APFEL v2.6.1: A PDF Evolution Library with QED corrections

Valerio Bertone[1,2], Stefano Carrazza[2] and Juan Rojo[1]

[1] *Rudolf Peierls Centre for Theoretical Physics,*
1 Keble Road, University of Oxford, OX1 3NP, Oxford, UK
[2] *PH Department, TH Unit, CERN, CH-1211 Geneva 23, Switzerland*

Abstract

In this document we present the user manual for the APFEL library. Written in FORTRAN 77, all the functionalities can also be accessed via the C/C++ and Python interfaces. For simplicity, we will restrict ourselves to the description of the C/C++ interface, but the usage of the FORTRAN 77 and Python interfaces is very similar and examples of their use are provided in the examples folder of the APFEL source code. First of all, we will discuss how to install APFEL and how to execute the basic example programs. After that, we will list the various customization options that can be accessed by the user for both the PDF evolution and the DIS structure functions modules.

Contents

1 Installation **1**

2 The PDF evolution module **2**
2.1 Customization of the PDF evolution 4
2.1.1 Setting functions 5

3 The DIS module **18**

1 Installation

The APFEL library is available from its HepForge website:

http://apfel.hepforge.org/

and from the GitHub webpage:

https://github.com/scarrazza/apfel

It can also be accessed directly from the git repository. The last development version can be downloaded by giving:

```
1 git clone https://github.com/scarrazza/apfel.git
```

For the tagged versions one can use the git tag commands:

```
git tag -l
git checkout tags/tag_name
```

to switch to any of the past releases. We strongly recommend to use the latest stables release.

The installation of the APFEL library can be easily done following the standard autotools sequence:

```
./configure
make
make install
```

which automatically installs APFEL in /usr/local/. Note that by default the APFEL library requires an installation of the LHAPDF PDF library[1]. However, an LHAPDF-less installation is also supported by giving:

```
./configure --disable-lhapdf
```

To use a different installation path, one simply needs to use the option:

```
./configure --prefix=/path/to/the/installation/folder
```

In this case, the APFEL installation path should be included to the environmental variable LD_LIBRARY_PATH. This can be done adding to the local .bashrc file (or .profile file on Mac) the string:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/to/the/installation/↩
    folder/lib
```

Once APFEL has been properly compiled and installed, the configuration script apfel-config should automatically be present. Such script is useful to determine the compiler flags. Type:

```
apfel-config --help
```

in a shell to see all the possible options. Particularly useful is the --list-funcs flag that lists all the functions available in APFEL along with a short explanation. In addition, also the shell script apfel is provided which starts an interactive console session of APFEL providing a prompt tool to use the library without coding.

In the following we will list and illustrate all the functionalities of APFEL and explaining how they can be accessed by the user. The most recent version of APFEL provide also an additional module to compute DIS (and SIA) structure functions in different mass scheme. Such module is dependent of the original PDF evolution module as it inherits from it many of the setting functions that we will describe in the next section. We will start describing

[1]The current release of APFEL assumes that LHAPDF version 6 has been previously installed as version 5 is no longer supported.

the functions of the PDF evolution module and we will devote the following section to a thourough description of the DIS module.

2 The PDF evolution module

The basic usage of the PDF evolution module of `APFEL` requires only two steps to have the complete set of evolved PDFs. The first step is the initialization of `APFEL` through the call of the following function:

```
APFEL::InitializeAPFEL();
```

This will precompute all the needed evolution operators that enter the discretized DGLAP equation. Let us recall that once the general settings of the evolution have been defined (perturbative order, heavy quark masses, reference value of α_s, and so on), the initialization needs to be performed only once, irrespective of the scales that are used in the PDF evolution. The second step consists in performing the actual PDF evolution between the initial scale `Q0` and the final scale `Q` (in GeV). This can be achieved using the function:

```
APFEL::EvolveAPFEL(Q0,Q);
```

Calling this function `APFEL` solves the discretized DGLAP equations using the evolution operators precomputed in the initialization step.

Now the user can access the evolved PDFs at the scale `Q` via the use of the functions:

```
APFEL::xPDF(i,x);
APFEL::xgamma(x);
```

where the real variable `x` is the desired value of Bjorken-x while the integer variable `i` in the function `xPDF`, which runs from -6 to 6, corresponds to the quark flavor index according to the following convention: Note that in `APFEL` we have explicitly separated

`i` :	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
`xPDF` :	$\bar{t}$	$\bar{b}$	$\bar{c}$	$\bar{s}$	$\bar{u}$	$\bar{d}$	g	d	u	s	c	b	t

the access to the quark and gluon PDFs (via `xPDF`) and from that to the photon PDF (via `xgamma`). Note also that the functions `xPDF` and `xgamma` return x times the PDFs (*i.e.* the momentum fractions).

The basic information given above is enough to write a simple and yet complete program that performs PDF evolution using `APFEL`. As an illustration, a `C/C++` program that computes and tabulates PDFs to be compared with the Les Houches PDF benchmark evolution tables would be the following:

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include "APFEL/APFEL.h"
```

```
5  using namespace std;
6
7  int main()
8  {
9    // Define grid in x
10   double xlha[11] = {1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2,
11       1e-1, 3e-1, 5e-1, 7e-1, 9e-1};
12
13   // Precomputes evolution operators on the grid
14   APFEL::InitializeAPFEL();
15
16   // Perform evolution
17   double Q0 = sqrt(2);
18   double Q  = sqrt(10000);
19   APFEL::EvolveAPFEL(Q0,Q);
20
21   cout << scientific << setprecision(5) << endl;
22   cout << "   x    "
23        << setw(11) << "   u-ubar    "
24        << setw(11) << "   d-dbar    "
25        << setw(11) << " 2(ubr+dbr)  "
26        << setw(11) << "   c+cbar    "
27        << setw(11) << "   gluon     "
28        << setw(11) << "   photon    " << endl;
29
30   cout << scientific;
31   // Tabulate PDFs for the LHA x values
32   for (int i = 0; i < 11; i++)
33     cout << xlha[i] << "\t"
34     << APFEL::xPDF(2,xlha[i]) - APFEL::xPDF(-2,xlha[i]) << "\t"
35     << APFEL::xPDF(1,xlha[i]) - APFEL::xPDF(-1,xlha[i]) << "\t"
36     << 2*(APFEL::xPDF(-1,xlha[i]) + APFEL::xPDF(-2,xlha[i])) << "\t"
37     << APFEL::xPDF(4,xlha[i]) + APFEL::xPDF(-4,xlha[i]) << "\t"
38     << APFEL::xPDF(0,xlha[i]) << "\t"
39     << APFEL::xgamma(xlha[i]) << "\t"
40     << endl;
41
42   return 0;
43 }
```

It should be noticed that this example code uses the default settings of `APFEL` for the evolution parameters such as: initial scale PDFs, perturbative order, heavy quark masses, values of the couplings, etc. In the following we will discuss how the user can customize the settings for the PDF evolution in `APFEL`.

2.1 Customization of the PDF evolution

The customization of the PDF evolution with `APFEL` can be achieved using a number of dedicated functions, to be called before the initialization stage, that is before calling `InitializeAPFEL`[2]. We will subdivide the available functions into three cathegories:

[2]This is not entirely precise because there is a number of customization functions that are effective only at the evolution level and thus can be called also after `InitializeAPFEL` but before `EvolveAPFEL`. We will discuss this feature case by case when going through all functions available in `APFEL`.

- the *setting functions* which provide the real costumization tools. These functions allow the user to change the way how the initialization and the output (see next items) functions behave.
- The *initialization functions* which are resposible to perform the "main" operations, like initializing the evolution factor and evolving PDFs.
- The *output functions* which finally return the result of a given set of setting and initialization functions.

In the following we will list and comment all the functions belonging to each of the cathegories above. Finally, we remind the reader that running the configuration script `apfel-config` with the `--list-funcs` flag will list all the functions available in `APFEL` along with a short explanation.

2.1.1 Setting functions

Bofore going through the various setting functions, the user should be aware of the fact that `APFEL` has a set of default settings that are used if the user does not intervene to change any of them. This is why the example described above needs only a very limited number of steps. However, while each time that `APFEL` is run a banner with a list of the main settings is displayed, it is usefull to report here the default settings of `APFEL`. We will first go through all the setting functions and only at the end we will report the default settings so that the meaning of all of them should be clear to the reader who went through this section.

Bofore proceeding with the descitpion of the setting functions of the evolution module, it is useful to notice that in the following we will use the identifiers `int`, `double`, `bool` and `string` to specify the type of entry expected by each of the fuctions described below.

```
1  APFEL::SetPerturbativeOrder(int pto);
```

This function sets the perturbative order of the PDF evolution to `pto`. The integer `pto` can take the values 0, 1 ot 2 according to whether the PDF evolution is performed at $\mathcal{O}(\alpha_s)$, $\mathcal{O}(\alpha_s^2)$, or $\mathcal{O}(\alpha_s^3)$, that is LO, NLO and NNLO, respectively. This function also sets the perturbative order of the evolution of the couplings α_s and α and possibly of the heavy quark masses. The default for `pto` is 2.

```
1  APFEL::SetTheory(string theory);
```

This function sets the theory to be used in the evolution. The alternatives for `theory` are:

- `"QCD"`: the PDF evolution is done solving the pure QCD DGLAP equations,
- `"QED"`: the PDF evolution is done solving the pure QED DGLAP equations,
- `"QUniD"`: the PDF evolution is done solving the coupled QCD+QED DGLAP equations as explained above.

There are more options available that access more "exotic" (and obsolete) solutions of the coupled QCD+QED DGLAP equations. They are:

- `"QCEDP"` for QCD+QED in parallel,
- `"QCEDS"` for QCD+QED in series,
- `"QECDP"` for QED+QCD in parallel,
- `"QECDS"` for QED+QCD in series,
- `"QavDP"` for the averaged solution in parallel,
- `"QavDS"` for the averaged solution in series,

and they refer to different combinations of the separate QCD and QED evolutions. The reader can refer to the original `APFEL` publication for a detailed explanation. However, the use of these solution is discouraged unless the user is well aware of their meaning. The default is `"QCD"`.

```
1  APFEL::SetVFNS();
```

This function sets the Variable-Flavour Number Scheme (VFNS) for the PDF, α_s, and α[3] evolution. In practice this means that, if any heavy quark threshold is encountered during the evolution, the solutions of the DGLAP equation below and above the threshold itself will be properly matched. This option is used as a default.

```
1  APFEL::SetFFNS(int nfl);
```

This function, as opposed to `SetVFNS`, sets the Fixed-Flavour Number Scheme (FFNS) with `nfl` active flavours for the PDF, α_s and α (and $\overline{\rm MS}$) evolution. This function forces the evolution to be done with `nfl` active flavours in any evolution range. The allowed values are `nfl` = 3, 4, 5 and 6.

```
1  APFEL::SetAlphaQCDRef(double alpharef, double Qref);
```

This function sets the reference values of the strong coupling α_s at the scale `Qref` in GeV to `alpharef`. The default is `alpharef` = 0.35 at `Qref` = $\sqrt{2}$ GeV.

```
1  APFEL::SetAlphaQEDRef(double alpharef, double Qref);
```

This function sets the reference values of the QED coupling α at the scale `Qref` in GeV to `alpharef`. The default is `alpharef` = $7.496252 \cdot 10^{-3}$ at `Qref` = 1.777 GeV.

```
1  APFEL::SetLambdaQCDRef(double lambdaref, int nref);
```

[3] In case the $\overline{\rm MS}$ definition for the heavy quark masses is used (see below) and the running of the masses has been enabled, this function sets the VFNS also for the running of the heavy quark masses.

This function sets the value of $\Lambda_{\rm QCD}$ in GeV with `nref` flavours to `lambdaref`. This value is used only if the `lambda` solution of the β-function equation (see below) is used to compute the running of α_s. The default is `lambdaref` $= 0.220$ GeV with `nref` $= 5$.

```
APFEL::SetPoleMasses(double mc, double mb, double mt);
```

This function sets the values of the heavy quark thresholds in GeV and sets the pole-mass scheme as a renormalization scheme for the heavy quark masses. This function is used as a default with `mc` $= \sqrt{2}$ GeV, `mb` $= 4.5$ GeV, `mt` $= 175$ GeV.

```
APFEL::SetMSbarMasses(double mc, double mb, double mt);
   sets the values of the heavy quark thresholds
   in GeV in the MSbar scheme.
```

This function, as opposed to `SetPoleMasses`, sets the values of the heavy quark masses in GeV and sets the $\overline{\rm MS}$ scheme as a renormalization scheme for the heavy quark masses. The reference scales at which the masses are defined can be specified using the `SetMassScaleReference` function (see below).

```
APFEL::SetMassScaleReference(double Qc, double Qb, double Qt);
```

This function sets the reference scales in GeV at which heavy quark masses are given. This function is effective only if the $\overline{\rm MS}$ definition for the heavy quark masses is used and has no effect if the pole masses are used. If this function is not called, `APFEL` will assume that the mass reference scales are equal to the masses themselves. In other words, in the absence of a call to this function when using the $\overline{\rm MS}$ definition for the heavy quark masses, the `SetMSbarMasses` function will define $m_c(m_c)$, $m_b(m_b)$ and $m_t(m_t)$. If instead this function is called and the reference scales are found to be different from the masses themselves, the values of $m_c(m_c)$, $m_b(m_b)$ and $m_t(m_t)$ are also evaluated by applying the RG evolution as they are needed as thresholds for the VFNS evolution when using the $\overline{\rm MS}$ definition for the heavy quark masses.

```
APFEL::EnableMassRunning(bool);
```

This function enables or disables the running of the $\overline{\rm MS}$ masses. This is effective only if the `SetMSbarMasses` is also called and in practice switches on and off the solution RG equation of the heavy quark masses. The default is `true`.

```
APFEL::SetTauMass(double mtau);
```

This function sets the values of the τ lepton in GeV to `mtau`. This function is effective only if the evolution of the lepton PDFs is enabled (see below). The default is `mtau` $=$ 1.777 GeV.

```
APFEL::SetMaxFlavourAlpha(int nf);
```

This function sets the maximum number of active flavours in the evolution of the couplings α_s and α (and the masses) to `nf`. In practice, this function forces the code not to match the solution of the β-function and γ-function equations at a given threshold if that threshold is above the maximum number allowed `nf`. The default is `nf` $= 6$.

```
APFEL::SetMaxFlavourPDFs(int nf);
```

This function sets the maximum number of active flavours in the evolution of PDFs to `nf`. In practice, this function forces the code not to match the solution of the DGLAP equation at a given threshold if that threshold is above the maximum number allowed `nf`. The default is `nf` $= 6$.

```
APFEL::SetRenFacRatio(double ratio);
```

This function sets the ratio between factorization scale μ_F (entering PDFs) and renormalization scale μ_R (entering the couplings and possibly the heavy quark masses) to `ratio`. If `ratio` is different from one, `APFEL` will assume that $\mu_R = \mu_F/$ `ratio` and, as explained above, this gives rise to additional terms in the higher order splitting functions. The default is `ratio` $= 1$.

```
APFEL::SetTimeLikeEvolution(bool);
```

This function enables or disables the time-like evolution. This evolution, as opposed to the space-like evolution used for PDFs, is used to evolve fragmentation functions (FFs). The default is `false`.

```
APFEL::SetSmallxResummation(bool, string la);
```

This function enables or disables the small-x resummation in the evolution and set the logarithmic accuracy of the resummation to `la`. The possible options for `la` are `"LL"` and `"NLL"`. The small-x resummation of the evolution relies on an external code called `HELL` that returns the difference between the fixed-order and the resummed splitting functions as explained above. By default, the small-x resummation is disabled.

```
APFEL::SetAlphaEvolution(string evol);
```

This function sets the solution of the β-function equations for the running couplings to `evol`. The variable `evol` can take the following strings:

- `"exact"`: the β-function equations are solved numerically in an exact way using the Runge-Kutta method. The boundary condition is given by the reference values of the coupling at the reference scales.
- `"expanded"`: the β-function equations are solved analytically by expanding the inverse of the β-function when computing the solution of the RG equation. See above for more details. The boundary condition is given by the reference values of the coupling at the reference scales.

- `"lambda"`: the β-function equations are solved in an analytical way in terms of the Landau pole $\Lambda_{\rm QCD}$. See above for more details.

It should be noticed that a different choice of `evol` only affects the running of α_s beyond LO while the running of α, being computed always at LO, is left unchanged. The default is `evol` = `"exact"`.

```
APFEL::SetPDFEvolution(string evolp);
```

This function sets the solution of the DGLAP equations for the evolution of PDFs to `evolp`. The variable `evolp` can take the following strings:

- `"exactmu"`: the DGLAP equation differential in the factorization scale μ_F is solved numerically in an exact way using the Runge-Kutta method.
- `"exactalpha"`: the DGLAP equation differential in the coupling α_s is solved numerically in an exact way using the Runge-Kutta method. This solution is completely equivalent to `exactmu`.
- `"expandalpha"`: the DGLAP equation differential in the coupling α_s is solved analytically by expanding the ratio between splitting functions and β-function. See above for more details.
- `"truncated"`: this solution mimics the N-space truncated solution and its implementation requires the numerical derivatives of the `expandalpha` solution. A detailed explanantion of the implementation of thsi particular solution is given above.

In all cases the boundary conditions are given by the input initial scale PDFs. The default is `evolp` = `"exactmu"`.

```
APFEL::SetEpsilonTruncation(double eps);
```

If the `truncated` evolution for PDFs has been chosen by calling the `SetPDFEvolution` function, the `SetEpsilonTruncation` function sets the truncation parameter ϵ used to compute the numerical dericatives to `eps`.

```
APFEL::SetPDFSet(string name);
```

This function sets the PDF set to be evolved to `name`. The string variable `name` can be the name of an LHAPDF set and in this case it must finish with the string `".LHgrid"`. This is needed to distinguish the LHAPDF sets from the other possible options available. Other possible options for `name` are:

- `"ToyLH"`: this option returns the so-called toy LH PDFs that have been conceived for benchmark purposes. Details can be found in Ref. ???.
- `"external"`: this option is useful to evolve an "external" set of PDFs provided by the user. If this option is chosen, `APFEL` will look for a routine called `ExternalSetAPFEL` whose structure is the following:

```
subroutine ExternalSetAPFEL(x,Q0,xf)
```

where `x` and `Q0` are the input values of the Bjorken variable and the scale in GeV, while `xf(-6:7)` is the output array whose entries from -6 to `6` are the usual quark and gluon distributions (times `x`) and the `7`-th entry corresponds instead to the photon PDF.

- `"external1"`: This option is completely analogous to `external` with the only difference that the name of the routined expected is now `ExternalSetAPFEL1`. A second slot for an external set might be useful in some cases if the user want to evolve a second PDF set in the same code.

- `"repexternal"`: This option is an extension of the previous two and the only difference is that the code expects a soubroutine called `ExternalSetAPFELRep` whose structure is:

```
subroutine ExternalSetAPFELRep(x,Q0,irep,xf)
```

where, in addition to the entries discussed above, there is the integer `irep` that identifies the `irep`-th replica of the input set. This option might be useful is the input set has more than one replica but it's not directly an `LHAPDF` set.

- `"leptexternal"`: This option is a further extension of the original `external` option in that it also provides as an output the lepton PDFs. In particular, if this option is chosen, the code looks for a subroutine called `ExternalSetAPFELLept` whose structure is the following:

```
subroutine ExternalSetAPFELLept(x,Q0,irep,xf,xl)
```

where there is the additional output array `xl(-3:3)` whose entries correspond to:

$$\begin{aligned}
\texttt{xl(-3)} &= x\tau^+(x,Q_0),\\
\texttt{xl(-2)} &= x\mu^+(x,Q_0),\\
\texttt{xl(-1)} &= xe^+(x,Q_0),\\
\texttt{xl(0)} &= x\gamma(x,Q_0),\\
\texttt{xl(1)} &= xe^-(x,Q_0),\\
\texttt{xl(2)} &= x\mu^-(x,Q_0),\\
\texttt{xl(3)} &= x\tau^-(x,Q_0).
\end{aligned}$$

In addition, as compared to the options above, the index of the array `xf` now runs from `-6` to `6` (and no longer to `7`) as the photon PDF is now the `0`-th entry of the array `xl`.

- `"pretabulated"`: This option is mainly conceived for an internal purposes but the user the expert user might want to exploit it becuase it allows one to evolve PDF sets whose functional form is not know as it makes possible to evolve PDFs starting from their value in a finite number of points. However, it is necessary to know the full structure of the internal grids used by `APFEL`. This information is anyway

(partially) accessible using some *ad hoc* functions of `APFEL`.In particular, if this option is chosen, the codes looks for a subroutine called `pretabulatedPDFs` with the following structure:

```
pretabulatedPDFs(igrid,alpha,xf,xl)
```

where the integer `igrid` identifies the subgrid and the integer `alpha` runs over the nodes of the `igrid`-th grid. Again, `xf(-6:6)` and `xl(-3:3)` are the quark and gluon, and the lepton and photon PDFs.

- `"pretabulated1"`: This option is a copy of `pretabulated` and the only difference is that in this case the code looks for a subroutine called `pretabulatedPDFs1` have the very same structure of that shown above.

- `"apfel"`: This option is again conceived for internal purposes but it might turn out to be useful if a higher performance of the evolution is required. If the `apfel` option is chosen, the code will use as initial PDFs the PDFs evolved from a previous call of the function `EvolveAPFEL`. Of course, this requires that a previous evolution with a PDF set different from `apfel` exists. This might be convenient when one want to tabulate PDFs over a grid in Q in a faster way. In fact, using the `apfel` option allows one to split a wide range in Q into small sequential steps without the need to do every single evolution starting from the same initial scale. This is convenient because the execution time is proportional to the wideness of the energy range covered by the evolution.

There are other options that are strictly used for internal purposes and thus there is no need to list them here. The default is `name` = `"ToyLH"`.

```
APFEL::SetReplica(int nr);
```

This function selects the member or replica of the PDF set to be evolved to `nr`. This function is effective only if an LHAPDF set is used and, of course, the integer `nr` must be within zero and the maximum number of members/replicas contained in the LHAPDF set in use. The default is default `nr` = 0, that is, according to the LHAPDF convention, the central member.

```
APFEL::SetQLimits(double Qmin, double Qmax);
```

This function sets the energy range in GeV within which the evolution is allowed. In practice, for a given pair of `Qmin` and `Qmax`, the initial scale `Q0` and the final scale `Q` of any evolution must be such that `Qmin < Q0, Q < Qmax`. The default is `Qmin` = 0.5 GeV and `Qmax` = 100000 GeV.

```
APFEL::SetNumberOfGrids(int ng);
```

This function sets the number of internal x-space subgrids to be used by `APFEL` to `ng`. The parameters of the single grids must be specified grid by grid using the `SetGridParameters` function discussed below. The default is `ng` = 3.

```
APFEL::SetGridParameters(int i, int n, int deg, double x);
```

This function must always be associated with one call to the `SetNumberOfGrids` function described above. It sets the parameter of the `i`-th subgrid where `i` must run from 1 to `ng` and thus there must be exactly `ng` calls to `SetGridParameters`. `n` corresponds to the number intervals (not nodes) of the `i`-th subgrid, `deg` identifies the degree of the lagrange polynomials and thus it is bound to be bigger than zero, and finally `x` is lower bound of the grid being the upper bound always assumed to be 1. It is necessary that the value of `x` is increasingly bigger as the index `igrid` increases. In practice this mean that the lower bound of the `igrid`-th subgrid must be in the range covered by the (`igrid` - 1)-th subgrid.

```
APFEL::SetExternalGrid(int i, int np, int deg, double *x);
   sets the external grid in the position i with
   np intervals, interpolation degree deg. x
   must be a one-dimentional array with upper bound
   in 1 (there cannot be more than 1 external grid).
```

```
APFEL::SetFastEvolution(bool);
   sets the fast PDF evolution (default true).
```

```
APFEL::GetVersion();
   returns the APFEL version in use.
```

```
APFEL::EnableWelcomeMessage(bool);
   enables the printing of the welcome message with
   the APFEL banner and the report of the evolution
   parameters (default true).
```

```
APFEL::EnableEvolutionOperator(bool);
   enables the computation of the external evolution
   parameters (default false).
```

```
APFEL::EnableLeptonEvolution(bool);
   enables the evolution of the lepton PDFs when the
   fast QUniD is used (default false).
```

```
APFEL::LockGrids(bool);
   locks the subgrids (default false).
```

```
APFEL::CleanUp();
```

```
    resets all the evolution parameters to the
    default settings.
```

```
 APFEL::SetLHgridParameters(int nx, int nxm, double xmin, double xm,
     double xmax, int nq2, double q2min, double q2max);
    sets the parameters of the grid over which PDFs
    will be tabulated in the LHAPDF format.
```

```
 APFEL::ListFunctions();
    lists all the functions available in APFEL.
```

```
Initialization functions:
 APFEL::InitializeAPFEL();
    initializes the APFEL library. If no settings has
    been specified, it uses the default ones.
 APFEL::EvolveAPFEL(double Q0, double Q);
    evolves PDFs on the grid to the scale Q [GeV]
    starting from the scale Q0 [GeV].
 APFEL::DeriveAPFEL(double Q);
    computes the logarithmic derivative with respect
    of Q of PDFs at the scale Q [GeV].
```

```
Output functions:
 APFEL::xPDF(int i, double x) and xgamma(double x);
    return "x" times the i-th and the photon PDF
    in "x" at the final scale "Q" [GeV] defined in
    "EvolveAPFEL".
 APFEL::xPDFall(double x, double *xf);
    returns at once "x" times all the PDF in the
    array xf[-6:6] computed in "x" at the final scale
    "Q" [GeV] defined in "EvolveAPFEL".
 APFEL::xPDFj(int i, double x) and xgammaj(double x);
    return "x" times the i-th and the photon PDF
    in "x" at the final scale "Q" [GeV] defined in
    "EvolveAPFEL" interpolated on the joint grid.
 APFEL::dxPDF(int i, double x) and dxgamma(double x);
    return "x" times the derivative in ln(Q^2) of
    the i-th and the photon PDF in "x" at the scale
    "Q" [GeV] defined in "DeriveAPFEL".
 APFEL::NPDF(int i, int N) and Ngamma(int N);
    return the N-th moment of the i-th and the
    photon PDF the final scale "Q" [GeV] defined in
    "EvolveAPFEL".
 APFEL::LUMI(int i, int j, double S);
    returns the partonic luminosity of the i-th and
    j-th partons for the CoM energy S [GeV^2] for the
    final invariant mass Mx = Q [GeV] defined in
    "EvolveAPFEL".
 APFEL::AlphaQCD(double Q);
    returns the QCD strong coupling alpha_s at the
```

```
    scale "Q" [GeV].
 APFEL::AlphaQED(double Q);
    returns the QED coupling alpha at the scale
    "Q" [GeV].
 APFEL::HeavyQuarkMass(int i,double Q);
    returns the mass of the i-th heavy quark
    (i = 4,5,6) scale "Q" [GeV] (the masses run only
    when using the MSbar scheme).
 APFEL::nIntervals();
    returns the number of intervals of the joint
    grid.
 APFEL::xGrid(int alpha);
    returns the value of "x" on the alpha-th node of
    the joint grid.
 APFEL::GetPerturbativeOrder();
    returns the perturbative order set for the
    evolution
 APFEL::ExternalEvolutionOperator(string fname, int i, int j, double x, int beta);
    returns the PDF evolution operator.
 APFEL::LHAPDFgrid(int Nrep, double Qin, string fname);
    produces a PDF interpolation grid in the LHAPDF
    format.
 APFEL::LHAPDFgridDerivative(int Nrep, string fname);
    produces an interpolation grid in the LHAPDF
    format for the derived PDFs.
```

```
---- Functions of the DIS module ----

Initialization functions:

 APFEL::InitializeAPFEL_DIS();
    initializes the DIS module. If no settings has
    been specified, it uses the default ones.
 APFEL::ComputeStructureFunctionsAPFEL(double Q0, double Q);
    computes the DIS structure functions on the grid
    at the scale "Q" [GeV] applying also the PDF
    evolution from the initial scale "Q0" [GeV].

Setting functions:

 APFEL::SetMassScheme(string ms);
    sets the mass scheme to be used to compute the
    structure functions ("ms" = "ZM-VFNS", "FFNS",
    "FONLL-A", "FONLL-B", "FONLL-C", default "ms" =
    "ZM-VFNS").
 APFEL::SetPolarizationDIS(double pol);
    sets the beam polarization (default "pol" = 0).
 APFEL::SetProcessDIS(string pr);
    sets process ("pr" = "EM", "NC", "CC", default
    "pr" = "EM").
 APFEL::SetProjectileDIS(string lept);
    sets the projectile ("lept" = "electron",
    "positron", "neutrino", "antineutrino", default
    "lept" = "electron").
```

```
APFEL::SetTargetDIS(string tar);
   sets the target ("tar" = "proton", "neutron",
   "isoscalar", "iron", default "tar" = "proton")
APFEL::SetZMass(double massz);
   sets the value of the mass of the Z boson
   (default "massz" = 91.1876 GeV).
APFEL::SetWMass(double massw);
   sets the value of the mass of the W boson
   (default "massw" = 80.385 GeV).
APFEL::SetProtonMass(double massp);
   sets the value of the mass of the proton
   (default "massp" = 0.938272046 GeV).
APFEL::SetSin2ThetaW(double sw);
   sets the value of sin^2(theta_W)
   (default "sw" = 0.23126).
APFEL::SetGFermi(double gf);
   sets the value of Fermi constant
   (default "gf" = 1.1663787e-5).
APFEL::SetCKM(double vud, double vus, double vub,
        double vcd, double vcs, double vcb,
        double vtd, double vts, double vtb);
   sets the absolute value of the entries of the
   CKM matrix
   (default: 0.97427d0, 0.22536d0, 0.00355d0,
             0.22522d0, 0.97343d0, 0.04140d0,
             0.00886d0, 0.04050d0, 0.99914d0).
APFEL::SetRenQRatio(double ratio);
   sets the ratio muR / Q (default 1)
APFEL::SetFacQRatio(double ratio);
   sets the ratio muF / Q (default 1)
APFEL::EnableDynamicalScaleVariations(bool);
   enables or disables the possibility to perform
   fact/ren scale variations point by point without
   requiring the ratio \mu_{R,F} / Q to be constant.
   Limitations: \mu_F = \mu_R and slower code.
APFEL::EnableTargetMassCorrections(bool);
   enables or disables the target mass corrections
   to the DIS structure functions due to the finite
   mass of the proton.
APFEL::EnableDampingFONLL(bool);
   enables or disables the damping factor when the
   FONLL structure functions are computed.
APFEL::SelectCharge(string selch);
   selects one particular charge in the NC structure
   functions ("selch" = "down", "up", "strange",
   "charm", "bottom", "top", "all", default
   "selch" = "all")
APFEL::SetPropagatorCorrection(double dr);
   sets the correction to the Z propagator involved
   in the NC DIS structure functions
   (default "dr" = 0).
APFEL::SetEWCouplings(double vd, double vu, double ad, double au);
   sets the vector and axial couplings of the up-
   and down-type quarks. If they are not set by the
   user the standard couplinglings are used.

```

```
Output functions:

 APFEL::F2light(double x), F2charm(double x), F2bottom(double x),
  F2top(double x), F2total(double x);
 APFEL::FLlight(double x), FLcharm(double x), FLbottom(double x),
  FLtop(double x), FLtotal(double x);
 APFEL::F3light(double x), F3charm(double x), F3bottom(double x),
  F3top(double x), F3total(double x);
    return the F2, FL and xF3 struncture functions in
    "x" at the final scale "Q" [GeV] defined in
    "ComputeStructureFunctionsAPFEL".
 APFEL::GetZMass();
    returns the value of the mass of the Z boson
 APFEL::GetWMass();
    returns the value of the mass of the W boson
 APFEL::GetProtonMass();
    returns the value of the mass of the proton
 APFEL::GetSin2ThetaW();
    returns the value of sin^2(theta_W)
 APFEL::GetGFermi();
    returns the value of Fermi constant
 APFEL::GetCKM(int u, int d);
    returns the absolute value of the (u,d) entry
    of the CKM matrix
 APFEL::GetSIATotalCrossSection(int pto, double Q);
    returns the SIA total cross section in natural
    units at the perturbative order "pto" and at the
    scale "Q" in GeV (only for time-like evolution).
 APFEL::ExternalDISOperator(string SF, int ihq, int i, double x, int ↩
     beta);
    returns the DIS operators.

Special functions for the production of FK tables:

 APFEL::SetFKObservable(string obs);
 APFEL::GetFKObservable();
 APFEL::FKSimulator(string obs, double x, double q, double y, int i, int↩
      beta);
 APFEL::FKObservables(string obs, double x, double q, double y);
 APFEL::ComputeHardCrossSectionsDY(string inputfile, string outputfile);
 APFEL::ComputeFKTables(string inputfile, double Q0, int flmap[196]);
```

Using the dafualt settings of `APFEL` this is how the evolution banner would look like:

```
Welcome to
      _/_/_/     _/_/_/_/_/   _/_/_/_/_/   _/_/_/_/_/   _/
   _/    _/   _/    _/   _/          _/          _/
  _/_/_/_/_/   _/_/_/_/_/   _/_/_/     _/_/_/     _/
 _/    _/   _/          _/          _/          _/
_/    _/   _/          _/          _/_/_/_/_/   _/_/_/_/_/
_____v2.6.1 A PDF Evolution Library, arXiv:1310.1394
      Authors: V. Bertone, S. Carrazza, J. Rojo

Report of the evolution parameters:

QCD evolution
```

```
Space-like evolution (PDFs)
Evolution scheme: VFNS at N2LO
Solution of the DGLAP equation: "exactmu" with maximum 6 active flavours
Solution of the coupling equations: "exact" with maximum 6 active ↩
    flavours
Coupling reference value:
- AlphaQCD(  1.4142 GeV) =  0.350000
Pole heavy quark thresholds:
- Mc =   1.414 GeV
- Mb =   4.500 GeV
- Mt = 175.000 GeV
muR / muF =  1.0000

Allowed evolution range [   0.50 : 100000.00 ] GeV
Fast evolution enabled
```

As clear, the most important settings are reported in the banner and we recommand to consult the banner every time that `APFEL` is run to make sure that the desired setting are actually used.

Here is the complete set of default settings used by `APFEL`:

```
APFEL::EnableWelcomeMessage(true);
APFEL::SetQLimits(0.5,100000);
APFEL::SetPerturbativeOrder(2);
APFEL::SetVFNS
APFEL::SetTheory("QCD");
APFEL::SetFastEvolution(true);
APFEL::SetTimeLikeEvolution(false);
APFEL::SetSmallxResummation(false,"NLL");
APFEL::SetAlphaQCDRef(0.35,sqrt(2););
APFEL::SetAlphaQEDRef(7.496252e-3,1.777);
APFEL::SetLambdaQCDRef(0.220,5);
APFEL::SetEpsilonTruncation(1e-5);
APFEL::SetAlphaEvolution("exact");
APFEL::SetPDFEvolution("exactmu");
APFEL::SetRenFacRatio(1);
APFEL::SetPoleMasses(sqrt(2),4.5,175);
APFEL::SetMassScaleReference(sqrt(2),4.5,175);
APFEL::SetTauMass(1.777);
APFEL::EnableMassRunning(true);
APFEL::SetMaxFlavourPDFs(6);
APFEL::SetMaxFlavourAlpha(6);
APFEL::SetPDFset("ToyLH");
APFEL::SetReplica(0);
APFEL::EnableEvolutionOperator(false);
APFEL::EnableLeptonEvolution(false);
APFEL::LockGrids(false);
APFEL::SetLHgridParameters(100,50,1e-9,1e-1,1,50,1,1e10);
APFEL::SetNumberOfGrids(3);
APFEL::SetGridParameters(1,80,3,1e-5);
APFEL::SetGridParameters(2,50,5,1e-1);
APFEL::SetGridParameters(3,40,5,8e-1);
```

As an illustration, if the user wants to perform the QCD evolution at NLO instead of the default NNLO, she/he needs to add to the code above, before the initialization routine

`InitializeAPFEL`, a call to the corresponding function, that is:

```
APFEL::SetPerturbativeOrder(1);
```

or if the user wants to use as a boundary condition for the PDF evolution a particular set available through the `LHAPDF` interface, say `NNPDF23_nlo_as_0118_qed.LHgrid`, she/he needs to call before the initialization the following function:

```
APFEL::SetPDFSet("NNPDF23_nlo_as_0118_qed.LHgrid");
```

By default, `APFEL` will use the central replica of the selected PDF set. Varying any other setting is similar, various example programs have been collected in the `examples` folder in the `APFEL` source folder.

When modifying the default settings, particular care must be taken with the number of interpolation grids, the number of points in each grid and the order of the interpolation. The default settings in `APFEL` use three grids whose ranges and number of points have been tuned to give accurate and fast results over a wide range of x. If the default parameters are modified, the user should check that the accuracy is still good enough, by comparing for instance with another run of `APFEL` with the default interpolation parameters.

The folder `examples` in the `APFEL` source directory contains several examples that further illustrate the functionalities of the code, and that can be used by the user as a starting point towards a program that suits her/his particular physics needs. All these examples are available in the three possible interfaces to `APFEL`: FORTRAN 77, `C/C++` and `Python`.

3 The DIS module