Status of xFitter version 2.2 (master branch)

Ivan Novikov

xFitter workshop
DESY, Hamburg

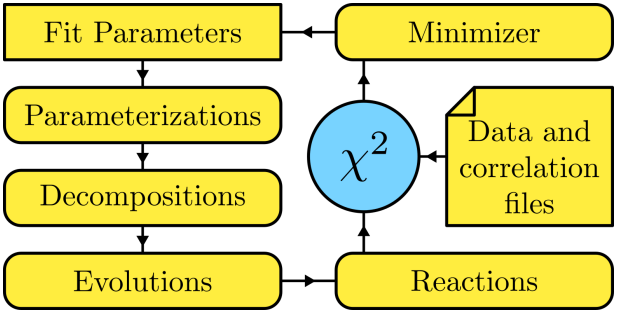
2020-02-27

Outline

- Overview of the modular fit scheme
- Other new features
- Unit tests and continuous integration
- Open merge requests
- Directions for future development

https://gitlab.cern.ch/fitters/xfitter

New features in the xFitter 2.2 (master)



- Modular fit scheme with different components separated into C++ classes, defined using a YAML steering file
- Profiler
- Improved LHAPDF6 input and output
- Support for CERES minimizer in addition to MINUIT
- Redesigned build system using cmake

Fitted parameters

```
Parameters:
      #[value, step, min, max, prior_mean, prior_sigma]
  Auv:  SUMRULE
  Buv: [ 0.810476, 0.016 ]
  Cuv: [ 4.823512, 0.06 ]
  Duv:   0 #fixed
  Euv: [ 9.921366, 0 ] #step=0 => fixed
  Adv:  SUMRULE
  Bdv: [ 1.029995, 0.06 ]
  Cdv: #alternative syntax
    value: 4.846279
    step: 0.3
  fs:   0.4
  DbarToS: "=fs/(1-fs)" #as a function of another parameter
  alphas: [0.118, 0.1] #theory parameters can also be fitted,
  #if the predictions can be recalculated at each iteration
```

Parameterizations

A `Parameterization` class represents a functional form that depends on fitted parameters. They can:

- Evaluate the function at a point
- Calculate N-th moment $\int_0^1 \mathrm{d}x x^N x f(x)$.
  If analytic integration is not implemented, numerical is used
- Set N-th moment to a given value by rescaling the normalization parameter

For example, the `HERAPDF` class implement the form:

$$xf(x) = Ax^B(1-x)^C(1 + Dx + Ex^2 + \ldots)$$

```
Parameterisations:
  u_valence: #name of parameterisation
    class: HERAPDF
    parameters: [A,B,C,D,E]
```

Parameterizations serve as input to `Decomposition` classes

Parameterizations

```
Parameterisations:
  par_g:
    class: NegativeGluon
    parameters: [Ag,Bg,Cg,ZERO,ZERO,Agp,Bgp,Cgp]
  par_dbar:
    class: HERAPDF
    parameters: [Adbar,Bdbar,Cdbar]
  par_s:
    class: Factor #multiple of some other parameterisation
    factor: DbarToS #name of parameter
    input: par_dbar
  par_s: #same, but as an expression
    class: Expression
    expression: "Adbar*fs/(1-fs)*(x^Bdbar*(1-x)^Cdbar)"
```

The `Expression` class allows defining a parameterisation as a formula. The `tinyexpr` library is used for expression parsing.

Decompositions

A `Decomposition` class rotates the parameterizaed PDFs to flavor basis and handles sum rules.

```
Decompositions:
  proton:
    class: UvDvUbarDbarS
    xuv: par_uv
    xdv: par_dv
    xubar: par_ubar
    xdbar: par_dbar
    xs: par_s
    xg: par_g
DefaultDecomposition: proton
```

A decomposition serves as input to `Evolution` classes.

Evolutions

An `Evolution` class provides evolved α_S and PDF to reaction modules. Evolution classes `QCDNUM`, `APFEL`, `APFEL++` interface these DGLAP solvers.

```
Evolutions:
  proton-QCDNUM:
    ? !include evolutions/QCDNUM.yaml
    decomposition: proton
  proton-APFEL:
    ? !include evolutions/APFELxx.yaml
  #if no decomposition is given, default decomposition is used
DefaultEvolution: proton-APFEL
```

Multiple separate evolutions can exist at once within the same fit.

Evolutions

```
Evolutions:
  proton-QCDNUM:
    ? !include evolutions/QCDNUM.yaml
    decomposition: proton
  proton-LHAPDF:
    class: LHAPDF
    set: "NNPDF30_nlo_as_0118"
    member: 0
  antiproton:
    class: FlipCharge
    input: proton-QCDNUM
DefaultEvolution: proton-APFEL
```

The class `LHAPDF` is an interface to LHAPDF. The classes `FlipCharge`, `FlipUD` swap flavors of another evolution to turn a particle into an antiparticle, or a proton into a neutron.

Reaction interface

The theory parameters and configuration settings for reaction modules can be defined at different scope, in order of decreasing specificity:

1. Dataset-specific in `TermInfo` in dataset
2. Reaction-specific in YAML steering
3. Global in YAML steering

There is now a uniform interface for a reaction module to get the parameters, as well as input PDFs. Additionally, any `double`-typed parameter can potentially be fitted.

More information can be found on the wiki:
https://gitlab.cern.ch/fitters/xfitter/-/wikis/ReactionParameters
https://gitlab.cern.ch/fitters/xfitter/-/wikis/ReactionInterface

Reaction interface

In YAML steering:

```
alphas: 0.118 #global
muR: 1 #global

Parameters:
  alphas: [0.118, 0.001] #also global, but fitted

byReaction:
  APPLgrid:
    muR: 2 #reaction-specific, overshadows global
```

In dataset, overshadows global and reaction-specific

```
TermInfo='evolution=proton;muR=4;GridName=path/to/grid.root'
```

Reaction interface

```
void MyConcreteReaction::compute(TermData* td, valarray<double>& val, ...
  double* Mz = td->getParamD("Mz");//double
  string grid_name = td->getParamS("GridName");//string
  int Npoints = 200;//Parameter with a default value
  if ( td->hasParam("Npoints") ) Npoints = td->getParamI("Npoints");
  //Get input PDFs
  xfitter::BaseEvolution* pdf1 = td->getPDF(0);// for first hadron
  xfitter::BaseEvolution* pdf2 = td->getPDF(1);// for another hadron
  //Get the PDF values at one example point
  double x = 0.5;
  double Q = 5.0;
  double xf[13]; //returned pdfs for all flavors
  pdf1->xfxQarray(x, Q, xf); //fills xf
  //Get alphaS at one example point
  double alS = pdf1->getAlphaS(Q);
}
```

Reaction modules

A `ReactionTheory` class computes theory predictions for terms in a theory expression for each dataset. The following reactions are implemented:

- `AFB`
- `APPLgrid`
- `BaseDISCC` (`QCDNUM` only)
- `BaseDISNC` (`QCDNUM` only)
- `BaseHVQMNR`
- `cbdiff`
- `FFABM_DISCC`
- `FFABM_DISNC`
- `FONLL_DISCC`
- `FONLL_DISNC`
- `Fractal_DISNC`
- `Hathor`
- `HathorSingleTop`
- `KFactor`
- `KMatrix`
- `RT_DISNC`
- `TensorPomeron`
- `HVQMNR_LHCb_7TeV_beauty`
- `HVQMNR_LHCb_7TeV_charm`
- `fastNLO`
- `DYTurbo` (work in progress)

Profiler

The `Profiler` class allows one to account for uncertainties of theory parameters and input PDFs. It generates nuisance parameters from variations of theory parameters, or from variation through error members of a LHAPDF set.

```
# Profiler allows to add variations of parameters and PDF eigenvectors
# as additional nuisance parameters
Profiler:
  Parameters:
    alphas: [ 0.118, 0.119, 0.117 ]  # central, up, (down) variation.
  #If down is not given, symmetrizes the up variation
  Evolutions:
    proton-LHAPDF:
      sets:    [CT10]
      members: [[0,1,end]] #loop through error members
  WriteTheo: "Off" # Can be "Off", "On" or "Asymmetric"
  #(to store asymmetric variations)
  getChi2: "Off"   # determine and report chi2 for each variation
```

LHAPDF6 output

Any evolution can be written to a LHAPDF6 file at the end of the fit

```
WriteLHAPDF6:
  name: "proton"
  evolution: proton-QCDNUM
  preferInternalGrid: #to get X and Q sampling points from the evolution
  #alternatively, specify them manually
  Xrange: [1e-4, 1]
  Qrange: [1,1000]
  Xnpoints: 200
  Qnpoints: 120
  #Optional information fields:
  description: "..."
  authors: "..."
  reference: "..."
```

CERES minimizer

CERES is an alternative minimizer

```
Minimizer: MINUIT # CERES
MINUIT:
  Commands: |
    set str 2
    call fcn 3
```

However, the support for CERES is not as good as for MINUIT. For example, the fit parameters are reported only to standard output.

The new build system

The build system has been completely rewritten using `cmake` The new system is faster and more reliable.
Two libraries are required: **QCDNUM** and **yaml-cpp**. All other libraries are optional, `cmake` automatically detects whether they are installed and disables optional modules/features as necessary. After installing dependencies, one can use the wrapper script:

```
./make.sh install     - configure, compile, and install
./make.sh build       - configure and compile
./make.sh             - same (configure and compile)
./make.sh run         - configure, compile, install, and run
./make.sh clean       - delete all build files
./make.sh uninstall   - delete all installed files
./make.sh reconfigure - configure from scratch
```

By default it builds in `./build` and installs in-source.

More information can be found on the wiki:
https://gitlab.cern.ch/fitters/xfitter/-/wikis/Installation

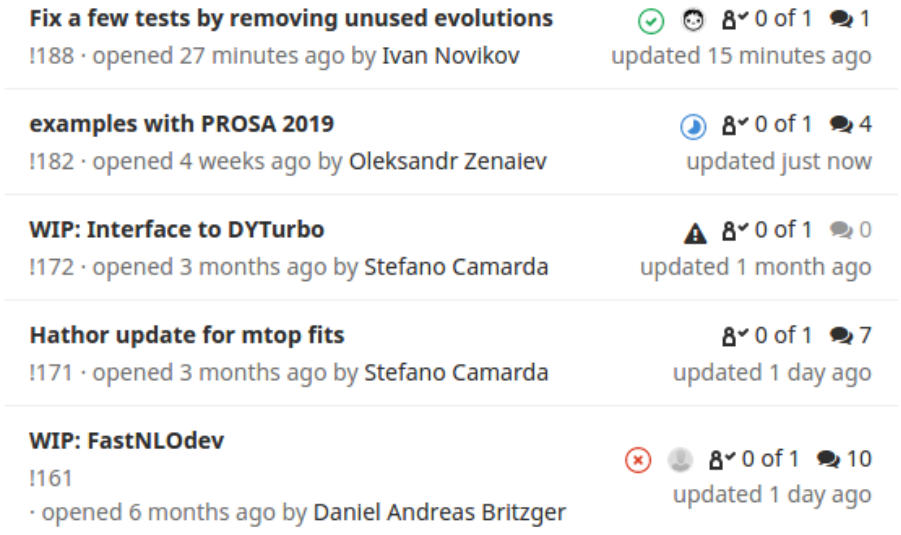
Tests

The test script runs all examples and verifies the output:

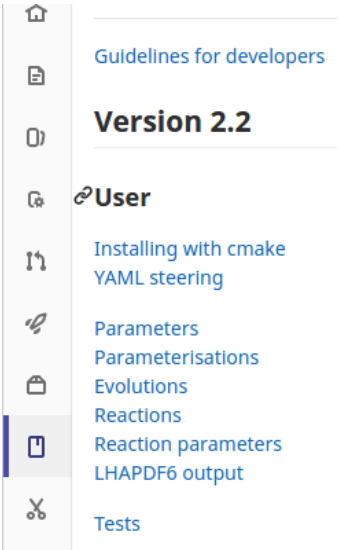
```
>./tools/test.sh
Testing AFB ... PASS
Testing ALLDATA ... PASS
Testing charmCCZEUSFFABM ... PASS
Testing charmCCZEUSFONLL ... PASS
Testing chi2scanMTOP ... FAIL
Testing defaultNLO ... PASS
Testing defaultNNLO ... PASS
Testing evolutionAPFELxx ... PASS
Testing FastNLOSymmetrize ... PASS
Testing FFABM ... PASS
Testing FONLL ... PASS
Testing HVQMNR-abs ... PASS
Testing HVQMNR-norm ... PASS
Testing LHeC ... PASS
Testing paramABMP16 ... PASS
Testing paramBG ... PASS
Testing profilerAs ... PASS
Testing profilerLHAPDF ... PASS
Testing PROSA2019FFNS ... PASS
Testing PROSA2019VFNS ... PASS
Testing SmallxResummation ... PASS
Testing TensorPomeron ... PASS
Testing ttbar3D ... PASS
-> 22 test(s) PASS
-> 1 test(s) FAIL
```

gitlab CI (Continuous Integration)

Tests build for all merge requests



Documentation: gitlab wiki



The most up-to-date documentation is available on the gitlab wiki. However, there is still much work to be done.

The wiki supports markdown formatting: lists, hyperlinks, highlighted code blocks, etc. I encourage you to read and contribute.

https://gitlab.cern.ch/fitters/xfitter/-/wikis/home

Overview of open merge requests

WIP: Interface to DYTurbo 0 of 1 0
!172 · opened 3 months ago by Stefano Camarda updated 1 month ago

Hathor update for mtop fits 0 of 1 7
!171 · opened 3 months ago by Stefano Camarda updated 2 days ago

- **Hathor update for mtop fits** and the χ^2 scan code need to be rewritten using the new C++ interface

Directions for future work

- Improve documentation: write gitlab wiki pages
- Finish `DYTurbo` reaction
- Rewrite χ^2 scan
- `C++` class for fit parameters to expose their step and limits, and output their values in a unified manner
- Move the remaining steering options to the new YAML steering
  - Improve support of `CERES` minimizer
- Actions/Tasks system to perform several actions one after another
- Provide `matplotlib`-based python plotting scripts as an alternative to `xfitter-draw`

Proposal: FitParameter class

```
class FitParameter{
 public:
  string name()const;
  double value()const;
  double step()const;
  double min()const;
  double max()const;
  double prior_mean()const;
  double prior_sigma()const;
  double* pointer()const;
  bool isFixed()const;//can be released
  bool isConstant()const;//cannot be released,
 private:
  //pointers to current value, step, min, max, etc...
};
const FitParameter Bg = xfitter::getFitParameter("Bg");
if(not Bg.isConstant()){
  //recalculate on Bg change
}//else Bg cannot change, no recalculation needed
```

Proposal: actions/tasks system

```
Actions: #executed sequentially
  - FixParameters: [Bg,Bs]
  - Fit
  - FreeParameters: [Bs]
  - Fit
  - WriteParameters: "pars_before_profiling.yaml"
  - Profile:
      Parameters:
        alphas: [0.118, 0.119, 0.117]
  - Fit
  - MakeHessianErrors
  - WriteLHAPDF6:
      name: "proton"
      evolution: proton-QCDNUM
  - WriteParameters: "parsout_final.yaml"
```

Thank you for your attention!