xFitter Release plans

S. Glazov, xFitter meeting, Krakow, 6 March 2018

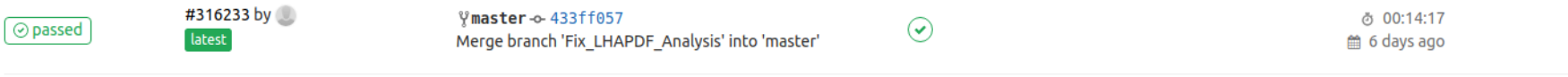
Version	Status	Progress	Start date	Release date	Description
2.0.0	RELEASED		09/Feb/17	17/Mar/17	first git release
2.1.0	UNRELEASED		09/Feb/17	31/Jul/18	New theory modules, beta

Outline

- Current status
- Next release new features present in `master`
- Remaining developments for the next release
- Long-term planing.

Release strategy

- Introduce sequence of more stable followed up by more code-development oriented releases
- The first stable git release is 2.0.0 FrozenFrog, to be followed by 2.1.0 in late summer 2018.
- Users looking for recent developments may use git `master` branch, which is ensured to run. The latest developments aimed for 2.1.0 release ARE merged to it.



Release 2.1.0

151 days left

24 Issues in version	8 Issues done	11 Issues in progress	5 Issues to do

1–24 of 24 — View in Issue Navigator

P	T	Key	Summary	Assignee	Status
		XFITTER-18	Theory module infrastructure	Alexander Glazov	IN PROGRESS
		XFITTER-56	Implement trigger of initAtiteration	Alexander Glazov	CLOSED
		XFITTER-9	Implement theory interface for DIS reactions	Alexander Glazov	IN PROGRESS
		XFITTER-20	New evolution code	Valerio Bertone	IN PROGRESS
		XFITTER-21	new HATHOR interface	Oleksandr Zenaiev	IN PROGRESS
		XFITTER-22	New FastNLO interface	Daniel Andreas Brit...	IN PROGRESS
		XFITTER-24	New interface HVQMNR	Oleksandr Zenaiev	IN PROGRESS
		XFITTER-41	Check that relevant files are copied after install	Alexander Glazov	CLOSED
		XFITTER-50	Various issues with openMP option	Alexander Glazov	CLOSED
		XFITTER-54	S-ACOT chi at NNLO	Fred Olness	IN PROGRESS
		XFITTER-59	Problems with install-yaml script	Alexander Glazov	CLOSED
		XFITTER-60	Extra minuit parameters behavior in non-fit modes	Oleksandr Zenaiev	CLOSED
		XFITTER-61	Theory interface FF ABM	Oleksandr Zenaiev	CLOSED
		XFITTER-62	verify that steering.txt.ALLdata works with the new theory interface	Oleksandr Zenaiev	IN PROGRESS

- Work on release `2.1.0` is in progress, major work done.
- Big change in theory modules interface, substantial progress already.

New theory modules interface: basic ideas

- xFitter FrozenFrog provides simple interface to include new data for existing methods of theory calculation (e.g. APPLGRID and FastNLO). The interface is flexible enough to include additional kFactors and operations, such as predictions of normalised cross sections. All is done inside text files, without need to touch xFitter code.
- Adding new theory modules, however, is a rather cumbersome task which contains in parts changes of the core xFitter code.

→ in current xFitter master branch:

- Provide new modular interface for the theory predictions, together with helper scripts.
- Loadable during execution.
- Maintain code optimization.
- Make sure that old Fortran codes are compatible with the new interface.

New theory modules: some technical details

- Currently can be tried using `master` branch, which is the starting point for release 2.1.0. Some implementation details may change
- Designed to have single theory module for (potentially) multiple data files.
- New interface class `ReactionTheory`:

```
class ReactionTheory  // must derrive from this class
  //! Main function to compute predictions for @param dataSetID
  virtual int compute(int dataSetID, valarray<double> &val, map<string, valarray<double> > &err) = 0;

  //! Perform optional re-initialization for a given iteration
  virtual void initAtIteration() {};

  //! Perform optional action when minuit fcn 3 is called (normally after fit)
  virtual void actionAtFCN3() {};

  //! Perform optional action when called from error band estimation sequence.
  virtual void errorBandAction(int ivector) {};

  //! Return pointer-function to XFX for external use
  const pXFXlike getXFX(const string& type="p") { return _xfx[type];};

  // Helper function to get a parameter (double)
  double GetParam(const string& name) const
```

- Can be then called in data files using `Reaction = 'reaction'` type; can be mixed with k-factors and other theory calculations in a usual way.

Example of `Compute`

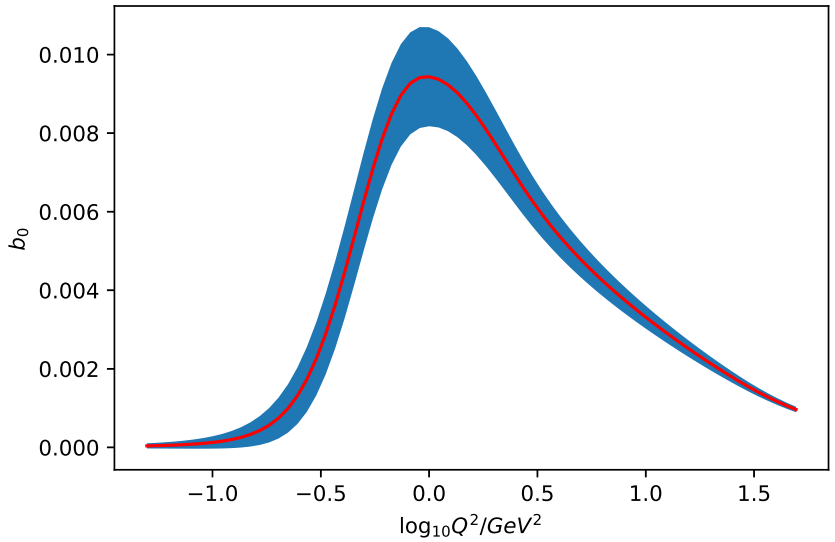
- Designed to be executed for a given dataset, update iteration-dependent parameters using `initAtIteration`.
- Snap-shot of implementation for APPLgrid (a bit simplified):

```
int ReactionAPPLgrid::compute(int dataSetID, valarray<double> &val,
                              map<string, valarray<double> > &err) {

      // Get the associated grid:
   auto grid = _grids[dataSetID];
      // Convolute it using PDFs and alphaS provided by the interface:
   val = grid->vconvolute( getXFX(), getAlphaS(),
      _order[dataSetID]-1, _muR[dataSetID], _muF[dataSetID], _eScale[dataSetID][g] );
}
```

- Note that `compute` may return theory uncertainties which can be used in the χ^2 computation.

Example of `errorBandAction`



- Recycle "bands" mechanism for fits with non-standard parameters (e.g. Dipole model).
- Simple implementation to e.g. write out predictions for given eigenvector:

```
void ReactionTensorPomeron::errorBandAction(int ivector) {
  std::string FileName = "pom_"+std::to_string(ivector)+".csv";
  writeOut(FileName);
}
```

Transfering parameters

Parameters can be specified in the data file:

```
TermInfo = 'GridName=datafiles/lhc/cms/wzProduction/1110.4973/grid-30-Z0_eta3.root',
           'GridName=datafiles/lhc/cms/wzProduction/1110.4973/grid-30-Z0_eta3.root:norm=1'
TheorExpr = 'A1/sum(A2)'
```

in the `parameters.yalm` file:

```
alphaS:
  value: 0.118
  step:  0.0
```

or in process-specific files, e.g. `yaml/reactions/APPLgrid/parameters.yaml`:

```
APPLgrid:  # APPLgrid specific
  muF: 1.0
  muR: 1.0
  Order : NLO   # APPLgrids are at max. NLO for now
```

More global parameter definition in `parameters.yaml` is re-defined by the dataset specific definition.

YAML is a simple language, can be learned in few minutes:
https://learnxinyminutes.com/docs/yaml

Testing 2.1.0 theory modules

- Implemented processes:

  APPLgrid
  BaseDISCC
  BaseDISNC
  BaseHVQMNR
  fastNLO
  FFABM_DISCC
  FFABM_DISNC
  FONLL_DISCC
  FONLL_DISNC
  Fractal_DISNC
  Hathor
  HVQMNR_LHCb_7TeV_beauty
  HVQMNR_LHCb_7TeV_charm
  KFactor (Extend to theory errors ?)
  RT_DISNC

- Missing, planned for the release: Dipole, APFELgrid, ACOT
- Optional for 2.1.0: DiffDIS, TMD

The code requires extensive testing of various options, their combination.

Testing can be started using `input_steering/steering.txt.ALLdata.thexp` which uses new theory interface for all "official" xFitter data samples.

Legacy codes in 2.1.0 and beyond

- For the release `2.1.0`, all legacy calculation codes will co-exist with the new methods.
- This will enable additional tests, however may generate some confusions: several ways to achieve the same goal.
- Warning messages will be added to the depreciated methods (however probably not to all of them).
- From the follow up release `2.X.X`, foresee major cleanup of the code, removing legacy codes.

Code optimization

- Fast computation is important for systematic studies of global PDF fits, other resource demanding studies such as MCMC uncertainties.
- Fast theory predictions using APFELgrid technology
- Fast χ^2 comutations using optimized libraries

Method	Time per iteration, msec
Standard, optimized loop order	11.0
Standard, "wrong" loop order	17.3
Ubuntu 16.04 default BLAS	17.5
Intel MKL library	1.4
CUDA cublas	7.7

Example how to enable is in the `steering.txt` file, `ReduceSyst` namelist:

```
&ReduceSyst
  do_reduce = .true.  ! turn-on to simplify/speedup chi2 calculation.
  tolerance = 0.0 ! keep exact calculation, > 0.0 further improved speed.
  useBlas = .true. ! turn on BLAS libraries
&End
```

Keeping up with the data

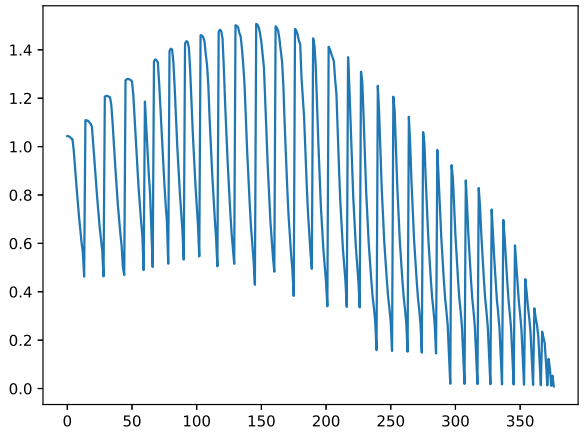
No	Collider	Experiment	Reaction	arXiv	Readme
1	fixedTarget	bcdms	inclusiveDis	cern-ep-89-06	README
2	hera	h1	beautyProduction	0907.2643	
3	hera	h1	inclusiveDis	1012.4355	
4	hera	h1	jets	0706.3722	README
5	hera	h1	jets	0707.4057	README
6	hera	h1	jets	0904.3870	README
7	hera	h1	jets	0911.5678	README
8	hera	h1	jets	1406.4709	README
9	hera	h1zeusCombined	charmProduction	1211.1182	
10	hera	h1zeusCombined	inclusiveDis	0911.0884	
11	hera	h1zeusCombined	inclusiveDis	1506.06042	
12	hera	zeus	beautyProduction	1405.6915	
13	hera	zeus	diffractiveDis	0812.2003	
14	hera	zeus	jets	0208037	
15	hera	zeus	jets	0608048	
16	hera	zeus	jets	1010.6167	
17	lhc	atlas	drellYan	1305.4192	
18	lhc	atlas	drellYan	1404.1212	
19	lhc	atlas	jets	1112.6297	
20	lhc	atlas	jets	1304.4739	
21	lhc	atlas	topProduction	1406.5375	
22	lhc	atlas	topProduction	1407.0371	
23	lhc	atlas	wzProduction	1203.4051	
24	lhc	atlas	wzProduction	1612.03016	README
25	lhc	cms	jets	1212.6660	
26	lhc	cms	jets	1609.05331	
27	lhc	cms	topProduction	1208.2671	
28	lhc	cms	topProduction	1211.2220	
29	lhc	cms	topProduction	1703.01630	
30	lhc	cms	topProduction	cms-pas-top-11-024	
31	lhc	cms	wzProduction	1110.4973	
32	lhc	cms	wzProduction	1206.2598	
33	lhc	cms	wzProduction	1310.1138	
34	lhc	cms	wzProduction	1312.6283	
35	lhc	cms	wzProduction	1603.01803	
36	lhc	lhcb	beautyProduction	1306.3663	
37	lhc	lhcb	charmProduction	1302.2864	
38	lhc	lhec	inclusiveDis	1206.2913	README

We should try to be more pro-active adapting exising data to `xFitter`.
Resent new samples come from CMS/ATLAS analyses directly, more published data/theory grids are present.

Python interface

```
#!/usr/bin/env python
import sys
sys.setdlopenflags(256|1)   # MAGIC !!!
sys.path.append("./lib")

import libxfitter_fit as xfitter
# execute using standard steering files:
xfitter.logo()
xfitter.read_steer()
xfitter.init_pars()
xfitter.read_data()
xfitter.init_theory()
xfitter.fit()
import matplotlib.pyplot as plt
th = [xfitter.theo(i) for i in range(xfitter.ndata())]
plt.plot(th)
plt.savefig("th.pdf")
```



- Present since FrozenFrog release, updated from `f2py` to `boost-python` C++ inteface possibility to execute `xFitter` from `Python`.
- Potentially extend further, to allow `Python` theory modules, get access to fitted parameters during the execution, etc.

Modular evolution

- The core of xFitter is the QCDNUM based evolution.
- The evolution code is flexible, fast, and resource friendly. QCDNUM allows for simple usage of other evolution programs.
- Nevertheless for many tasks QCDNUM is not required, e.g. LHAPDF6+Grid based profiling, kT-evolution.
- Several new developments are foreseen in changing the evolution codes: would be nice to allow inclusion of them without touching core of xFitter.

$\rightarrow$

- New modular evolution, move to C++ QCDNUM.
- Loadable during execution.
- Interfaces to the new ReactionTheory modules, χ^2 codes, PDFs, α_S, etc.

No concrete code yet, only ideas.

New physics in 2.1.0

- Already included or in planning:
  - Total $t\bar{t}$ cross section: updates to `Hathor2.0` and new implementation of `Top++`.
  - SACOT$-\chi$ at NNLO.
  - New TMD codes.
- May be added:
  - Interfaces to DY resummation codes.
  - ACOT using fast QCDNUM convolution.
  - NNLO CC RT-scheme
  - MMHT tolerance method
  - Updates in Hessian PDF uncertainties:
    - Update in ITERATE error band code
    - Introduce Levenberg-Marquardt algorithm
  - Merge with `mcgen`
  - IPython/mathematica notebooks
- Other improvements may come "automatically" with improvements of FastNLO and APPLGRID.

Beyond 2.1.0

- Fully modular code
- Revision of user interfaces
- Better connection to HepData
- Improved PDF parameter transfer.
- Better intefraces for non-standard PDF fits:
  - Nuclear PDFs
  - Meson PDFs
- Fragmentation functions, mixed PDF/FF fits.

$\rightarrow$ user's input is essential for future planning.

Improved PDF parameter transfer.

- Move away from PDF parameterization using old-style minuit steering
- Define decomposition:

```
Composition  = 'u=uval+usea',
               'd=dval+dsea',
```

- Define free parameters:

```
Parameters    =
                'Buv =  0.5 0.001 [ 0.1  20.]',
                'Cuv =  0.3 0.01',
```

- Define PDF functional form:

```
Parametrisation = 'uval = Auv * x^Buv * (1-x)^Cuv * (1 + Fuv*x^0.5 + Duv*x + Euv*x*x)',
                  'dval = Adv * x^Bdv * (1-x)^Cdv * (1 + Fdv*x^0.5 + Ddv*x + Edv*x*x)',
```

Development frozen since late 2016, can be re-activated. Can be merged with updates towards a modular minimization package.

Summary

- Stable release 2.0.0 FrozenFrog: preparation for major leap forward. In active use by most of the users.
- Planned release 2.1.0 with significant changes in the internal structures.
- Simplifications of the code development and maintenance.
- A few theory developments to be included.
- Follow up release 2.X.X with removal of legacy codes, modular evolution and minimization.

→ User's feedback is needed to define long-term strategy.

Extras

Getting full install

To get most of the packages automatically, use `install-xfitter` script from the release or from the xFitter download page.

```
tools/install-xfitter

usage:
tools/install-xfitter <version|deps>

available versions:
2.0.0
master

to reinstall only dependences, run:
tools/install-xfitter deps
```

- For SL6, CentOS7 uses /cvmfs/sft.cern.ch compilers
- Most of the main packages are installed and configured: LHAPDF6, APPLGRID (HOPPET), APFEL, APFELGRID, MELA.
- Optional script for YAML libraries (`tools/install-yaml`)

Getting data

The release comes with minimal set of data files. Many more data are available from http://xfitter.hepforge.org/data.html and can be downloaded using helper script `xfitter-getdata.sh`

```
bin/xfitter-getdata.sh -p
-------------------------------------------------------------------------------
 available data sets in xFitter (to download use arXivNumber or reportNumber)
      Collider          Experiment            Reaction   arXivNumber or reportNumber
   fixedTarget               bcdms        inclusiveDis                 cern-ep-89-06
          hera                  h1    beautyProduction                     0907.2643
...
           lhc                atlas            drellYan                     1305.4192
...
           lhc                  cms                jets                     1212.6660
...
      tevatron                  cdf                jets                     0807.2204

bin/xfitter-getdata.sh 1212.6660
```

The script has other useful options, e.g. download all the data at ones.

→ Future releases may also include automatic download of the APPLGRID/FastNLO tables.

User interface

Standard data file format using "namelist" format (predating HTML):

```
&DATA
  Name = 'ATLAS 2.76 TeV Jet data 0 <= |y| < 0.3 R=0.4'
  NDATA = 10
  NColumn = 28
  ColumnType = 2*'Bin',2*'Dummy','Sigma','Error',22*'Error'
  ColumnName = 'pt1','pt2','YBinSize','NPCorr','Sigma','stat',
               'JES1_RelSys_1' , 'JES2_RelSys_7'
  ...
  Reaction  = 'pp jets APPLGRID'
  TheoryType = 'expression'
  TermName = 'A1', 'K'
  TermType = 'applgrid','kfactor'
  TermSource = 'datafiles/lhc/atlas/jets/1304.4739/atlas-incljets-R04-eta1.root',
              'datafiles/lhc/atlas/jets/1304.4739/inclusivejetsKF_R04_00_03.dat'
  TheorExpr= 'K*A1*1000'
...
&End
```

Moveing to YAML:

```
SetDesc: HERAPDF
Authors: ...
Reference: ...
Format: lhagrid1
DataVersion: 1
NumMembers: 1
Flavors: [-6, -5, -4, -3, -2, -1, 1, 2, 3, 4, 5, 6, 21]
....
```

→ perhaps we will allow for both formats for future releases.

BaseDIS interface

```
Data file:

  TheoryType = 'expression'
  TermName = 'R'
  TermType = 'reaction'
  TermSource = 'use:hf_scheme_DISCC'  ! can be set in parameters.yaml
 ! TermSource = 'BaseDISCC'  ! can be assigned directly
  TermInfo = ''
  TheorExpr = 'R'

parameters.yaml file:

hf_scheme_DISCC :
  value : 'BaseDISCC'  # this is zero mass scheme
```

→ implementation very similar the to NC scheme.

Updates in parameters.yaml file

```
parameters.yaml file:

#
# RT DIS scheme settings:
#-------------------------
include : yaml/reactions/RT_DISNC/parameters.yaml

reactions/RT_DISNC/yaml/parameters.yaml file:

RT_DISNC:                  # Reaction-specific settings
  varin : [0.0, 1.0, -0.666666666667, 1.0]
```

- (optionally) Move theory-module specific settings to `reactions/MODULE_NAME/yaml/parameters.yaml` file.
- These files are installed automatically to `--prefix/yaml/reactions/MODULE_NAME/parameters.yaml`
- `tools/AddReaction.py` generates required directories, updates `Makefile.am` automatically.

(further) optimization of χ^2 computation

- Recall that slowest part of χ^2 computation was matrix multiplcation of $S^s_d = S(N_{\rm sys}, N_{\rm data})$, $C^{s_1 s_2} = \sum_d S^{s_1}_d S^{s_2}_d = S S^T$ (here, opposite to the data covariance matrix, the sum runs over data points and not over the systematic sources.)
- Signiciant $\sim \times 1.5 - 3$ speedup when the loop over data points is moved from inner most to outermost (probably releated to CPU L1/L2 and L3 cache sizes).
- Try to use standard BLAS libraries which may be significantly optimized for the architecture. Use:
  - Default ubuntu 16.04 library
  - Intel optimized `mkl` library
  - GPU-accelerated `cublas`

χ^2 computation time comparison

Method	Time per event, msec
Standard, optimized loop order	11.0
Standard, "wrong" loop order	17.3
Ubuntu 16.04 default BLAS	17.5
Intel MKL library	1.4
CUDA cublas	7.7

- Test using HERA2 inclusive data, new theory interface, ZMVFNS on Dell precision 5520 (Core i7-7820HQ, Nvidia Quadro M1200 GPU), gcc version 5.4, default O2 optimization.
- Intel MKL is better vs standard computation by factor > 5.
- CUDA is not as fast, probably limited by CPU-GPU data transfer.