Improving theory interfaces in xFitter

S. Glazov, xFitter workshop, Dubna, 20 Feb 2016

Current implementation

- Fortran-based implementation
- Theory modules are activated for given data set using REACTION variable in the data namelist, e.g. REACTION = ’NC PP’
- All modules are called at several stages: initialization; at the beginning of iteration; and computation of predictions for a data set.
- For adding new theory modules, corresponding functions need to be modified. They already contain many if-statements.

Theory expression user interface

We already have a flexible way to compute predictions based on APPLGRID, FastNLO and k-factor corrections, using `expression` interface

```
TheoryType     = 'expression'
 TermName = 'A1', 'K1', 'K2', 'K3'
 TermType = 'applgrid','kfactor', 'kfactor',  'kfactor'
 TermSource = 'datafiles/lhc/atlas/drellYan/1404.1212/low_fidu.root' ,
              'datafiles/lhc/atlas/drellYan/1404.1212/kf.nominal.NNLO-NLOEW.txt',
              'datafiles/lhc/atlas/drellYan/1404.1212/kf.nominal.PI.txt'
              'datafiles/lhc/atlas/drellYan/1404.1212/kf.nominal.scheme.txt'
 TheorExpr= 'K1*K3*A1/1.e5+K2'
```

At the moment expressions are limited to APPLGRID, FastNLO and k-Factors. Would be great to extend to all theory modules, including DIS.

ITheoryModule.h

```
class ITheoryModule   {
 public:
  ITheoryModule(std::string name)
    { fName = name; }
  virtual ~ITheoryModule();
  virtual void InitializeModule() {}
  virtual void AddDataSet(int IDDataSet) {}
  virtual void InitBeforeInteration( const abstractParameters* pars, int* const iFlag )
  virtual void GetPrediction(int IDDataSet, double* prediction) = 0;
  virtual void AfterIteration() {}
  virtual void FinalizeModule() {}
 private:
  std::string name;
}
;
static std::map <std::string, ITheoryModule*>  theoryModulesRegistry;
```

- Interface class to register theory modules with a pure virtual GetPrediction method.
- Registry of theory modules by name in a map.
- Theory modules can become a part of Theory Expression, using the same name.

Things to consider

- Single instance theory module for multiple data set of the same type (easier fortran interface). Data sets are registered using AddDataSet function. Need to figure out simplest exchange of binning information, could be perhaps done by a helper function (or method), based on IDDataSet.
- Allow theory modules to have individual input parameters, not always namelist driven (?)
- Move to shared libraries for theory modules, dynamically loadable using common python interface (?)

In case of fire

1. git commit
2. git push
3. leave building