Golec-Biernat, M. Wusthoff, Phys. Rev. D 60, 114023 (1999). hep-ph/9903358
40. A. Grinyuk, A. Lipatov, G. Lykasov, N. Zotov, Phys. Rev. D 87, 074017 (2013). arXiv:1301.4545
41. H. Jung et al., TMDlib and TMDplotter. DESY-14-059
42. H1 and ZEUS Collaboration, F. Aaron et al., JHEP 1001, 109 (2010). arXiv:0911.0884
43. H1 and ZEUS Collaboration, H. Abramowicz et al., Eur. Phys. J. C 73, 2311 (2013). arXiv:1211.1182