Recent Developments of the fastNLO Toolkit

Daniel Britzger, Klaus Rabbertz, Georg Sieber,
Fred Stober, Markus Wobisch

Karlsruher Institut für Technologie

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

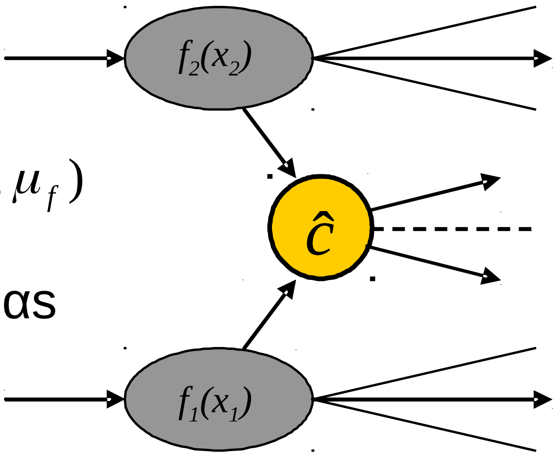
Motivation

- Calculation of hadron-hadron collisions at higher orders of perturbative QCD require large amounts of processing power
- Calculations are repeated many times for different parameters.
  - Multiple PDF sets (CT, MSTW, NNPDF, …)
  - Single PDF set: PDF uncertainties
  - Scale uncertainties
  - Used in fitting procedures
- The fastNLO framework allows to quickly evaluate the cross section for different PDFs, values of αs and scale choices

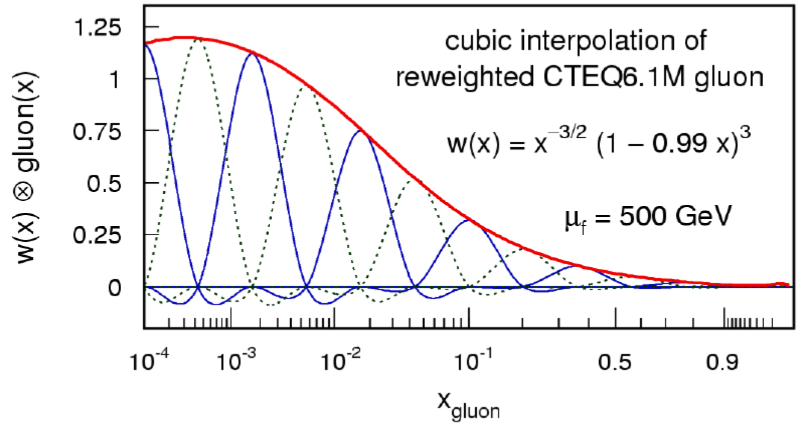
Introduction

- Cross section in hadron-hadron collisions in pQCD

$$\sigma = \sum_{a,b,n} \int_0^1 dx_1 \int_0^1 dx_2 \alpha_s^n(\mu_r) \cdot c_{a,b,n}(x_1, x_2, \mu_r, \mu_f) \cdot f_{1,a}(x_1, \mu_f) f_{2,b}(x_2, \mu_f)$$

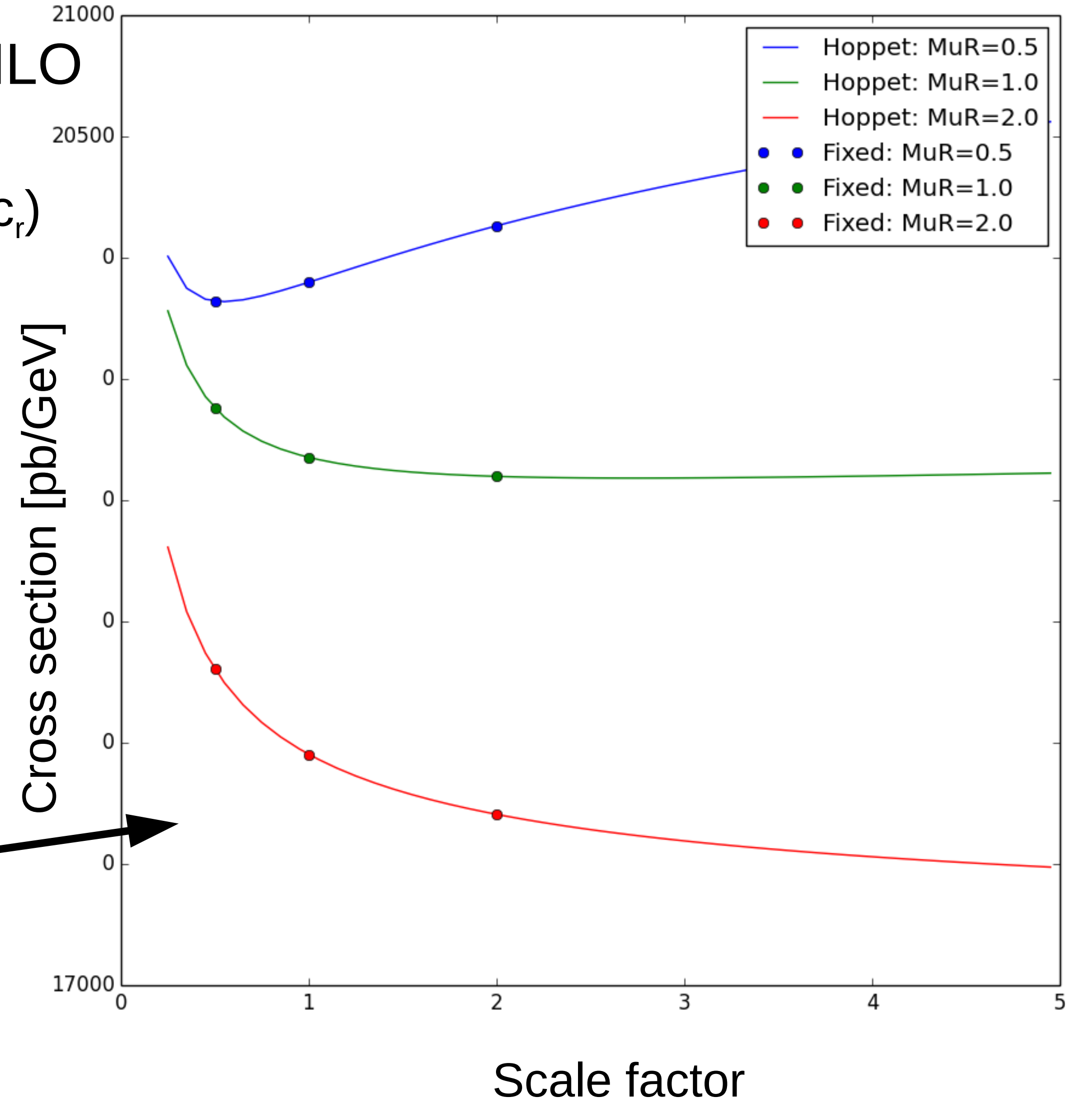


- Perturbative coefficients are independent from PDF and αs
- Factorize PDF, αs and scale dependence
- Interpolation
  - Single PDF is decomposed into interpolation kernels
  - Similar interpolation procedure also used for scales
- Convolution integrals become discrete sums
  - Values of perturbative coefficients can be stored in a table
  - Interpolation nodes in x and scales are stored together in look-up table



Scale variations for NLO

- Renormalization scale variations at NLO
  - Standard method:
    RGE → LO matrix element times $n\beta_0 \ln(c_r)$
  - Flexible-scale:
    using scale-independent weights
    $\omega(\mu_R,\mu_F) = \omega_0 + \log(\mu_R)\,\omega_R + \log(\mu_F)\,\omega_F$
- Factorization scale variations at NLO
  - Store coefficient for given set of scale factors
  - Flexible-scale
  - Calculate LO DGLAP splitting functions with HOPPET



Scale variations for NNLO

- In General: NLO splitting functions are needed for factorization scale variations → slow
- Flexible scale: analogous to **NLO** + **NNLO** terms using scale-independent weights → additional weights

  $$\omega(\mu_R,\mu_F) = \omega_0 + \log(\mu_R)\,\omega_R + \log(\mu_F)\,\omega_F + \log^2(\mu_R^2)\,\omega_{RR} + \log^2(\mu_F^2)\,\omega_{FF} + \log(\mu_R^2)\log(\mu_F^2)\,\omega_{RF}$$

- Advantages
  - Vary mR, mF independently and by any factor
  - NLO splitting functions are not needed
- fastNLO implementation:
  - Two different observables can be used for the scales
  - Any function of these two observables can be used for calculating scales

fastNLO Toolkit

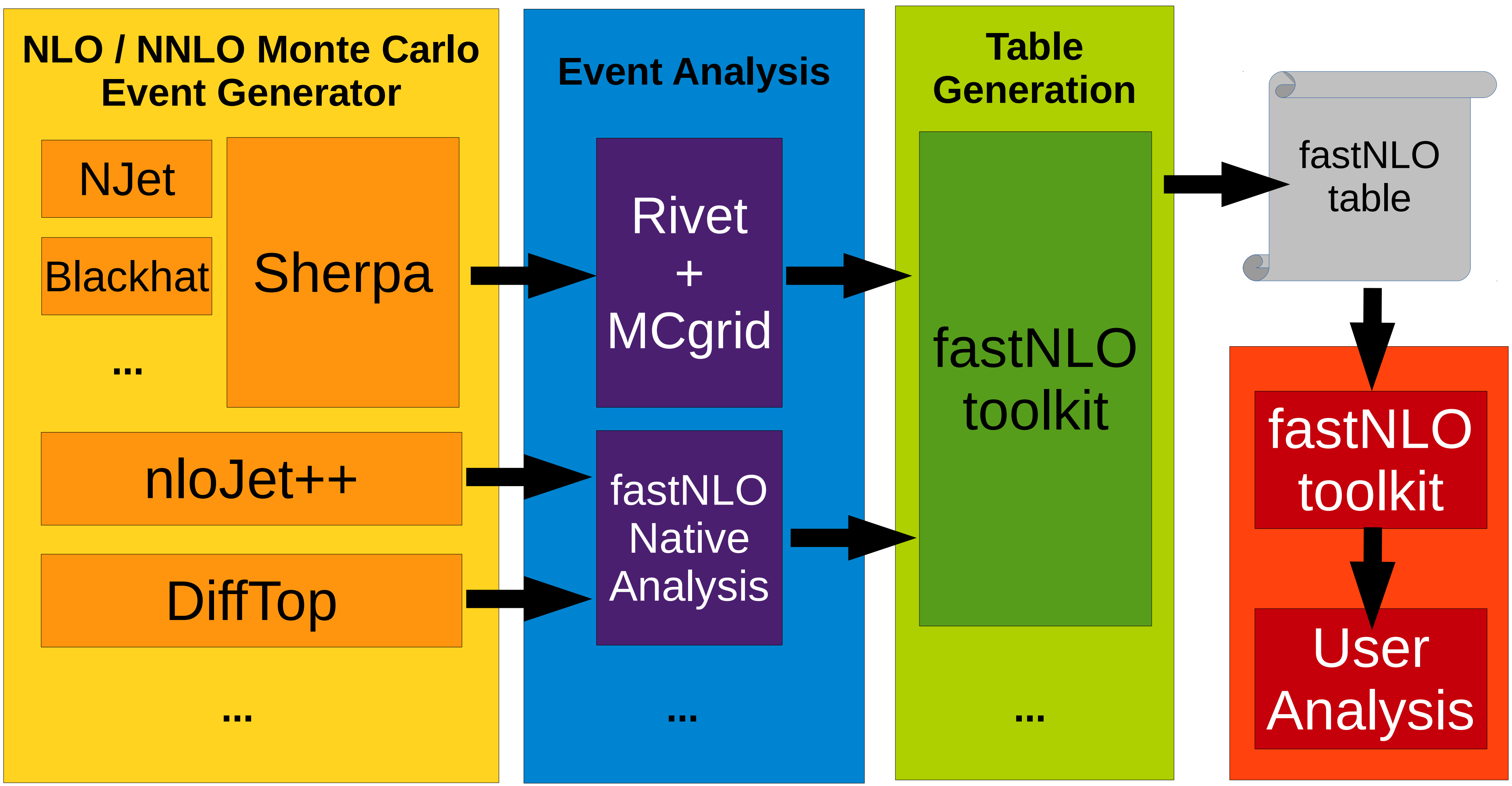
- Easy integration into NLO / NNLO Programs


```
fastNLOCreate fnlo(“steering.str“);
fnlo.SetOrderOfCalculation(int order)

fnlo.fEvent.SetProcessID(int id);
fnlo.fEvent.SetWeight(double w);
fnlo.fEvent.SetX1(double x1);
fnlo.fEvent.SetX2(double x2);
fnlo.fScenario.SetObservable0(double pt);
fnlo.fScenario.SetObsScale1(double s1);
fnlo.Fill();

fnlo.SetNumberOfEvents(double nevents);
fnlo.WriteTable();
```

Analysis Workflow

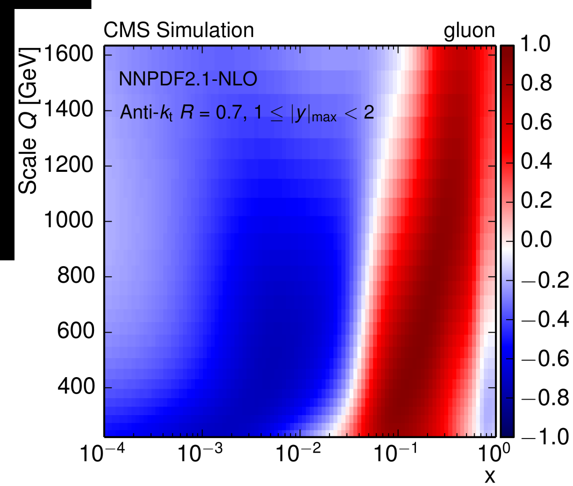


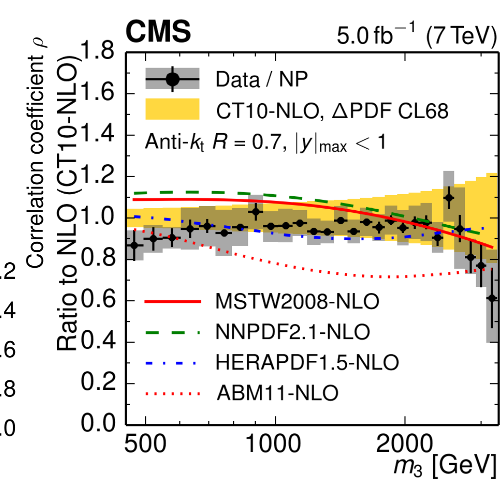
Reading tables

```
from fastnlo import fastNLOLHAPDF
import numpy
fnlo = fastNLOLHAPDF('fnlotable.tab')
fnlo.SetLHAPDFFilename('CT10nlo.LHgrid')
fnlo.SetLHAPDFMember(0)
mufs = np.arange(0.1, 1.5, 0.10)
murs = np.arange(0.1, 1.5, 0.10)
xs = np.zeros((mufs.size, murs.size))
for i, muf in enumerate(mufs):
  for j, mur in enumerate(murs):
    fnlo.SetScaleFactorsMuRMuF(mur, muf)
    fnlo.CalcCrossSection()
    xs[i][j] =
np.array(fnlo.GetCrossSection())[0]
```

- Access Tables with Python or C++
- Generate yoda output with `fnlo-tk-yodaout` (with contributions by Stefanos Tyros)

  Foreseen to be used in MCplots.
- LHAPDF 5 / 6
- Different αs Evolution codes available

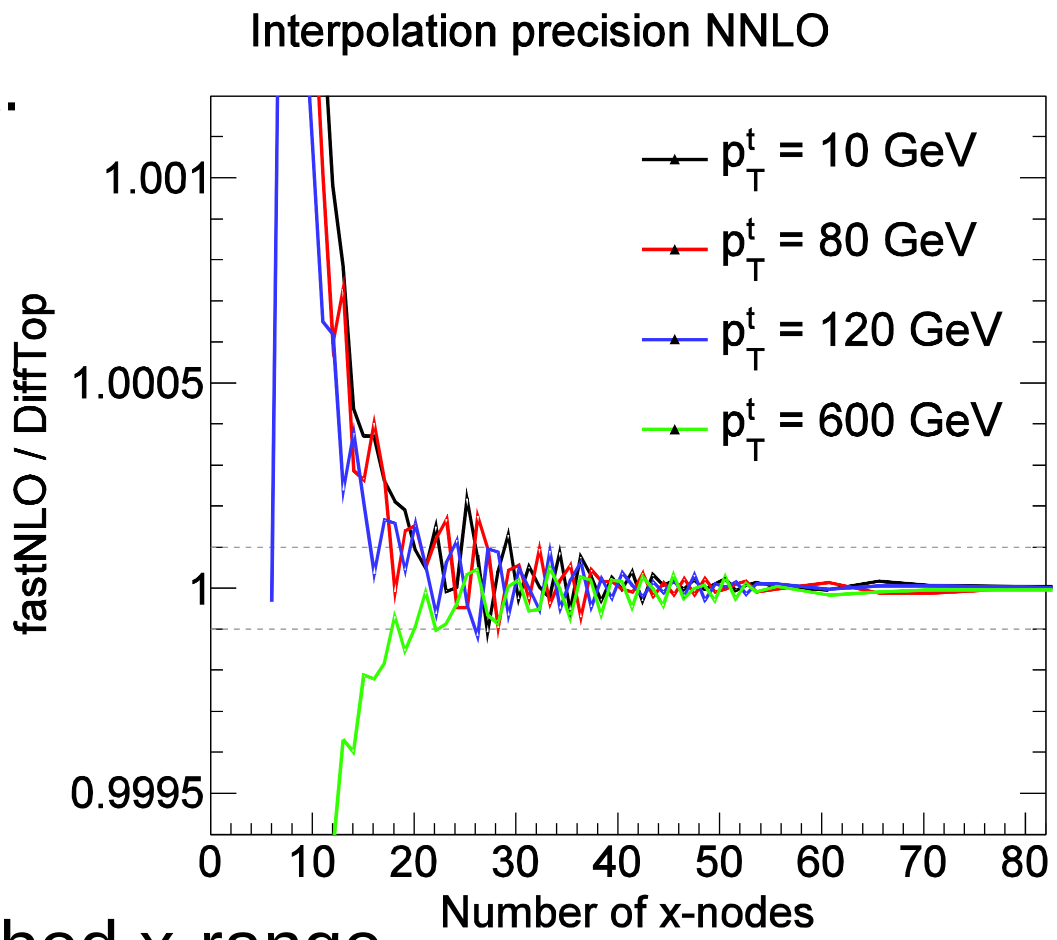




Example – DiffTop analysis: NNLO Accuracy with fastNLO

- Differential ttbar in approx. NNLO: dσ/dpT , dσ/dy
- Uncertainties:
  - PDF, scale, αs, m_t

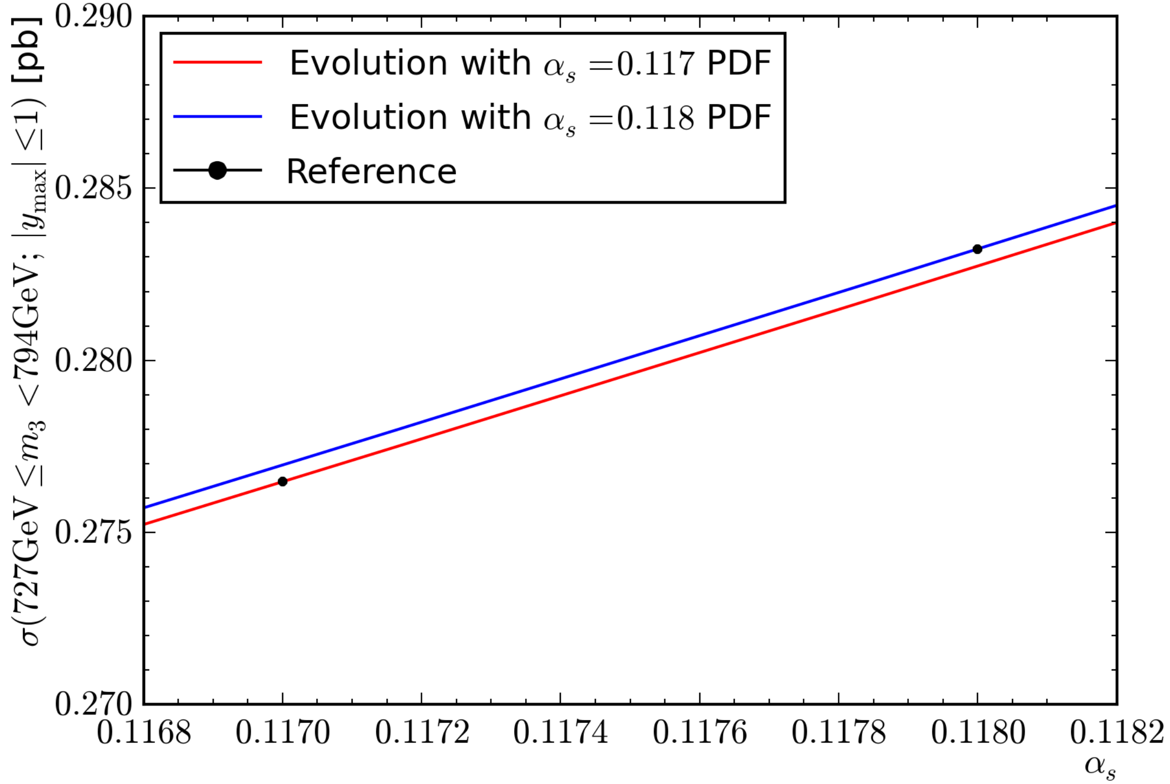
    786 variations needed
- Precision study of fastNLO tables over standalone DiffTop vs. no. of x nodes



- Perfect agreement for probed x-range of 2 ·10-3 < x < 1

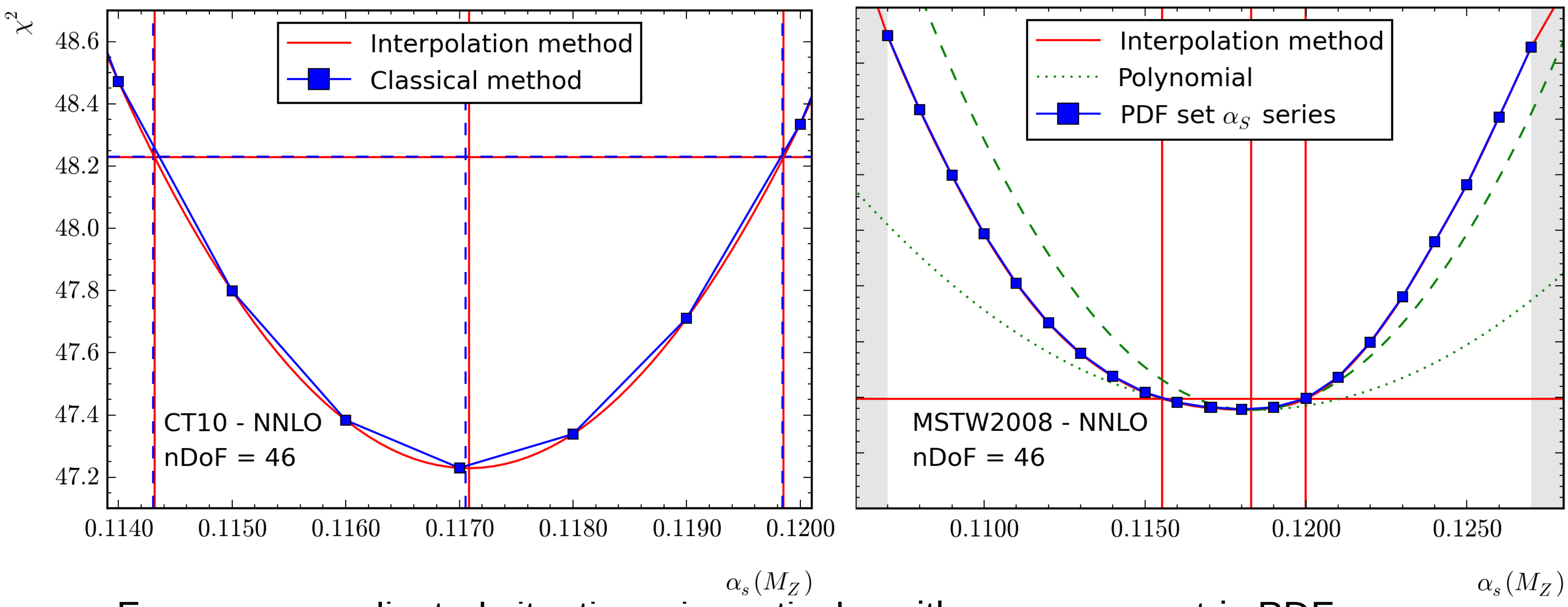
Precision measurement of $\alpha_S(M_Z)$: Three-Jet Mass Cross Section - Interpolation

- fastNLO allows to use user defined α_S evolution codes during the evaluation of the coefficient tables
- Not limited to the α_S values supplied by the PDF set anymore
  - This enables the use of proper minimizer libraries like Minuit2 when fitting α_S
- Two common :
  - Interpolation:
    - Interpolate between two adjacent α_S points
  - Extrapolation:
    - Some observables with large scale uncertainties can reach the edge of the α_S series of the PDF.



Precision measurement of $\alpha_S(M_Z)$: Three-Jet Mass Cross Section - Interpolation

- For simple cases, there is good agreement between the interpolation method and the classical parameterization method



- For more complicated situations, in particular with very asymmetric PDF uncertainties, the interpolation method avoids all parameterization related issues

New public release

- fastNLO toolkit
  - Support for MCgrid (> v1.2) and Sherpa (including all NLO processes available via this chain)
  - fnlo-tk-yodaout (with contributions by Stefanos Tyros) for YODA-formatted output
  - Triple differential observable Binnings are supported
  - Convenience functions to derive scale uncertainties
  - fnlo-tk-config executable is provided for easier compiling/linking
  - Doxygen documentation
- fastNLO interface to NLOJet++
  - Updated to work with new toolkit prerelease
  - Major simplification in NLOJet++ usage – many use cases possible without code changes
    - InclusiveNJets - one entry per jet of an event
    - InlusiveNJetsEvent - one entry per event.
  - Steering files to reproduce calculations for numerous published cross sections
- fastNLO reader (compatibility update)
  - Adapted to new fastNLO_toolkit
  - Up to 3-dimensional binnings are supported
  - 7 subprocesses for LO hh->jets as used in toolkit is supported
  - Internally used cross section units are rescaled to published units

Summary and Outlook

- **New public release of the fastNLO toolkit this week**
- Toolkit provides all necessary functions to **create** and **evaluate** fast interpolation tables in the fastNLO format
- Number of interfaced NLO / NNLO theory programs steadily increasing (via MCgrid or direct interface)
- Fast and accurate evaluation of NLO / NNLO calculations with flexible scales / different as evolutions open up new possibilities for analysis
- Large scale production effort underway to produce tables for published results – will be available on

fastnlo.hepforge.org