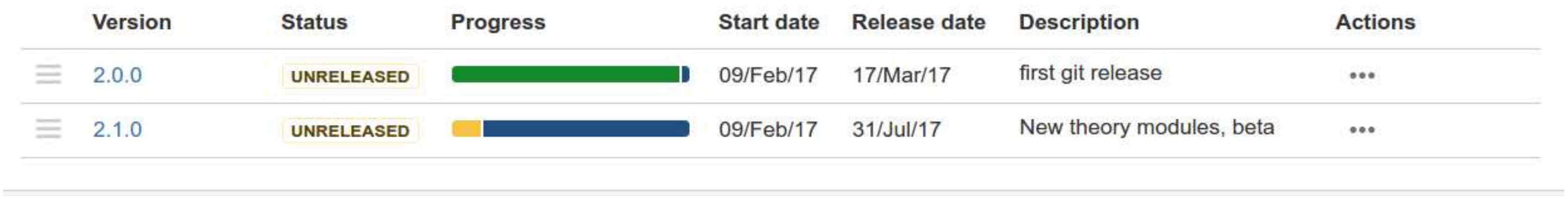
xFitter Releases

S. Glazov, xFitter meeting, Oxford, 21 March 2017

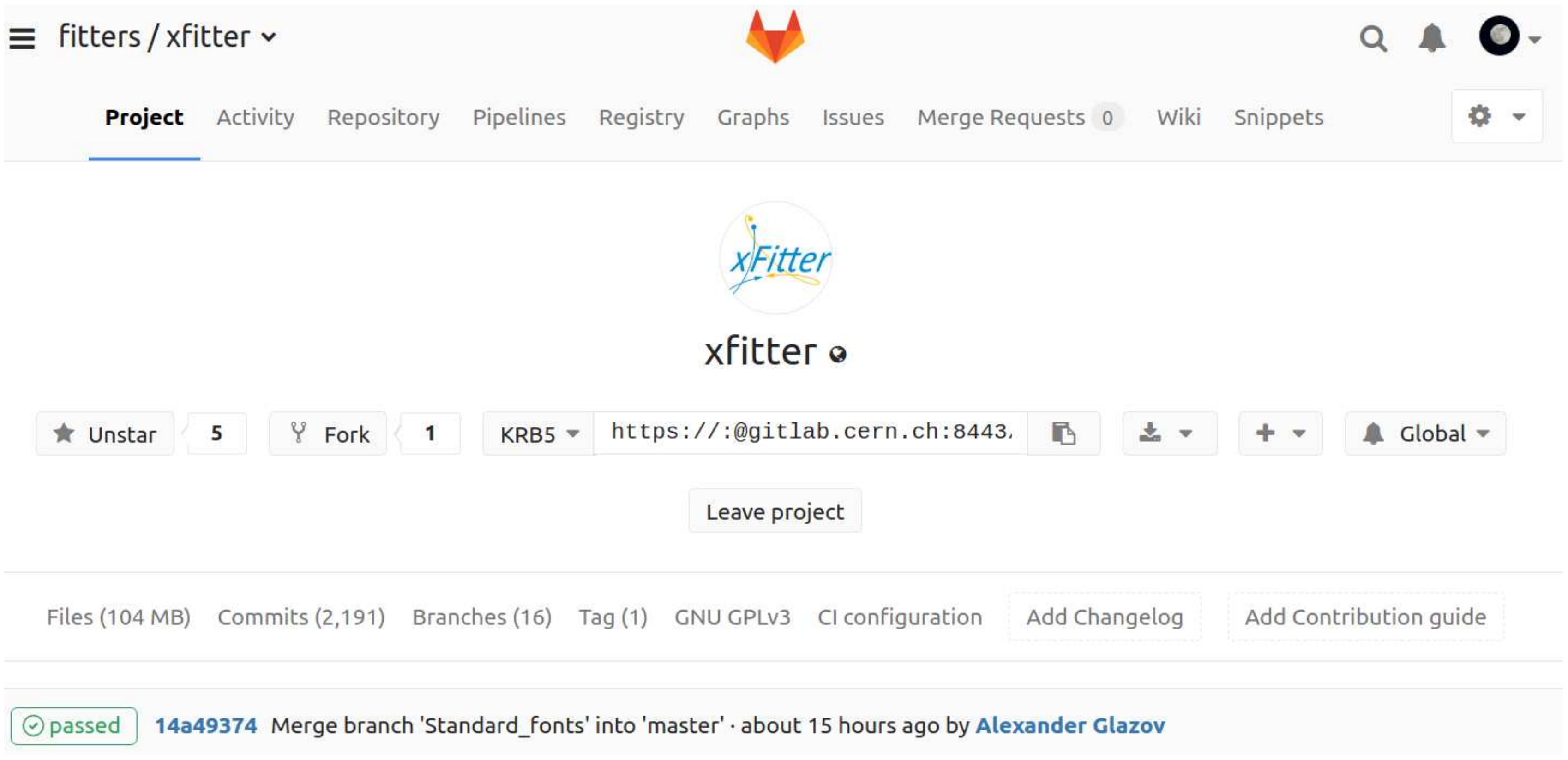


Version	Status	Progress	Start date	Release date	Description	Actions
2.0.0	UNRELEASED		09/Feb/17	17/Mar/17	first git release	•••
2.1.0	UNRELEASED		09/Feb/17	31/Jul/17	New theory modules, beta	•••

Outline

- Updates in code management: gitlab, jira
- Release strategy
- The first git release: 2.0.0 FrozenFrog
- Step forward: 2.1.0
- Long-term planing.

Work flow


fitters / xfitter

Project | Activity | Repository | Pipelines | Registry | Graphs | Issues | Merge Requests 0 | Wiki | Snippets

xfitter

Unstar 5 | Fork 1 | KRB5 | https://:@gitlab.cern.ch:8443/ | Global

Leave project

Files (104 MB) | Commits (2,191) | Branches (16) | Tag (1) | GNU GPLv3 | CI configuration | Add Changelog | Add Contribution guide

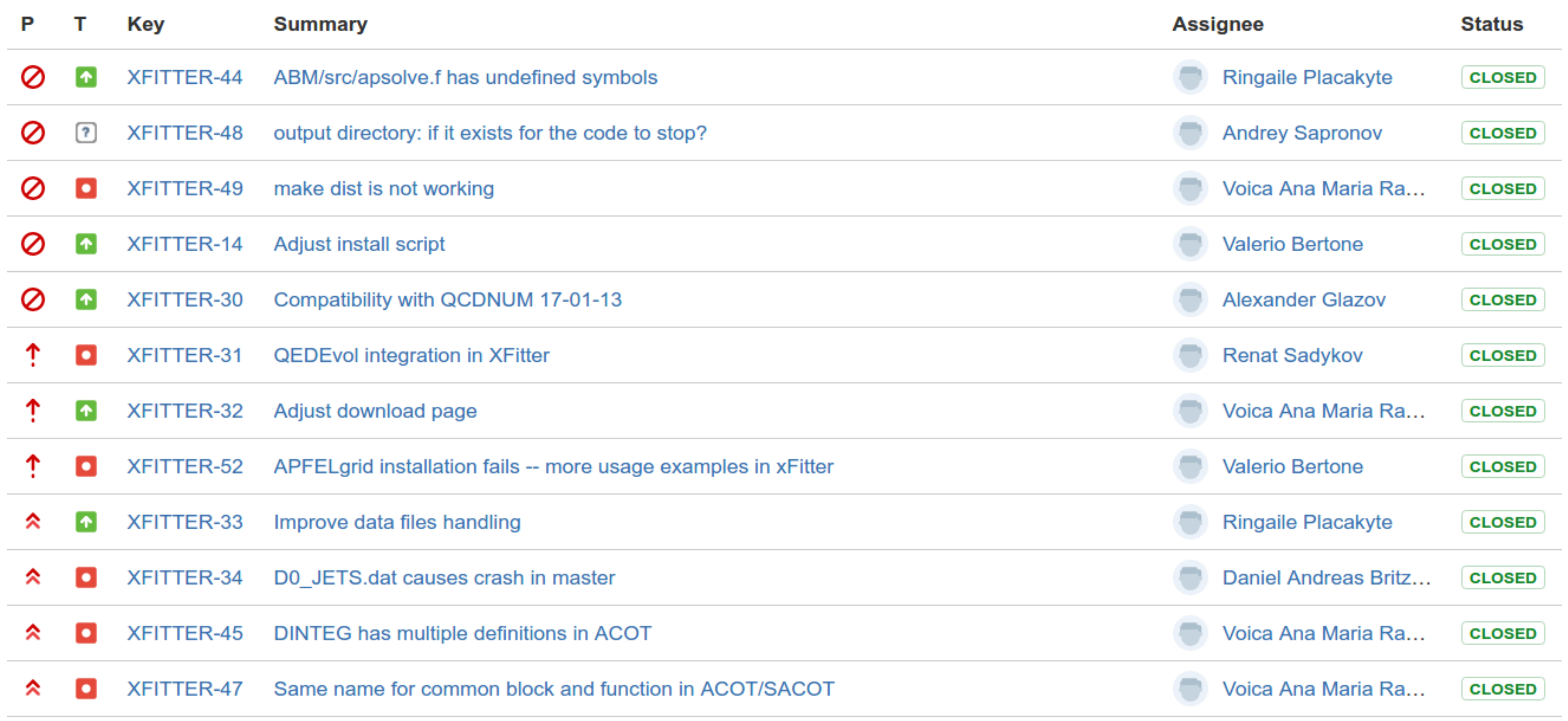
passed 14a49374 Merge branch 'Standard_fonts' into 'master' · about 15 hours ago by Alexander Glazov

- Make use of CERN GitLab and JIRA for the code management and controlling feature development and bug fixes. The git repository is open for reading for everybody.
- Developers without CERN account can contribute using open mirror `https://gitlab.com/fitters/xfitter`.

Release strategy

- Introduce sequence of more stable followed up by more code-development oriented releases
- First stable git release is 2.0.0 FrozenFrog, to be followed by 2.1.0 in late summer
- A few developments on 2.1.0 already, using git dedicated branch, while last $\sim$ month before the release was focused to make sure that 2.0.0 is stable.
- Users looking for recent developments may use git master branch, which is ensured to be compilable. The latest developments aimed for 2.1.0 release will be soon merged to it.

Release 2.0.0: FrozenFrog



P	T	Key	Summary	Assignee	Status
⊘	⬆	XFITTER-44	ABM/src/apsolve.f has undefined symbols	Ringaile Placakyte	CLOSED
⊘	?	XFITTER-48	output directory: if it exists for the code to stop?	Andrey Sapronov	CLOSED
⊘	🟥	XFITTER-49	make dist is not working	Voica Ana Maria Ra...	CLOSED
⊘	⬆	XFITTER-14	Adjust install script	Valerio Bertone	CLOSED
⊘	⬆	XFITTER-30	Compatibility with QCDNUM 17-01-13	Alexander Glazov	CLOSED
↑	🟥	XFITTER-31	QEDEvol integration in XFitter	Renat Sadykov	CLOSED
↑	⬆	XFITTER-32	Adjust download page	Voica Ana Maria Ra...	CLOSED
↑	🟥	XFITTER-52	APFELgrid installation fails -- more usage examples in xFitter	Valerio Bertone	CLOSED
︽	⬆	XFITTER-33	Improve data files handling	Ringaile Placakyte	CLOSED
︽	🟥	XFITTER-34	D0_JETS.dat causes crash in master	Daniel Andreas Britz...	CLOSED
︽	🟥	XFITTER-45	DINTEG has multiple definitions in ACOT	Voica Ana Maria Ra...	CLOSED
︽	🟥	XFITTER-47	Same name for common block and function in ACOT/SACOT	Voica Ana Maria Ra...	CLOSED

- First git release, focused on stability of the code.
- No new major changes in the code structure, freezing old developments.
- A few rarely used options checked and some of them fixed to work again.
- Work of many developers helping to release the code.

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.

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.

Getting fast with APFELgrid

The *pp* APPLGRIDs can be converted to “APFELgrids” which combine evolution and convolution in one step, saving both computing time and memory. In xFitter this can be done on the fly, by using ’APFELGRID’ as the reaction type and changing the file name suffix to ’.fk’. In addition, there is a helper script for that:

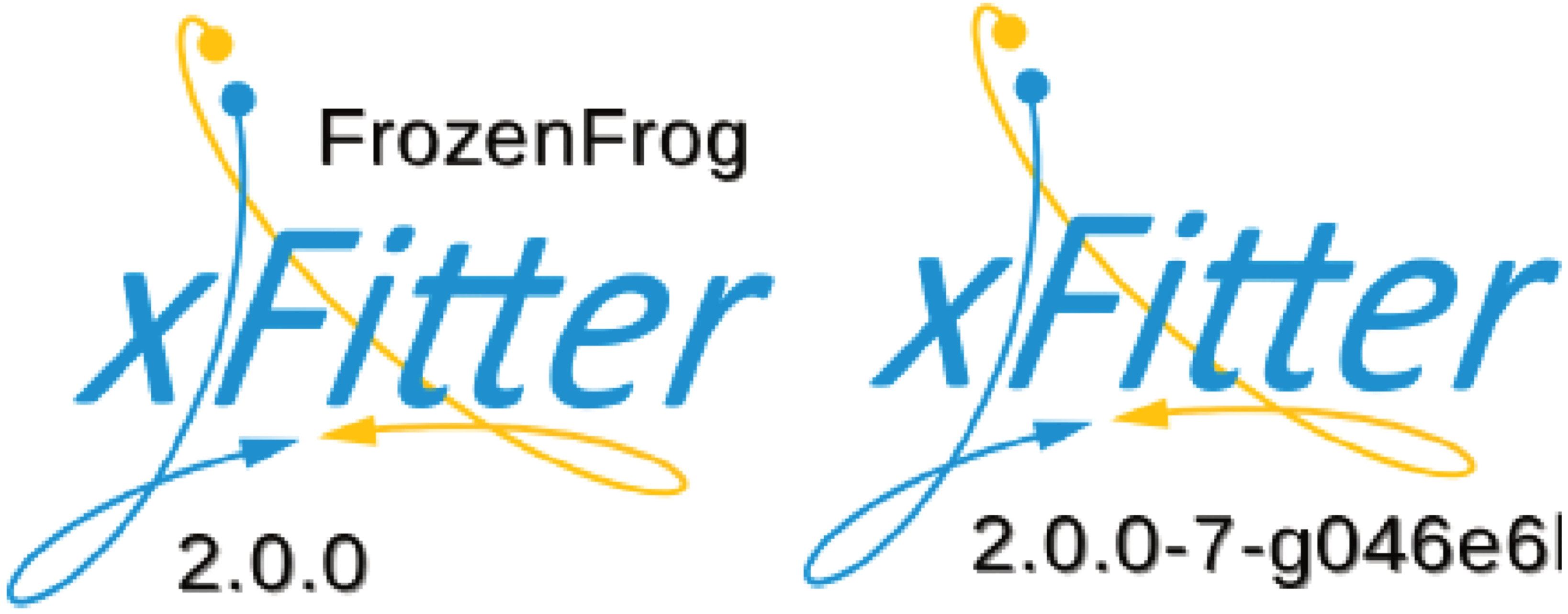
```
tools/ToAppfelGrid.py

Usage:
   file.dat       -- convert data file, generate file.dat_fk
   steering.txt   -- convert steering.txt and connected data files, generate steering.txt_fx (pl
   ALL            -- convert all data files following standard datafiles/  data tree, generate _
   steering.txt SPLIT -- prepare steering.txt files to run in parallel on the batch, to speedup
```

Example speedup: for ATLAS `inclusivejets_R06_00_03.dat` from 65 to 25 msec per iteration (probably χ^2 calculation dominated).

Computation of fk grids requires some time; they are to be re-calculated if evolution parameters such as m_C, m_B, α_S change.

Logo matters



- Default logo for the release displays release name
- For working on a git branch, number of commits from the release and last commit label are displayed (from `git describe`)
- `--no-version` removes version, as before.
- `LiberationSans-Regular.ttf` is used by default, best matching font (selected by `fc-match`) is used if not present. Recommendation is to install `LiberationSans-Regular.ttf` for common look.

User interface

Standard data file format using “namelist” format (predating HTML):

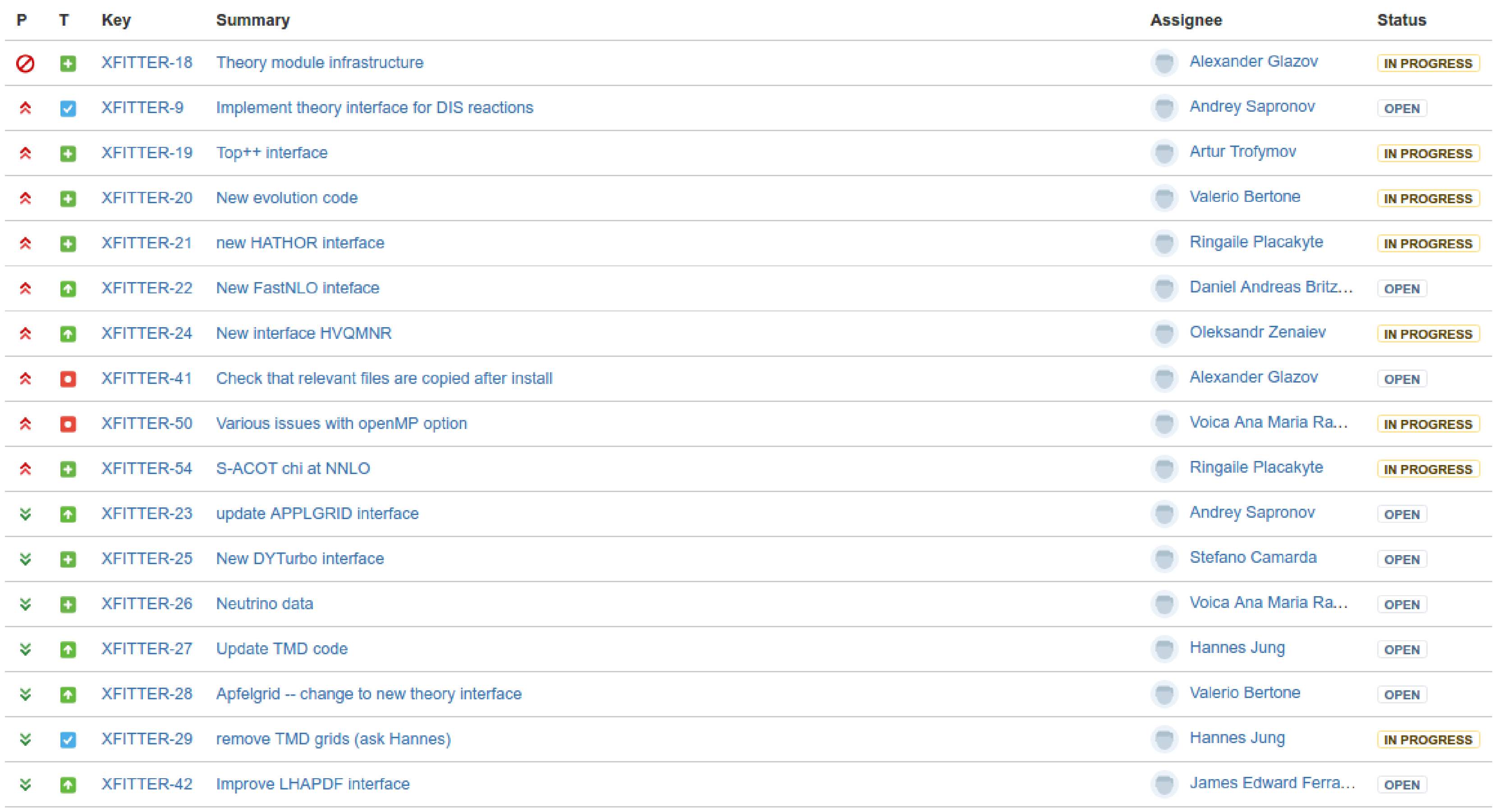
```
&DATA
  Name = 'ATLAS 2.76 TeV Jet data 0 <= |y| < 0.3 R=0.4'
  NDATA = 10
  NColumn = 28
  ColumnType = 2*'Bin',2*'Dummy','Sigma','Error',22*'Error'
  ColumnName = 'pt1','pt2','YBinSize','NPCorr','Sigma','stat',
                'JES1_RelSys_1' , 'JES2_RelSys_7'
  ...
  Reaction  = 'pp jets APPLGRID'
  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
```

Move to yaml ? Example of LHAPDF info file:

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

$\rightarrow$ perhaps we will allow for both formats for future releases.

Release 2.1.0



P	T	Key	Summary	Assignee	Status
⊘	➕	XFITTER-18	Theory module infrastructure	Alexander Glazov	IN PROGRESS
⇈	☑	XFITTER-9	Implement theory interface for DIS reactions	Andrey Sapronov	OPEN
⇈	➕	XFITTER-19	Top++ interface	Artur Trofymov	IN PROGRESS
⇈	➕	XFITTER-20	New evolution code	Valerio Bertone	IN PROGRESS
⇈	➕	XFITTER-21	new HATHOR interface	Ringaile Placakyte	IN PROGRESS
⇈	⬆	XFITTER-22	New FastNLO inteface	Daniel Andreas Britz…	OPEN
⇈	⬆	XFITTER-24	New interface HVQMNR	Oleksandr Zenaiev	IN PROGRESS
⇈	🟥	XFITTER-41	Check that relevant files are copied after install	Alexander Glazov	OPEN
⇈	🟥	XFITTER-50	Various issues with openMP option	Voica Ana Maria Ra…	IN PROGRESS
⇈	➕	XFITTER-54	S-ACOT chi at NNLO	Ringaile Placakyte	IN PROGRESS
⇊	⬆	XFITTER-23	update APPLGRID interface	Andrey Sapronov	OPEN
⇊	➕	XFITTER-25	New DYTurbo interface	Stefano Camarda	OPEN
⇊	➕	XFITTER-26	Neutrino data	Voica Ana Maria Ra…	OPEN
⇊	⬆	XFITTER-27	Update TMD code	Hannes Jung	OPEN
⇊	⬆	XFITTER-28	Apfelgrid -- change to new theory interface	Valerio Bertone	OPEN
⇊	☑	XFITTER-29	remove TMD grids (ask Hannes)	Hannes Jung	IN PROGRESS
⇊	⬆	XFITTER-42	Improve LHAPDF interface	James Edward Ferra…	OPEN

- Work on release `2.1.0` is in progress, major goals and tasks distributed.
- Major change in theory modules interface, substantial progress already. See talks from Sasha and Artur later today.

New theory modules interface: basic ideas

- xFitter 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.

$\rightarrow$

- 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 `theory-interface` branch, which is the starting point for release 2.1.0. Implementation details may change

New interface class `ReactionTheory`:

```
class ReactionTheory  // must derrive from this class
  virtual string getReactionName() const =0;  // Should return expected reaction name.
                                              // normally generated automatically by AddReaction.py
                                              // To be used in data files.
  virtual int  initAtStart(const string &) =0; // Initialization first time ReactionTheory implementation is called
  virtual int compute(int dataSetID, valarray<double> &val, map<string, valarray<double> > &err) = 0;
                                               // Return back results for a given dataset. Optionally return uncertainties
...
     // Global parameters are transferred as maps:
  virtual void setxFitterParameters(map<string,double*> &xfitter_pars) {_xfitter_pars = xfitter_pars; }; ///< Set environment map
  virtual void setEvolFunctions(double (*palpha_S)(double *), map<string, pTwoParFunc> *func2D  ) { alpha_S = palpha_S; PDFs = func2D
                                ///< Set alpha_S and PDF maps

...
  // Helper functions to emmulate LHAPDF6 calls:
  void xfx(double x, double q, double* results){double q2=q*q; (*_xfx)(&x,&q2,&results[0]); };
  double xfx(double x, double q, int iPDF){ double pdfs[13]; xfx(x,q,pdfs); return pdfs[iPDF+6];};

  // Helper function to get a parameter
  double GetParam(string name) const
  {
    return *_xfitter_pars.at(name);
  }
```

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.

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.
- Loadable during execution.
- Interfaces to the new ReactionTheory modules, χ^2 codes, PDFs, α_S, etc.

No concrete code yet, only ideas.

Gluing things together

- Already in FrozenFrog xFitter is provided as a shared library.
- Further experimental developments in `boost_python` branch which uses boost libraries for python-c++ interface.
- Different modules can be loaded in scripts as they needed.
- Full implementation is beyond 2.1.0

```
#!/usr/bin/env python

import sys

sys.path.append("./lib")
import libxfitter_fit as xfitter

# execute using standard steering files:
xfitter.logo()
xfitter.read_steer()
xfitter.read_data()
xfitter.init_theory()
xfitter.fit()
```

New physics in 2.1.0

- Already included 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 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
- Alternative evolutions (e.g. inclusion of resummation)
- Fragmentation functions, mixed PDF/FF fits.

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

Sharing is good

xFitter is built on contributions from many people, many groups.

We have a policy to acknowledge them all, described in REFERECES file, distributed with the package. We checked/updated this file for the FrozenFrog release.

Further ways to share:

- Share data files (we can arrange basic checks)
- Share appl/FastNLO grids – we should agree what is the best place to store them.

Summary

- New stable release 2.0.0 FrozenFrog: preparation for major leap forward.
- 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.

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

Extras

New theory modules interface: basic ideas

- Based on C++ code. To simplify interface to fortran codes, use single instance of an object for each theory module working on multiple data sets.
- Theory modules are included as a part of expression interface: usual operations as for APPLGRID and fastNLO are supported.
- Connection between core of xFitter and theory modules is implemented using standard C++ objects, ensuring modularity: modules can be developed outside xFitter.
- Theory modules are stored as shared libraries, loaded at run-time. Currently loading is performed in C++ code, can eventually be replaced by python.

Communication between modules and xFitter

- Basic idea: use standard library maps.
- xFitter, minuit parameters: maps of double, e.g.
  double fs = pars['fs'];
  This means that ExtraParameters are auotmatically visible by the modules, without need for extra xFitter code.
- α_S, PDFs: maps of functions, e.g.
  double (*xg)(double *, double *) = PDFs['xg'];
  double gluon = xg(0.001,10.0);
- Data bins are provided as maps of vectors of doubles, e.g
  double y[] = BINS['y'];

Interface class for the module

```
typedef double (*pxFx)(double*, double*);
typedef double (*pZeroParFunc)();
typedef double (*pOneParFunc)(double*);
typedef double (*pTwoParFunc)(double*, double*);

class ReactionTheory
{
 public:
  ReactionTheory() {};
  ~ReactionTheory() {};

  ReactionTheory(const ReactionTheory &);
  ReactionTheory & operator =(const ReactionTheory &);

 public:
  virtual string getReactionName() const =0;
  virtual void initAtStart(const string &) =0;
  virtual void setxFitterParameters(map<string,double> &xfitter_pars) {*_xfitter_pars = xfitter_pars; };
  virtual void setEvolFunctions(double (*palpha_S)(double *) , map<string, pxFx> &) { alpha_S = palpha_S; };
  virtual void setExtraFunctions(map<string, pZeroParFunc>, map<string, pOneParFunc>, map<string, pTwoParFunc>) { };
  virtual void initAtIteration() {};
  virtual void setBinning(map<string,vector<double> > dsBins){ *_dsBins = dsBins; } ;
  virtual int compute(valarray<double> &val, map<string, valarray<double> > &err) = 0;
 protected:

  virtual int parseOptions() { return 0;};
  double (*alpha_S)(double *);

...
};
```

Code organization

- Store modules in `reactions/NAME` subdirectories.
- Relations between name and shared library is given in `Reactions.txt` file
- Script to automatically generate templates for the source code, Makefile, ...:

```
glazov@padushka3:~/xfitter$ python tools/AddReaction.py MyModule
Creating directories in reactions/MyModule
Creating header file  reactions/MyModule/include/ReactionMyModule.h
Creating source file  reactions/MyModule/src/ReactionMyModule.cc
Creating source file  reactions/MyModule/src/Makefile.am
Update configure.ac file
Update Makefile.am
```

Examples of auto-generated files

- `Reactions.txt`:

```
fastNLO libfastnlo_xfitter.so
MyModule libmymodule_xfitter.so
```

- `ReactionMyModule.h`:

```
class ReactionMyModule : public ReactionTheory
{
  public:
    ReactionMyModule(){};
  public:
    virtual string getReactionName() const { return  "MyModule" ;};
    void initAtStart(const string &);
    virtual int compute(valarray<double> &val, map<string, valarray<double> > &err);
  protected:
    virtual int parseOptions(){ return 0;};
};
```

- `ReactioMyModule.src`:

```
#include "ReactionMyModule.h"
...
// the class factories
extern "C" ReactionMyModule* create() {
  return new ReactionMyModule();
}
...
```

Test run

- The auto-generated files are ready to be compiled:

```
make install
...
  CXX      ReactionMyModule.lo
  CXXLD    libmymodule_xfitter.la
..
```

- Use `.. 1506.06042/HERA1+2_NCep_920-thexp.dat` file as a template:

```
  TheoryType = 'expression'
  TermName = 'R'
  TermType = 'reaction'
  TermSource = 'MyModule'
  TermInfo = ''
  TheorExpr = 'R'
```

- Run xfitter, you should see “MyModule” printed before the data file is loaded.
- If you are done with tests and want to remove the test module, revert to the git head:

```
git checkout – Reactions.txt Makefile.am configure.ac
```

Further steps and other discussions

- While the main infrastructure is in place, a few things require implementation: pointers to α_S, PDFs, data are not yet in place. Plan to implement in near future
- APPLGRID, FastNLO and other “simple” modules such as `Hathor` should be easy to transfer to the new interface.
- For DIS, some further ideas are needed: basic building blocks are SF (F_2, F_L, ...) but theory expression mechanism does not foresee complex formula.
- Ideas for improving global HF steering, e.g. by process and dataset name:

```
DIS_Schemes = 'NC_DIS = RTOPT',
              'CC_DIS = ZMVFNS', 'H1cc = ACOT'
```