Contents

1 APFEL library documentation

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. Finally, we will describe how to install the APFEL Graphical User Interface (GUI), giving some basic examples on how to use the associated plotting modules.

1.1 Installation and basic execution

The APFEL library is available from its HepForge website:

`http://apfel.hepforge.org/`

and it can also be accessed directly from the svn repository, both for the development trunk:

```
svn checkout http://apfel.hepforge.org/svn/trunk apfel
```

as well as for the current stable release, in the case of v2.0.1 for instance one has:

```
svn checkout http://apfel.hepforge.org/svn/tags/2.0.1 apfel-2.0.1
```

We strongly recommend to use the latest stables release suggested on the APFEL website.

The installation of the APFEL library can be easily performed using the standard autotools sequence:

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

which automatically installs APFEL in /usr/local/. Note that the APFEL library requires an installation of the LHAPDF PDF library.[1] To use a different installation path, one simply needs to use the option:

[1]The current release of APFEL assumes that LHAPDF5.9.0 or a more recent version has been previously installed.

```
./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 user has at her/his disposal a set of routines that can be called from a main program. In the installation bin directory there is the apfel-config script, useful to determine the compiler flags in custom makefiles, together with a shell script apfel which starts an interactive console session of APFEL providing an immediate instrument to use the library without coding. In the following we will illustrate these functionalities and how they can be accessed by the user. The basic usage 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 routine:

```
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 routine:

```
EvolveAPFEL(Q0,Q)
```

With this routine APFEL numerically solves the discretized DGLAP equations in t 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:

```
xPDF(i,x)
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 quark flavor index according to the same convention used in the LHAPDF library, that is:

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

In APFEL we have explicitly separated the access to the quark and gluon PDFs (via xPDF) and from that to the photon PDF (via xgamma). Notice that the functions xPDF and xgamma return x times the PDFs (the momentum fractions).

In addition to the PDF values, the user can also access the integer Mellin moments of the PDFs,[2] using the routines:

```
NPDF(i,N)
Ngamma(N)
```

which are useful for instance to evaluate the momentum and valence sum rules at the scale `Q`, using `N=2` and `N=1` respectively. Finally, two functions return the value of the QCD coupling α_s and of the QED coupling α using the same settings used for the PDF evolution, these are:

```
AlphaQCD(Q)
AlphaQED(Q)
```

In `APFEL` we use the exact numerical solution of the QCD beta function equations using Runge-Kutta methods, while for the QED coupling the analytical leading-order solution is used.

In a more recent release of `APFEL` we introduced also the following function:

```
HeavyQuarkMass(n,Q)
```

which returns the value of the `n`-th heavy quark, with `n = 4,5,6`, at the scale `Q` GeV. In case the pole heavy quark masses have been chosen for the evolution (see below), this function returns the value of the `n`-th quark mass no matter the value of `Q` (the pole mass does not run). If instead the $\overline{\text{MS}}$ masses have been chosen, this function evaluates the running of the masses using as initial conditions those given in the heavy quark mass initialization.

The basic information above is enough to write a simple and yet complete program to perform 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"
using namespace std;

int main()
{
 // Define grid in x
  double xlha[11] = {1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2,
      1e-1, 3e-1, 5e-1, 7e-1, 9e-1};
```

[2]We follow the standard definition of Mellin moments:

$$\mathtt{NPDF(i,N)} \equiv \int_0^1 dx\, x^{\mathtt{N}-2}\, \mathtt{xPDF(i,x)}\,. \tag{1}$$

```

 // Precomputes evolution operators on the grids nodes
  APFEL::InitializeAPFEL();

  // Set initial and final evolution scales
  double Q02, Q2,
  cout << "Enter initial and final scales in GeV2" << endl;
  cin >> Q02 >> Q2;

  // Perform evolution
  double Q0 = sqrt(Q02);
  double Q  = sqrt(Q2);
  APFEL::EvolveAPFEL(Q0,Q);

  cout << scientific << setprecision(5) << endl;

  cout << "alpha_QCD(mu2F) = " << APFEL::AlphaQCD(Q) << endl;
  cout << "alpha_QED(mu2F) = " << APFEL::AlphaQED(Q) << endl;
  cout << endl;

  cout << "    x    "
       << setw(11) << "    u-ubar    "
       << setw(11) << "    d-dbar    "
       << setw(11) << " 2(ubr+dbr) "
       << setw(11) << "    c+cbar    "
       << setw(11) << "    gluon     "
       << setw(11) << "    photon    " << endl;

  cout << scientific;

  // Tabulate PDFs for the LHA x values
  for (int i = 0; i < 11; i++)
    cout << xlha[i] << "\t"
    << APFEL::xPDF(2,xlha[i]) - APFEL::xPDF(-2,xlha[i]) << "\t"
    << APFEL::xPDF(1,xlha[i]) - APFEL::xPDF(-1,xlha[i]) << "\t"
    << 2*(APFEL::xPDF(-1,xlha[i]) + APFEL::xPDF(-2,xlha[i])) << "\t"
    << APFEL::xPDF(4,xlha[i]) + APFEL::xPDF(-4,xlha[i]) << "\t"
    << APFEL::xPDF(0,xlha[i]) << "\t"
    << APFEL::xgamma(xlha[i]) << "\t"
    << endl;

  return 0;
}
```

This example code uses the default settings of APFEL for the evolution parameters such as perturbative order, heavy quark masses, values of the couplings etc. In the following we will discuss how the user can choose her/his own settings for the PDF evolution in APFEL.

1.2 Customization of the PDF evolution

The customization of the PDF evolution with APFEL can be achieved using a number of dedicated routines, to be called before the initialization stage, that is before calling InitializeAPFEL. These routines are:

- `SetTheory(Theory)`: this routine defines the theory to be used for the PDF evolution. The string variable `Theory` can take the following values:
  - `"QCD"` for pure QCD,
  - `"QED"` for pure QED,
  - `"QCEDP"` for QCD$\otimes$QED in parallel,
  - `"QCEDS"` for QCD$\otimes$QED in series,
  - `"QECDP"` for QED$\otimes$QCD in parallel,
  - `"QECDS"` for QED$\otimes$QCD in series,
  - `"QavDP"` for the averaged solution in parallel,
  - `"QavDS"` for the averaged solution in series,
  - `"QUniD"` for the QCD$\otimes$QED unified evolution (here the order of QCD and QED evolution does not matter).

  Let us recall that all the options for the solution of the combined QCD$\otimes$QED evolution equations are equivalent up to subleading $\mathcal{O}(\alpha\alpha_s)$ terms.

- `SetPerturbativeOrder(pt)`: this routine sets the perturbative order of the QCD evolution. The integer variable `pt` can take the values 0, 1 or 2 corresponding to LO, NLO and NNLO evolution respectively. The QED evolution, when activated, is always LO.

- `SetAlphaQCDRef(alphasref,Qref)`: this routine sets the reference value of the QCD coupling α_s, `alphasref`, at the reference scale, `Qref` in GeV.

- `SetAlphaQEDRef(alpharef,Qref)`: same as `SetAlphaQCDRef` but for the QED coupling α.

- `SetPoleMasses(mc,mb,mt)`: this routine sets the values for the heavy quark masses in the pole mass scheme. The real variables `mc`, `mb` and `mt` correspond to the numerical values in GeV of the pole heavy quark masses m_c, m_b and m_t. Calling this routine also determines that pole heavy quark masses are used as thresholds for the VFN scheme PDF evolution.

- `SetMSbarMasses(mc,mb,mt)`: this routine sets the values for the heavy quark masses in the $\overline{\text{MS}}$ scheme. Here the real variables `mc`, `mb` and `mt` correspond to the numerical values in GeV of the renormalization-group-invariant (RGI) heavy quark masses $m_c(m_c)$, $m_b(m_b)$ and $m_t(m_t)$. Calling this routine also determines that $\overline{\text{MS}}$ heavy quark masses are used as thresholds for the VFN scheme PDF evolution.

- `SetRenFacRatio(Ratio)`: this routine sets the ratio between renormalization and factorization scales. The real variable `Ratio` corresponds to the ratio μ_R/μ_F. The default choice in `APFEL` is `Ratio=1`.

- `SetVFNS`: this routine determines that the variable-flavor-number scheme is used for the PDF evolution.

- `SetFFNS(NF)`: this routine determines that the fixed-flavor-number scheme is used for the PDF evolution. The integer variable `NF` corresponds to the number of active quark flavor and can then take any values between 3 and 6.

- `SetMaxFlavourAlpha(NF)`: this routine sets the maximum number of active flavors that enter the QCD and QED beta functions for the α_s and α running. The integer variable `NF` can then take any value between 3 and 6.

- `SetMaxFlavourPDFs(NF)`: this routine sets the maximum number of active flavors that can contribute to the PDF evolution. The integer variable `NF` can then take any value between 3 and 6.

- `SetPDFSet(name)`: this routine defines the PDF set to be evolved from the initial to the final scale. The string variable `name` can take the value `"ToyLH"`, corresponding to the toy PDF model used in the Les Houches PDF benchmarks, or the name (including the `LHgrid` extension) of any PDF set available from the `LHAPDF` library.

  There is yet a third option, `name="private"`, which can be easily modified by the user (in `src/toyLHPDFs.f`) if new PDF boundary conditions not covered by the two other options are required. Note that each time a new `private` parametrization is coded, the library needs to be complied and installed again.

  Since release 2.0.2 of `APFEL` we intruduced also a number of hardcoded *fragmentation functions*.

- `SetReplica(irep)`: this routine selects the replica (for a Monte Carlo PDF set) or the specific eigenvector (for Hessian PDF sets) of the PDF set defined above to be evolved with `APFEL`. The integer variable `irep` can then take any value included between 0 (the central PDFs) and the maximum number of PDF members contained in the selected PDF set.

- `SetQLimits(Qmin,Qmax)`: this routine sets the scale bounds between which the evolution can be performed. The real variables `Qmin` and `Qmax` correspond to the numerical values of the lower and upper bounds (in GeV). On top of making sure that PDF evolution is performed only in the physical range, this option also allows to reduce the initialization time, for instance if `Qmax` is below some heavy quark thresholds, reducing the number of evolution operators to be precomputed will be smaller.

- `SetNumberOfGrids(n)`: this routine sets the number of x-space interpolation grids that will be used for the evolution. The integer variable `n` can be any positive integer number.

- `SetGridParameters(i,np,deg,xmin)`: this routine sets the parameters of the `i`-th x-space interpolation grid. The integer variable `i` must be between 1 and `n`, where the latter has been defined in `SetNumberOfGrids(n)`. The integer variable `np` corresponds to the number of (logarithmically distributed) points of the grid, the integer variable `deg` corresponds to the interpolation degree and the real variable `xmin` corresponds to the lower bound of the grid. The upper bound is always taken to be equal to one.

- `CleanUp`: this routine restores the default settings of the `APFEL` evolution. This function may be useful in case the user wants to perform PDF evolutions with different settings in the same run of her/his own code.
- `EnableWelcomeMessage(true/false)`: this routine enables (`true`) or disables (`false`) the `APFEL` welcome message. By default `APFEL` writes the welcome message, but it may be useful to disable it in case the user wants to perform a large number of evolutions in series and does not want the welcome message to be written as many times.
- `GetVersion`: this function returns the version of `APFEL` in use.
- `LHAPDFgrid(Nrep,Qin,name)`: this routine...
- `ExternalEvolutionOperator(Q0,Q,n,extgrid,M)`: this routine...
- `LUMI(i,j,s)`: this function...
- `EnableEvolutionOperator(b)`: this routine...
- `LockGrids(b)`: this routine...
- `SetTimeLikeEvolution(b)`: this routine...
- `SetAlphaEvolution(evol)`: this routine...
- `SetLambdaQCDRef(Q,n)`: this routine...
- `SetPDFEvolution(evol)`: this routine...
- `dxPDF(i,x)`: this function...
- `dxgamma(x)`: this function...
- `xPDFj(i,x)`: this function...
- `xgammaj(x)`: this function...
- `LHAPDFgridDerivative(Nrep,name)`: this routine...
- `SetFastEvolution(b)`: this routine...

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.

1.3 Computation of the DIS observables

Now we turn to the description of the module that computes the DIS neutral- and charged-current observables. This module can be either used together with the PDF evolution provided by APFEL or directly interfaced to the LHAPDF library.

The computation of DIS structure functions is provided by a single routine, DIS_xsec, which takes a set of input parameters needed to specify the computation to be performed. The usage of the DIS_xsec routine is the following:

```
APFEL::DIS_xsec(x,q0,q,y,proc,scheme,pto,pdfset,irep,target,proj,F2,F3,FL↩
    ,sigma);
```

where the *input* parameters are:

- the real variable x: the value of Bjorken x,
- the real variable q0: the value of the initial scale (in GeV) used in the PDF evolution (this input is ignored if the LHAPDF evolution is used),
- the real variable q: the value of the scale (in GeV) where the DIS observables are to be computed,
- the real variable y: the value of the inelasticity,
- the string variable proc: it can take the values "EM" for the purely electromagnetic DIS observables (photon-only exchange), "NC" for neutral-current observables and CC" for charged-current observables,
- the string variable scheme: it can take the values "FONLL" for FONLL, "FFNS" for the FFN scheme and ZMVN" for the ZM-VFN scheme,

- the integer variable `pto`,: it can take the values `0`, `1` or `2` corresponding to LO, NLO and NNLO, respectively. Notice that choosing `pto=1` with `scheme="FONLL"` implies using the FONLL-A scheme, while choosing `pto=2` with `scheme="FONLL"` leads to using the FONLL-C scheme. The implementation of the FONLL-B scheme is postponed to a future release of the program.

- The string variable `pdfset`: it can take any of the PDF sets available in `LHAPDF` (including the `.LHgrid` extension). This way `APFEL` will use the selected PDF set to compute the DIS observables using the `LHAPDF` evolution rather than the internal one. As an alternative, the user can choose `pdfset="APFEL"`. This way the DIS observables will be computed using the evolution provided by `APFEL` between the scales `q0` and `q`. For the setting of the PDF evolution, the user can refer to Sect. 1.2.

- The integer value `irep`: it specifies the member of the PDF set to be used,

- the string variable `target`: it takes the value `"PROTON"` in case the target is a proton, `"NEUTRON"` in case the target is a neutron (assuming isospin symmetry) or `"ISOSCALAR"` if the target is an isoscalar, *e.g.* a deuteron (also this option assumes isospin asymmetry). There is a further option, which is `target="IRON"`, which uses the cross-section definition used in the NuTeV experiment which is on iron nuclei.

- The string variable `proj`: it takes the values `"ELECTRON"` and `"POSITRON"` if `proc="EM"`, `"NC"`, in case the projectile is either an electron or a positron. If instead `proc="CC"`, the variable `proj` can also take the values `"NEUTRINO"` or `"ANTINEUTRINO"` with obvious meaning.

Once all these input parameters have been specified, the *output* array variables are `F2, F3, FL` and `sigma`. Each of them has 5 entries corresponding to light, charm, bottom, top and total components of the corresponding quantity. The user should be careful because in the `Fortran` interface the arrays are numbered from 3 to 7 (*e.g.* `F2(3)` $= F_2^l$, `F2(4)` $= F_2^c$, `F2(5)` $= F_2^b$, `F2(6)` $= F_2^t$, `F2(7)` $= F_2^p$), while in the C++ version the arrays run from 0 to 4 (*e.g.* `F2(0)` $= F_2^l$, `F2(1)` $= F_2^c$, `F2(2)` $= F_2^b$, `F2(3)` $= F_2^t$, `F2(4)` $= F_2^p$). As in the case of the PDF evolution, the user will find an example on how to use the DIS module in `example` folder.