# fastNLO Toolkit

Daniel Britzger, **Klaus Rabbertz,** Georg Sieber, Fred Stober, Markus Wobisch
(DESY, KIT * 3, Louisiana Tech University)

## in cooperation with

Enrico Bothmann, Steffen Schumann
(Uni Göttingen)

Karlsruhe Institute of Technology

# Part 1

# fastNLO v2 & Toolkit Development

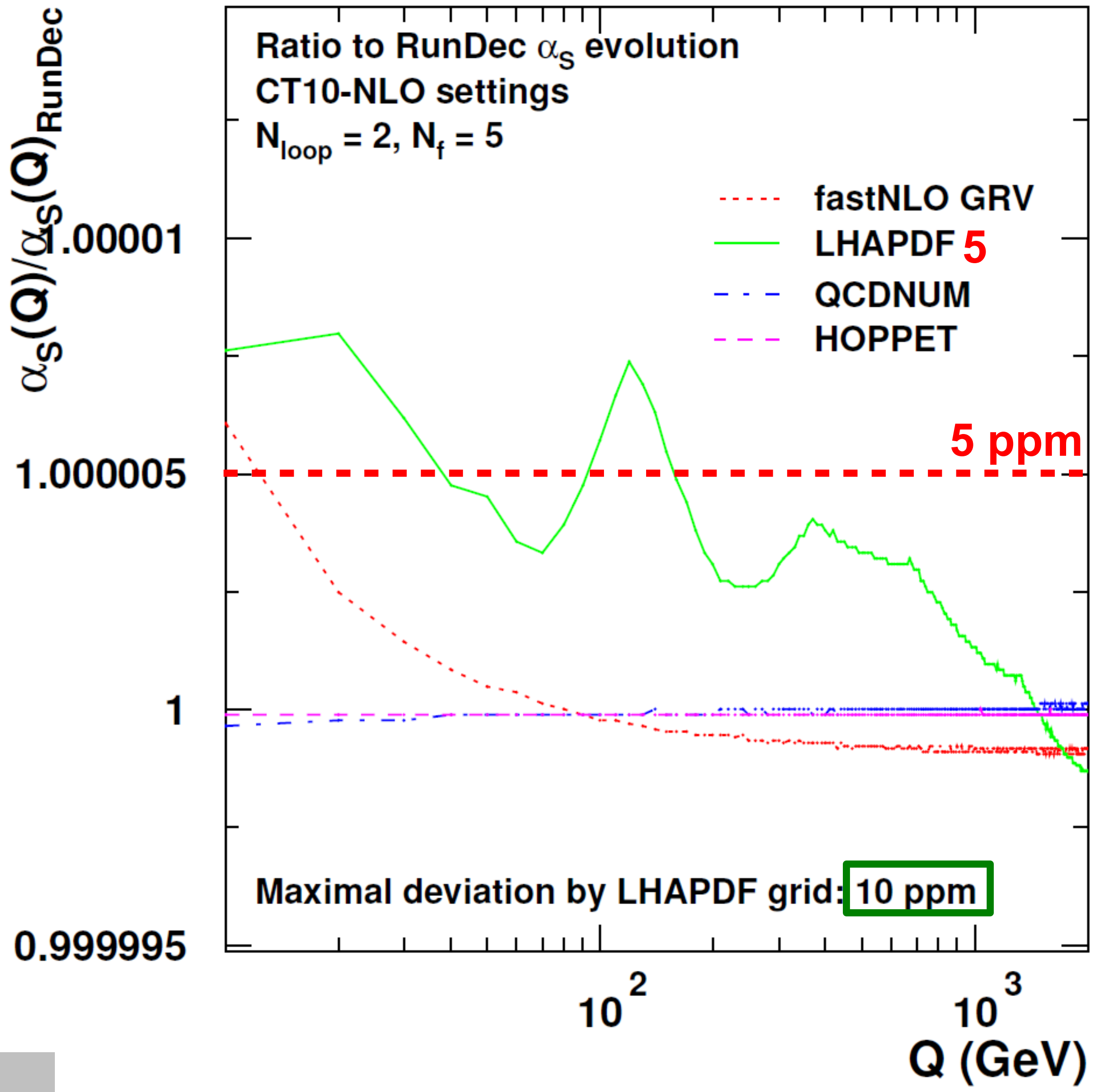
# From v1.4 to v2.1

- 1. Cross-checked old v1.4 versus new v2.1 tables
- 2. Converted existing v1.4 tables to new format
- 3. Cross-checked new table reader code in C++ vs. Fortran
- 4. Public release of table reader code as autotools tarball:
  - First release 14.02.2012: fastNLO_reader 2.1.0-1062
- 5. Transformed C++ reader code into linkable library
  - Latest release 14.02.2014: fastNLO_reader 2.1.0-1689
- Installation:
  - Requirements: LHAPDF5 or 6
  - `./configure --prefix=/path/to/install/directory [--with-lhapdf=/path]`
  - `fnlo-tk-config available to list config info and compiler/linker options`
- 6. Implemented new functionalities …

LHAPDF, M. Whalley et al., hep-ph/0508110,
LHAPDF6, A. Buckley et al., arXiv:1405.1067.

# Use of alternative $\alpha_s$ evolutions

- **LHAPDF5/6**
- **CRunDec 08/2012**
  - included in fastNLO
- **QCDNUM v17-00-06**
  - … `[--with-qcdnum=/path/...]`
  - `Makefiles adapted, need -fPIC on x86_64 systems`
- **HOPPET v1.1.5**
  - … `[--with-hoppet=/path/...]`

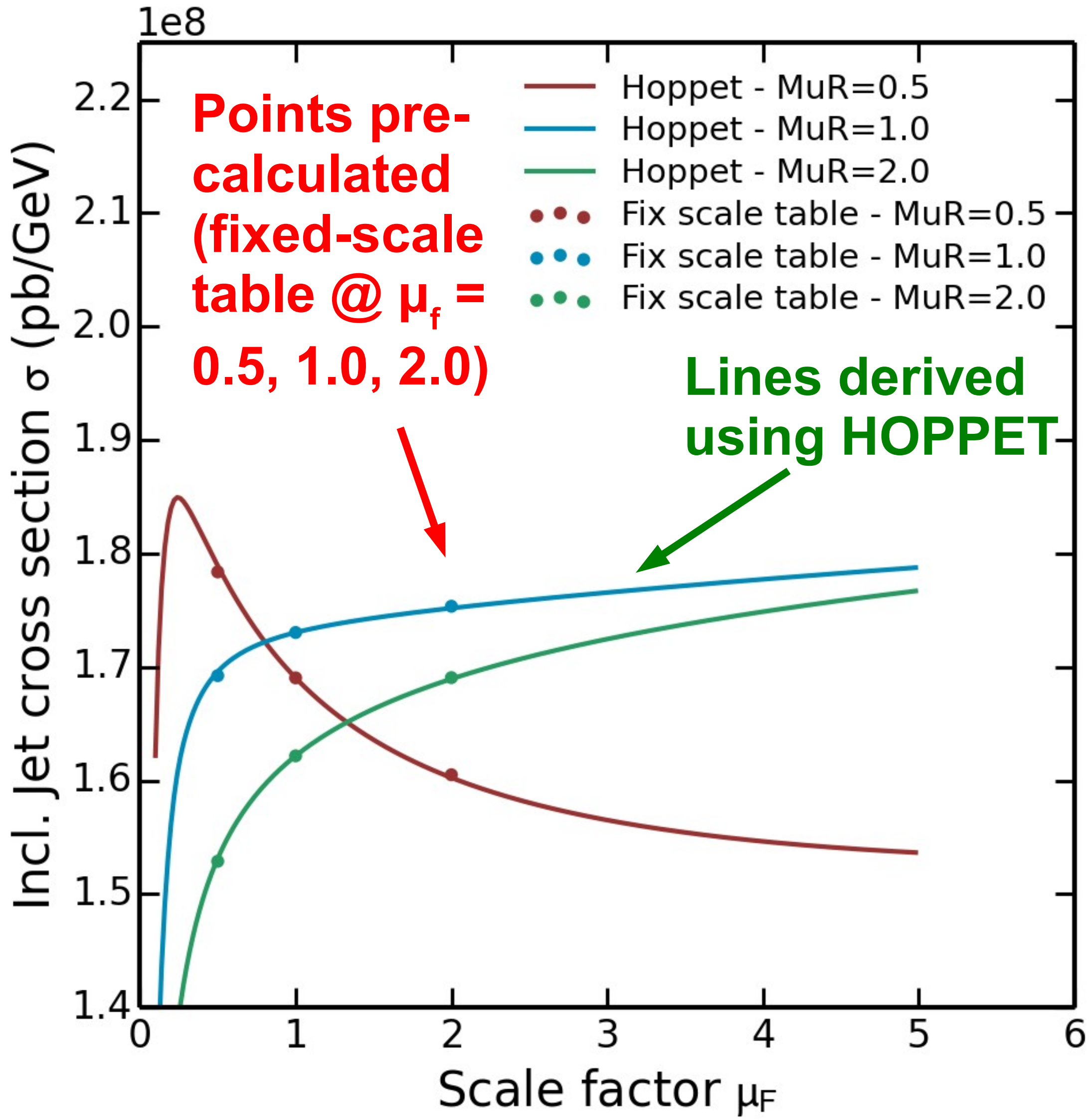



RunDec, B. Schmidt, M. Steinhauser, CPC183, 2012;
K. Chetyrkin, J. Kühn, M. Steinhauser, CPC133, 2000.
QCDNUM, M. Botje, CPC182, 2011.
HOPPET, G. Salam, J. Rojo, CPC180, 2009.

# Use of HOPPET for $\mu_f$ variation

- **fastNLO v1.4 used extra tables for $\mu_f$ variation with fixed scale factors**
  - **straightforward also @ NNLO**
  - **avoids additional integrations**
  - **increases table size**
- **In fastNLO v2.3 can also use HOPPET for $\mu_f$ variation**
  - **Same method as used in APPLgrid**
  - **continuous curve**

APPLgrid, T. Carli et al., EPJC 66 (2010) 503.



# Calculations with fastNLO in NNLO

- Problem
  - Scale variations become more difficult in NNLO than in NLO
- Current available implementations for NLO calculations

Renormalization scale variations
  - Scale variations applying RGE
    - Use LO matrix elements times $n\beta_0\ln(c_r)$ **[fastNLO, APPLgrid]**
  - Flexible-scale implementation
    - Store scale-independent weights: **[fastNLO]**

Factorization scale variations
  - Calculate LO DGLAP splitting functions using HOPPET **[APPLgrid, fastNLO]**
  - Store coefficients for desired scale factors **[fastNLO]**
  - Flexible-scale implementation **[fastNLO]**

- Scale variations for NNLO calculations
  - renormalization scale variations become more complicated
  - NLO splitting functions are needed for factorization scale variations e.g. with HOPPET
    - Calculations become slower again => Not desired for fast repeated calculations

# Flexible-scale tables

- Storage of scale-independent weights enable full scale flexibility also in NNLO
  - Additional logs in NNLO

$$\omega(\mu_R,\mu_F) = \underbrace{\omega_0 + \log(\mu_R^2)\omega_R + \log(\mu_F^2)\omega_F}_{\text{log's for NLO}} + \underbrace{\log^2(\mu_R^2)\omega_{RR} + \log^2(\mu_F^2)\omega_{FF} + \log(\mu_R^2)\log(\mu_F^2)\omega_{RF}}_{\text{additional log's in NNLO}}$$

  - Store weights: $w_0$, $w_R$, $w_F$, $w_{RR}$, $w_{FF}$, $w_{RF}$ for order $\alpha_s^{n+2}$ contributions

- Advantages
  - Renormalization and factorization scale can be varied *independently* and by *any* factor
    - No time-consuming ‘re-calculation’ of splitting functions in NLO necessary
  - Only small increase in amount of stored coefficients

- fastNLO implementation
  - *Two* different observables can be used for the scales
    - e.g.: $H_T$ and $p_{T,max}$
    - or e.g.: $p_T$ and $|y|$
    - …
  - *Any function* of those *two observables* can be used for calculating scales

**‘Flexible-scale concept‘: Best choice for performant NNLO calculations**

# Flexible-scale tables in DIS

fastnlo @ HepForge

**Tables from recent H1 multi-jet study use $\sqrt{Q^2}$ and pT**

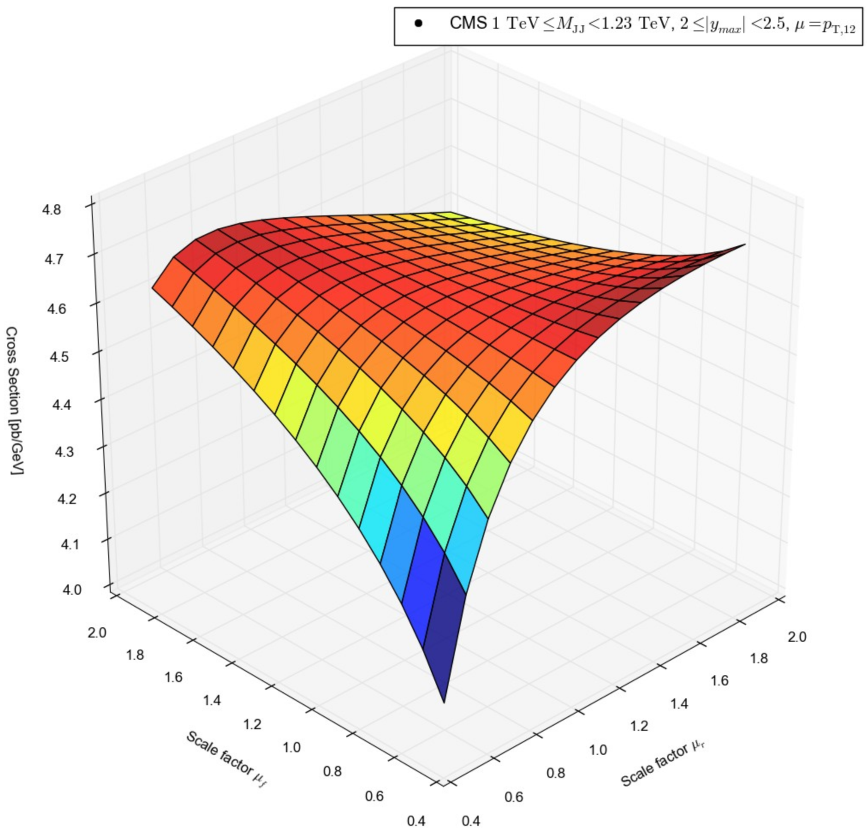
**Use of this method in fastNLO dates back to 2011 when going from v1.4 to v2.1. Useful for DIS, now also for pp, e.g. with scales $M_Z$ and $pT_Z$.**

Note: All HERA tables are flexible-scale tables ==> The C++ reader versions must be used.

| | | | |
|---|---|---|---|
| **HERA: ep @ sqrt(s) = 319 GeV** | | | |
| fnh5001_I1301218 | H1 inclusive jet HERA-II (kt and anti-kt); LO, NLO | | |
| | inSPIRE | no HepData yet | no RIVET analysis available |
| fnh5002_I1301218 | H1 dijet HERA-II (kt and anti-kt); LO, NLO | | |
| | inSPIRE | no HepData yet | no RIVET analysis available |
| fnh5003kt_I1301218 | H1 dijet HERA-II (kt); LO, NLO | | |
| | inSPIRE | no HepData yet | no RIVET analysis available |
| fnh5003ak_I1301218 | H1 dijet HERA-II (anti-kt); LO, NLO | | |
| | inSPIRE | no HepData yet | no RIVET analysis available |
| fnh4002_I875006 | ZEUS inclusive dijet HERA-I+II (kt); LO, NLO | | |
| | inSPIRE | no HepData | no RIVET analysis available |
| | (Note: This table only works with the new fastnlo_toolkit reader, but not yet with the old fastnlo_reader.) | | |
| fnh5201_I838435 | H1 inclusive jets at low Q^2 HERA-I (kt); LO, NLO | | |
| | inSPIRE | no HepData | no RIVET analysis available |
| | (Note: This table only works with the new fastnlo_toolkit reader, but not yet with the old fastnlo_reader.) | | |
| fnh5401_I818707 | H1 inclusive jets at high Q^2 HERA-I (kt); LO, NLO | | |
| | inSPIRE | no HepData, only normalized x section publ. | no RIVET analysis available |
| | (Note: This table only works with the new fastnlo_toolkit reader, but not yet with the old fastnlo_reader.) | | |
| fnh5101_I753951 | H1 inclusive jets HERA-I (kt); LO, NLO | | |
| | inSPIRE | HepData | no RIVET analysis available |
| | (Note: This table only works with the new fastnlo_toolkit reader, but not yet with the old fastnlo_reader.) | | |
| fnh4401_I724050 | ZEUS inclusive jets HERA-I (kt); LO, NLO | | |
| | inSPIRE | HepData | no RIVET analysis available |
| | (Note: This table only works with the new fastnlo_toolkit reader, but not yet with the old fastnlo_reader.) | | |
| **HERA: ep @ sqrt(s) = 300 GeV** | | | |
| fnh4301_I593409 | ZEUS inclusive jets HERA (kt); LO, NLO | | |
| | inSPIRE | HepData | no RIVET analysis available |
| | (Note: This table only works with the new fastnlo_toolkit reader, but not yet with the old fastnlo_reader.) | | |

# Demo plot using Python extension

- **Python extension available**
  - … [--enable-pyext]
- **Easy example plotting 2D scale dependence:**




**Setup Python with fastNLO**

```
#! /usr/bin/env python2

from fastnlo import fastNLOLHAPDF

import matplotlib
import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import axes3d
import numpy as np
```

**Select table, PDF & mem.**

```
fnlo = fastNLOLHAPDF('fnlotable.tab')
fnlo.SetLHAPDFFilename('CT10nlo.LHgrid')
fnlo.SetLHAPDFMember(0)
```

**Define $\mu_r$, $\mu_f$ ranges**

```
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))
```

**Loop over $\mu_r$, $\mu_f$**

```
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]
```

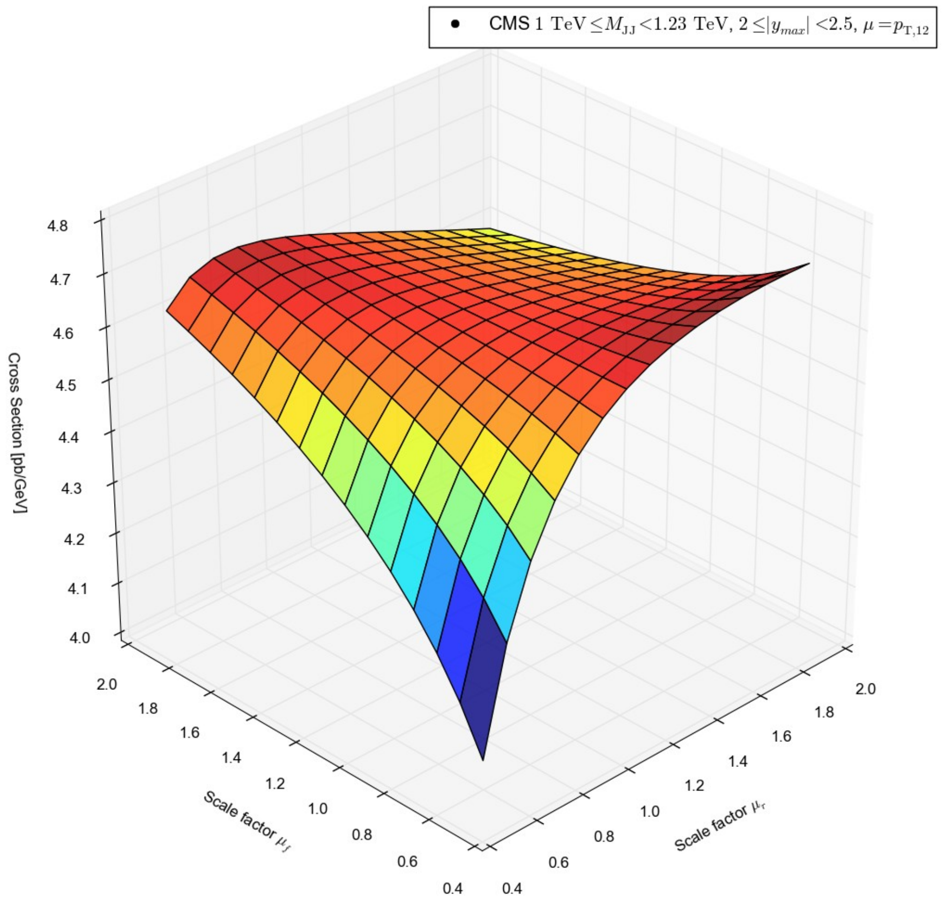
**Plot**

```
fig = plt.figure(figsize=(13,13))

… plotting details
ax.set_ylabel('Scale factor $\mu_F$')
ax.set_xlabel('Scale factor $\mu_R$')
ax.set_zlabel('Cross Section [pb/GeV]')
plt.show()
… plotting details
```

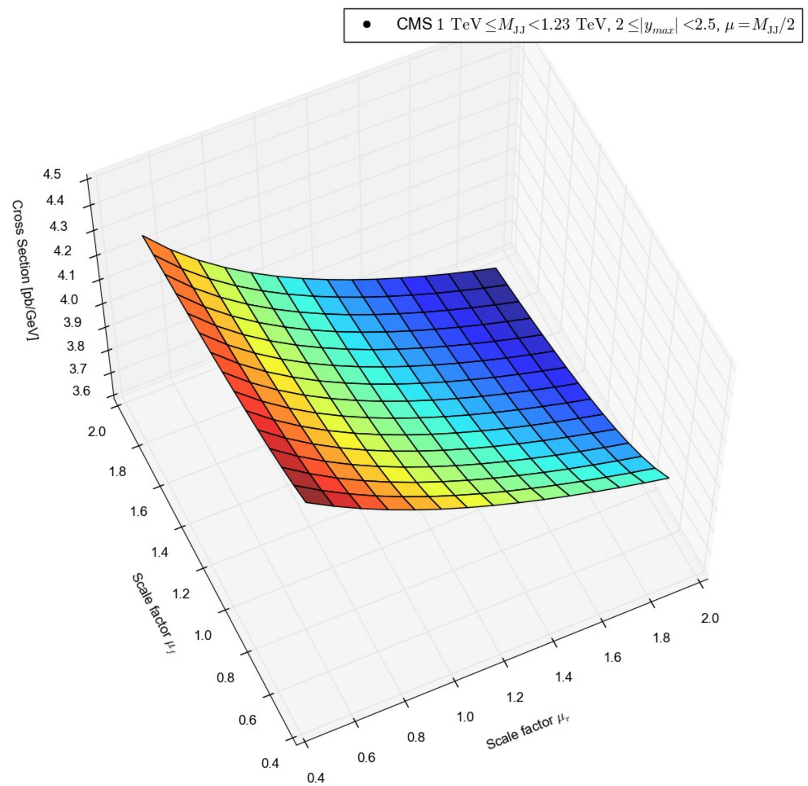
# Extra slide: CMS dijet mass

Central scale: $\mu = \langle pT_{1,2} \rangle$

Outer $|y_{max}|$ bin!




Central scale: $\mu = M_{JJ}/2$



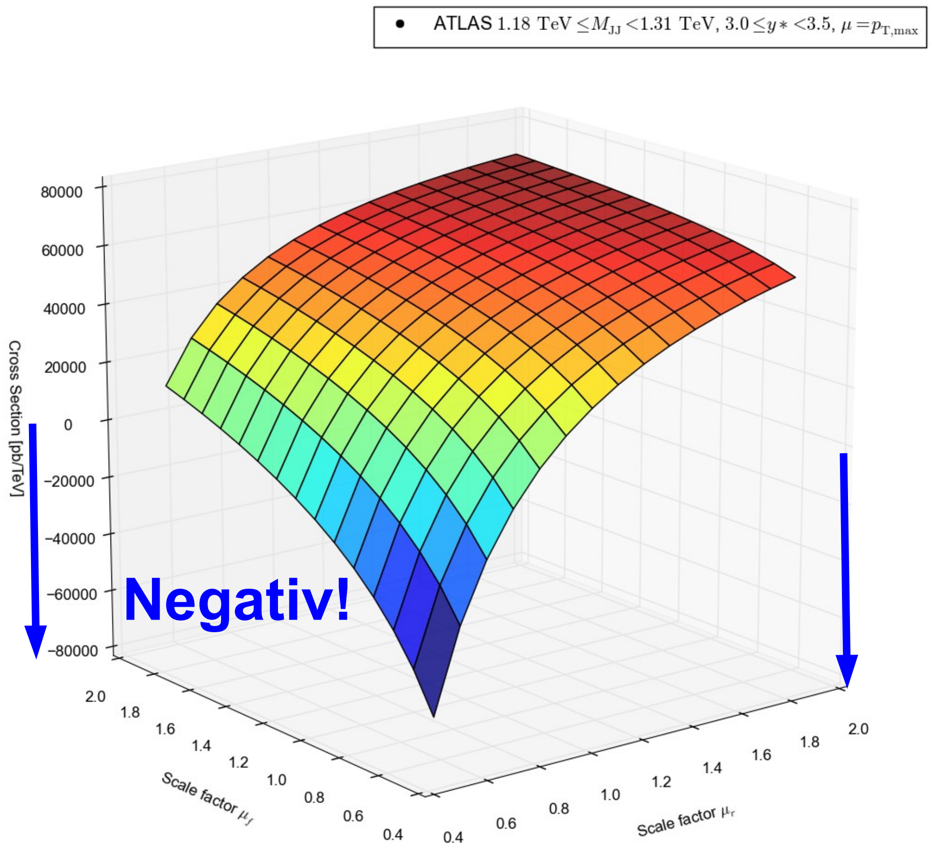

Derived from one fastNLO flexible-scale table

# Extra slide: ATLAS dijet mass

## Central scale: μ = $pT_{max}$

**Outer y* bin!**

+80k

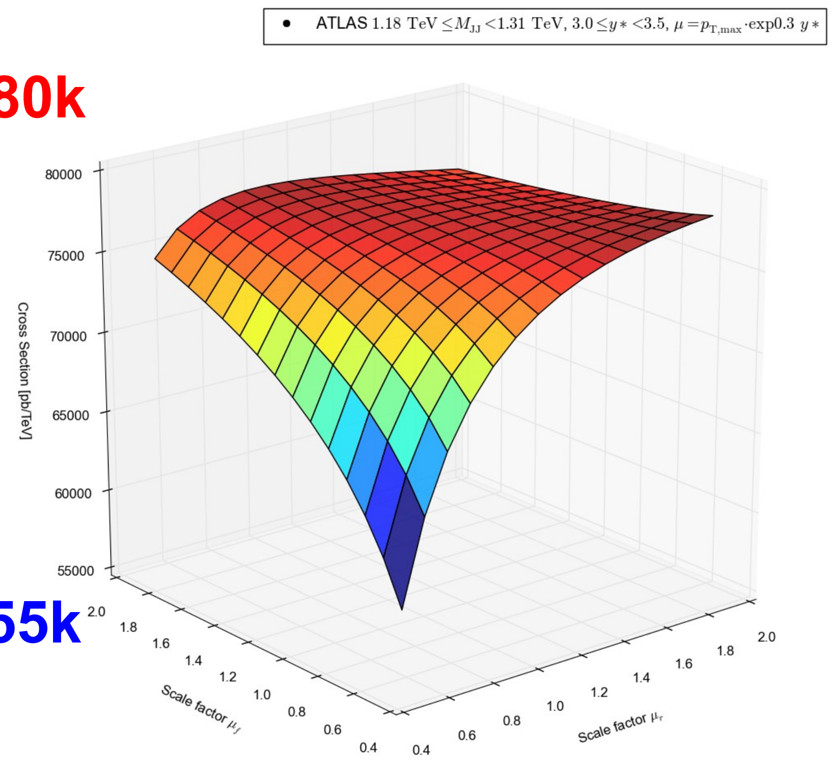



Negativ!

-80k

## Central scale: μ = $pT_{max}$ · exp(0.3 y*)

+80k



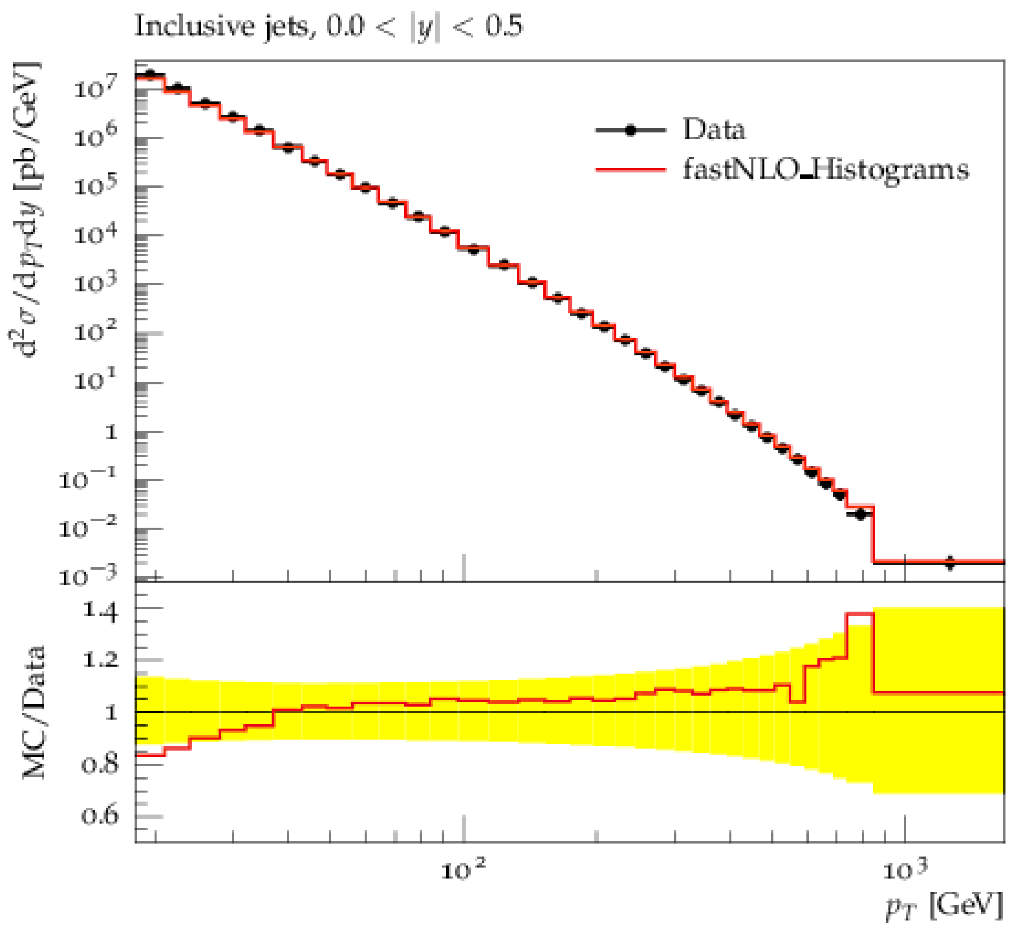

+55k

**Derived from one fastNLO flexible-scale table**

# Use with Rivet 2 & YODA Format

**Summer student project of Stefanos Tyros (with Peter Skands):**

- **Provide YODA formatted output for fastNLO tables**
  - `fnlo-tk-yodaout fnlotable.tab`
- **Compare with data (or MC) histograms using Rivet**
  - `rivet-mkhtml fnlotable.yoda`
  - `browser plots/index.html`
- **Can provide e.g. NLO plots to mcplots.cern.ch**
- **Test inclusion of fastNLO in GENSER successful**




**It is very desirable to have the RIVET analyses from the experiments!**

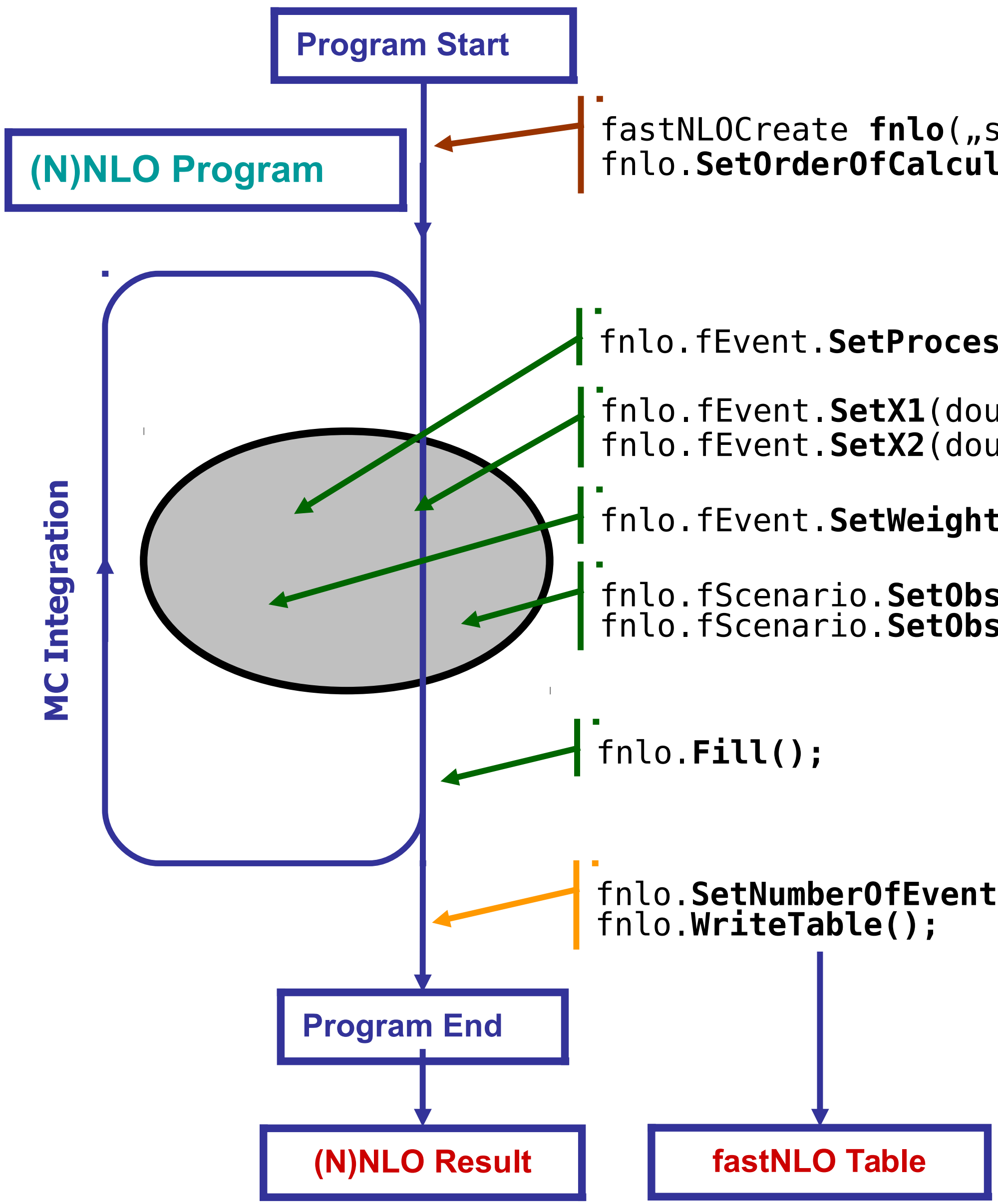
RIVET, A. Buckley et al., CPC184 (2013), rivet.hepforge.org, yoda.hepforge.org.

# Re-factoring v2.1 into Toolkit

- 1. Code split into:
- Toolkit library for creating & evaluating interpolation grids
  - Independent of any generator
  - First pre-release 17.07.2014: fastnlo_toolkit 2.3.1pre-1854
- Specific helper interfaces, if required, to N(N)LO programs
  - e.g. to use with NLOJet++: fastnlo_interface_nlojet 2.3.1pre-1855
- 2. Checked backwards compatibilty with v2.1
- 3. Facilitated use with extensible steering files
- 4. Interface other theory programs ...

NLOJet++, Z.Nagy,
PRD68 2003, PRL88 2002

# Simple example for use of Toolkit




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

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

fnlo.Fill();

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

**Initialize fastNLO class(es)**

**Pass the process specific variables during the ‘event loop’ to fastNLO**

- **Order does not matter**
- **Many other convenient implementations possible**

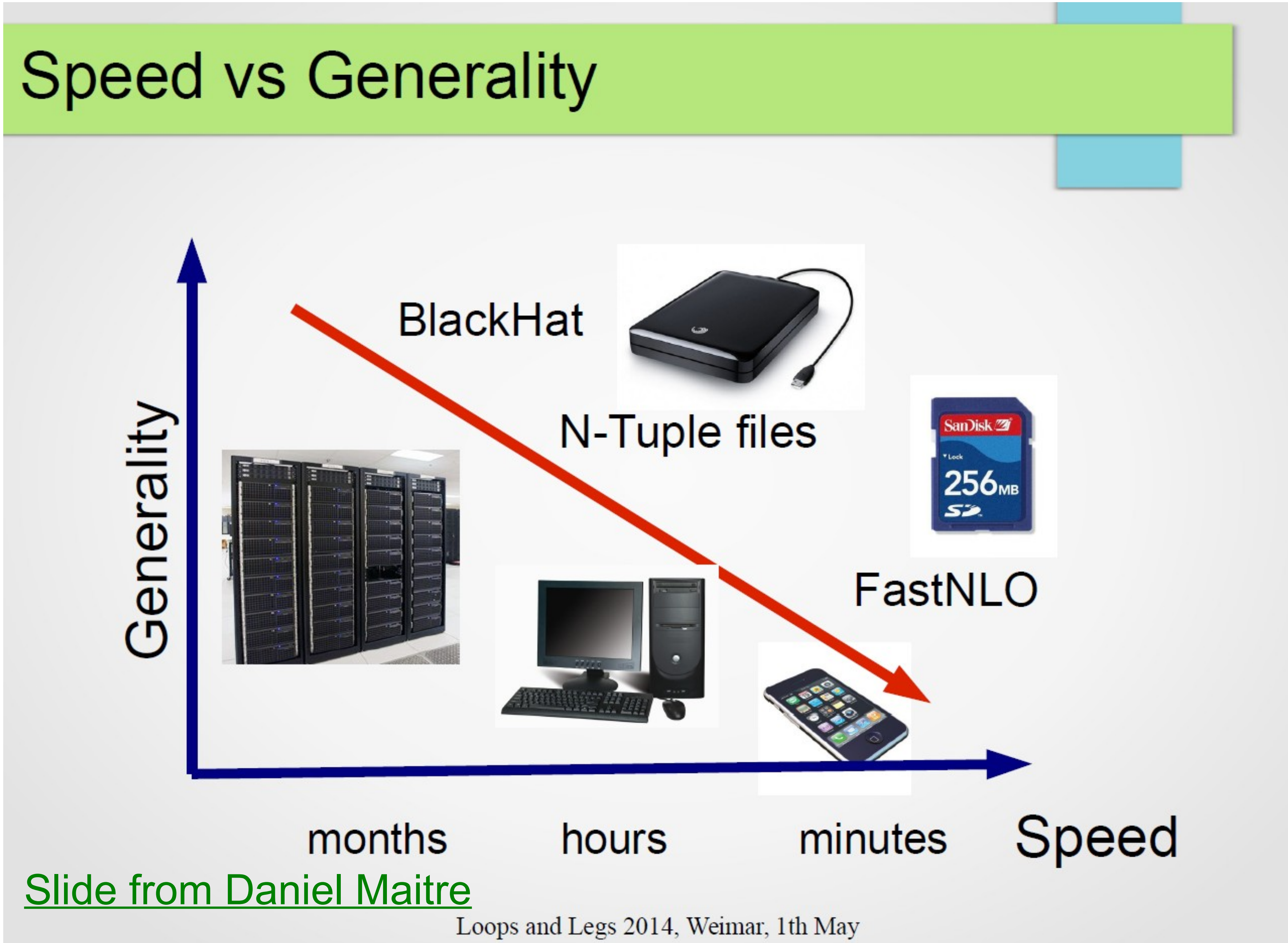
**Pass all information to fastNLO**

**Set normalization of the MC integration and write table**

**Minimum implementation: 11 lines of code**

**Convenient implementation of fastNLO into any (N)NLO program possible!**

# Use with BlackHat N-Tuples

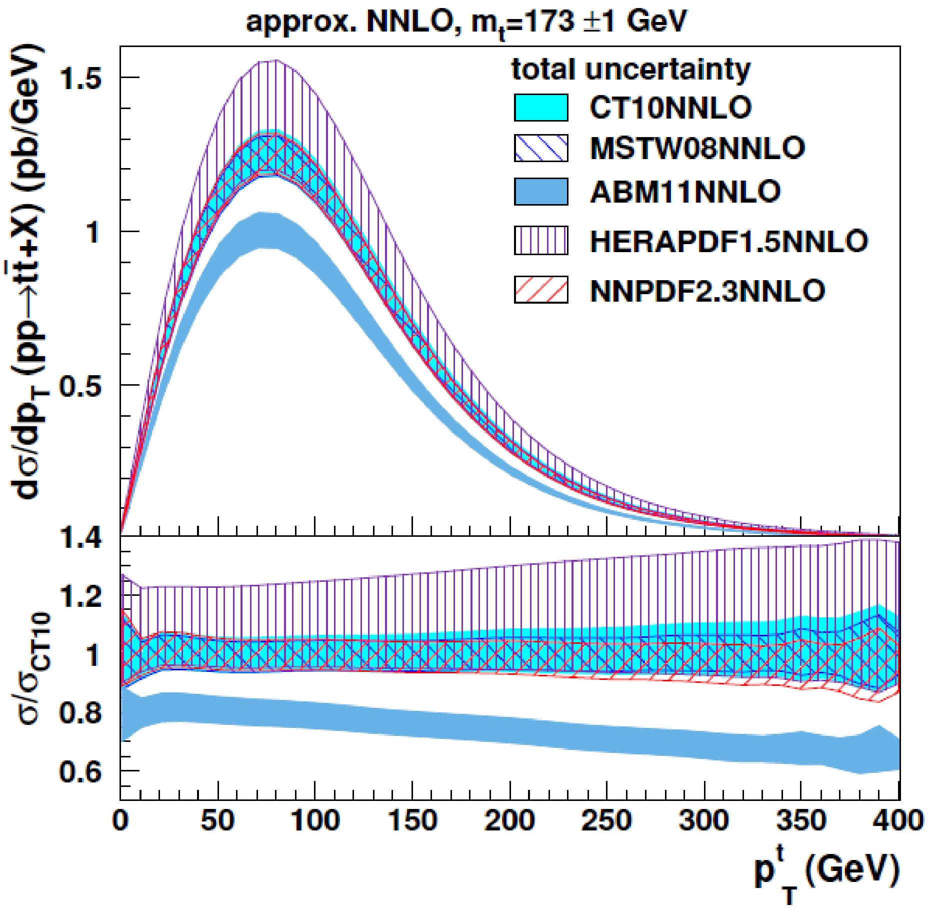



Slide from Daniel Maitre

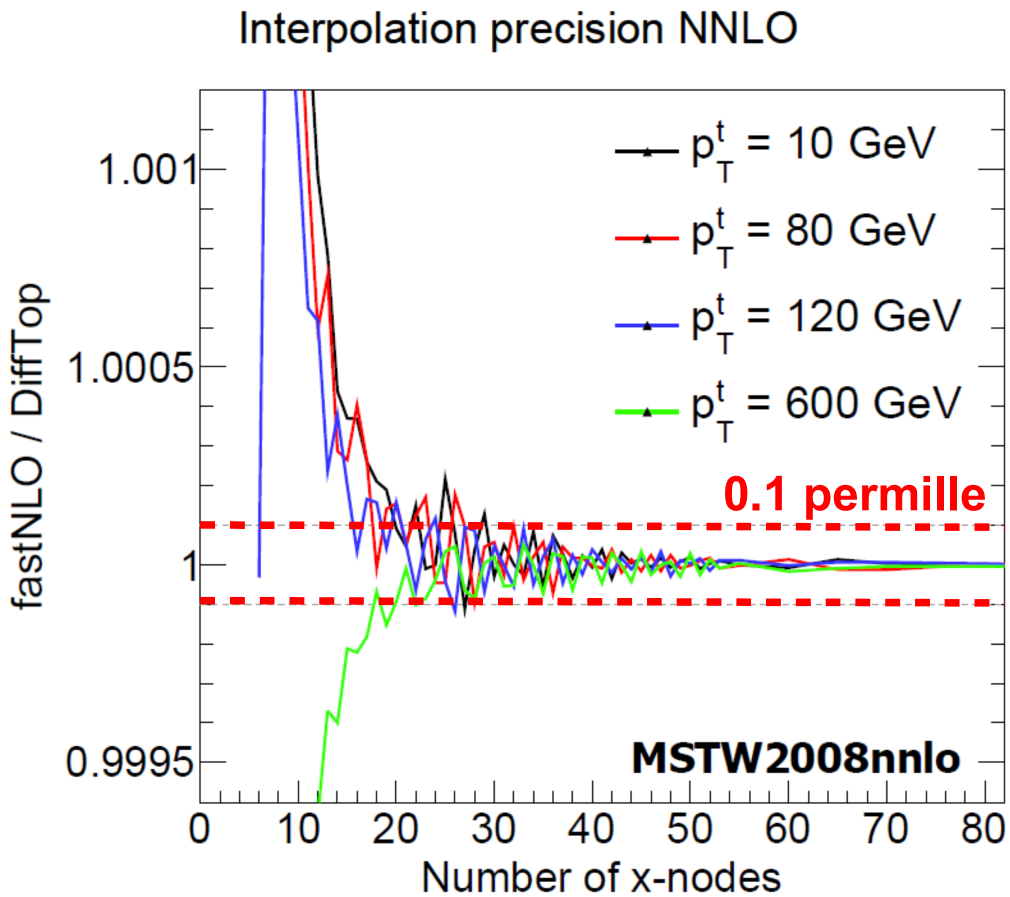
## Differential ttbar in approx. NNLO:

$d\sigma/dp_T$, $d\sigma/dy$

(total uncertainty: quadr. sum of PDF, scale, $\alpha_s$, $m_t$ variations)




## Precision study of fastNLO tables over DiffTop standalone vs. no. of x nodes




786 repeated calculations needed including (separate) variation of $m_t$

DiffTop, M. Guzzi et al., JHEP01, 2015.

Perfect agreement for probed x-range of $2 \cdot 10^{-3} < x < 1$

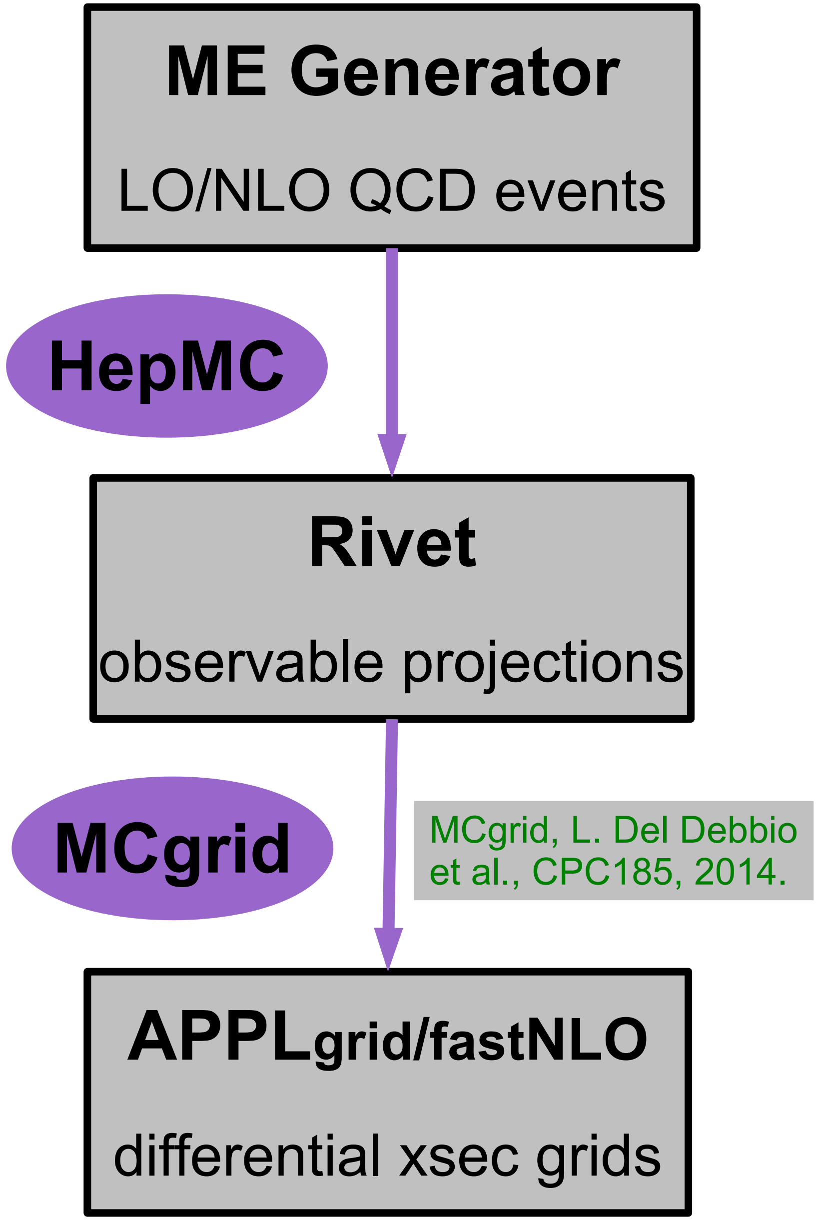
# Part 2

# New: Interface to Sherpa via MCgrid

**in collaboration with**
**Enrico Bothmann & Steffen Schumann**

# Use with Sherpa & MCgrid

- **fastNLO Toolkit access implemented:**
  - **Events generated with Sherpa 2.1.1**
  - **Two analyses from Rivet 2.2.0 tested**
  - **MCgrid 2.0 for cross section projection into grids (to be released)**
  - **Same toolkit functions accessed either via direct calls from MCgrid-enabled Rivet analysis or via steering file**
  - **Usable with large number of processes available via Sherpa and one-loop generators like BlackHat, GoSam, OpenLoops, NJET, ...**




Sherpa, T. Gleisberg et al., JHEP02, 2004; JHEP02, 2009.
BlackHat, C.F. Berger et al., PRD78, 2008.
GoSam, G. Cullen et al., EPJC72, 2012.
OpenLoops, F. Cascioli et al., PRL108, 2012.
NJET, S. Badger et al., CPC184, 2013.
...

# Snippets of Rivet+MCgrid analysis

```
#include "Rivet/Analysis.hh”
#include "mcgrid/mcgrid.hh"
...
namespace Rivet {

  /// CDF Z boson rapidity modified to generate grid files
  class MCgrid_CDF_2009_S8383952 : public Analysis {
  public:
...
using namespace MCgrid;
Histo1DPtr _hist_yZ;  // Rivet histogram
gridPtr _grid_yZ;     // Corresponding grid

// Init phase
subprocessConfig subproc("DY-ppbar.str", BEAM_PROTON, BEAM_ANTIPROTON);
fastnloGridArch arch(50, 1, "Lagrange", "OneNode", "sqrtlog10", "linear");
fastnloConfig config(0, subproc, arch, 1960.0);
_hist_yZ = bookHisto1D(2, 1, 1);                       // Book Rivet
_grid_yZ = bookGrid(_hist_yZ, histoDir(), config);     // Book MCgrid/fastNLO

// Analyse phase
PDFHandler::HandleEvent(event, histoDir());   // Update subprocess statistics
_hist_yZ->fill(yZ, weight);                   // Fill Rivet
_grid_yZ->fill(yZ, event);                    // Fill MCgrid/fastNLO

// Finalise phase
scale(_hist_yZ, normalisation);               // Scale Rivet
_grid_yZ->scale(normalisation);               // Scale MCgrid/fastNLO
PDFHandler::CheckOutAnalysis(histoDir());     // Finalise
```

**Setup Rivet with MCgrid**

**Basic scheme very similar with APPLgrid, differences in steering & functionalities.**

**Book & config grid and histos**

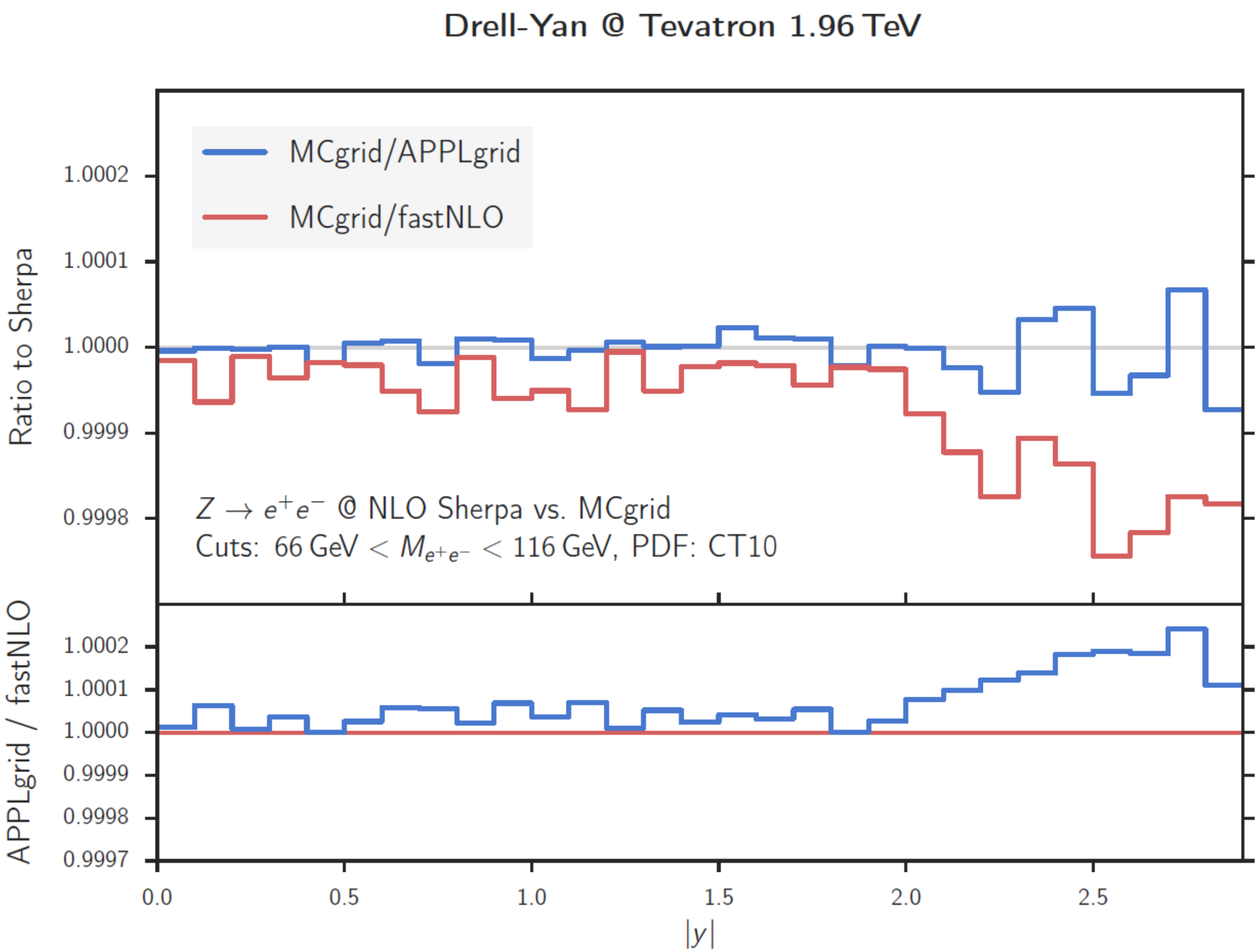
**Fill events in event loop.**

**Final check out, normalize, write table.**

# Drell-Yan @ Tevatron

## Drell-Yan Z rapidity:

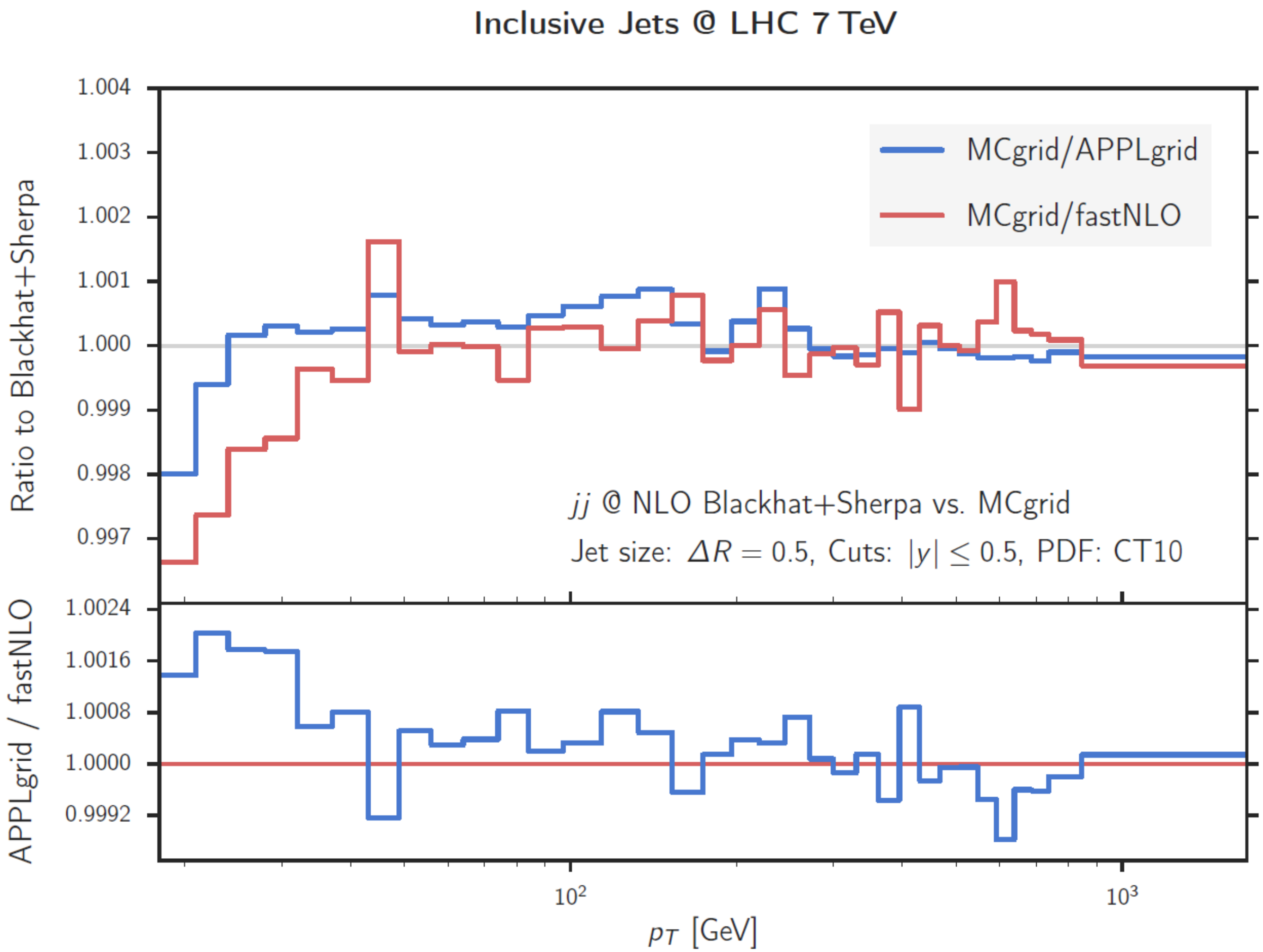
- 1M (phase space)/10M (fill) events
- Interpolation in x only
- No optimizations performed for either fastNLO or APPLgrid




Agreement between interpolations and to Sherpa at sub-permille level!

# Jets @ LHC

**Jet pT distribution:**
- **10M (phase space)/10M (fill) events**
- **Interpolation in x and scales**
- **No optimizations for either fastNLO or APPLgrid**




**Updated versions of fastNLO Toolkit and MCgrid to be released soon.**

**Agreement between interpolations and to BlackHat+Sherpa at permille level!**

# Outlook

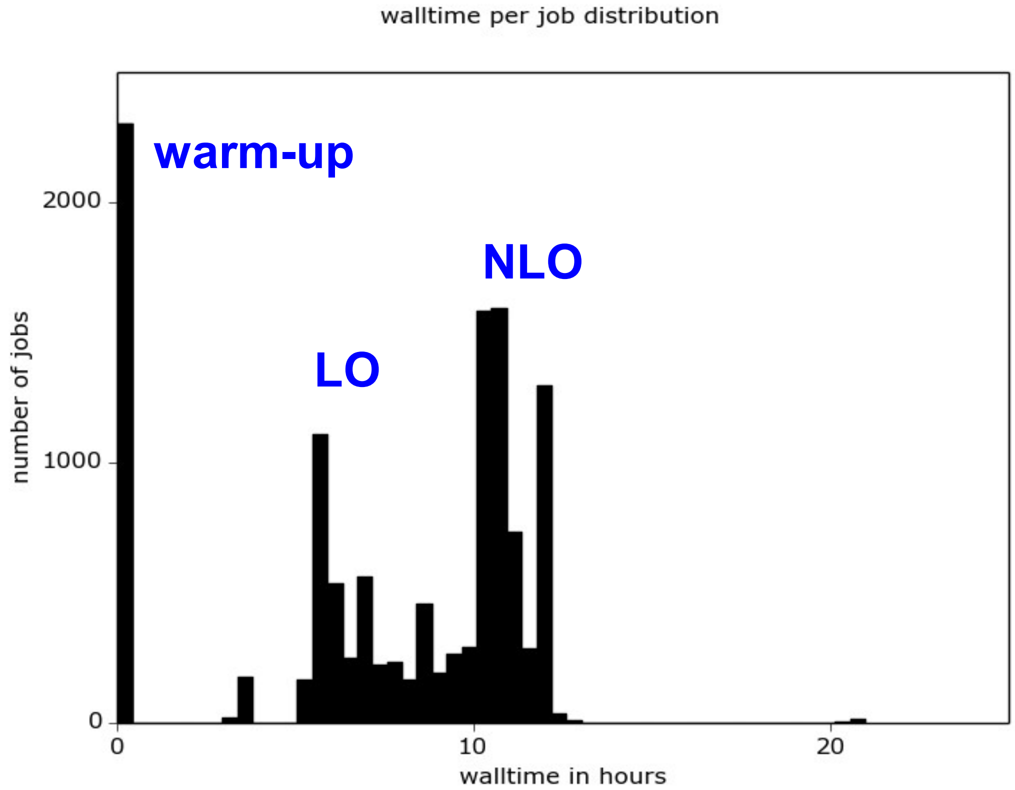
- The toolkit provides simple access to full capability of creating, filling, reading, and evaluating fast interpolation tables in the fastNLO format
- A simplified interface to NLOJet++ is publically available as well
- Uses flexible-scale table format, well-suited for NNLO
- Tested at (approx.) NNLO with DiffTop and by BlackHat
  ==> well prepared for i.a. jets at NNLO :-)
- Other theory programs can be/have been interfaced
- Demonstrated new application with MCgrid and Sherpa
- New release of the fastNLO Toolkit imminent
- Will be synchronized with new release of MCgrid
- Work in progress with Herwig++/Matchbox
- Work on inclusion of statistical uncertainty of calculation within table
- and last but not least ...

# Table production initiative

- Started large-scale table production as stress test on computing centers as private cloud providers
  - Uses Karlruher generic grid submission tool grid-control and HTCondor & OpenStack
- Test production at Xmas:
  - 800 virtualised CPUs
  - 13000 jobs
  - 95000 h of CPU time
  - $10^{12}$ events
  - 13 fastNLO tables
- Anything on your wishlist?

Many thanks to the Computing Centre of the University Freiburg for providing the bwForCluster test system!

Summary of job timings



# Backup Slides

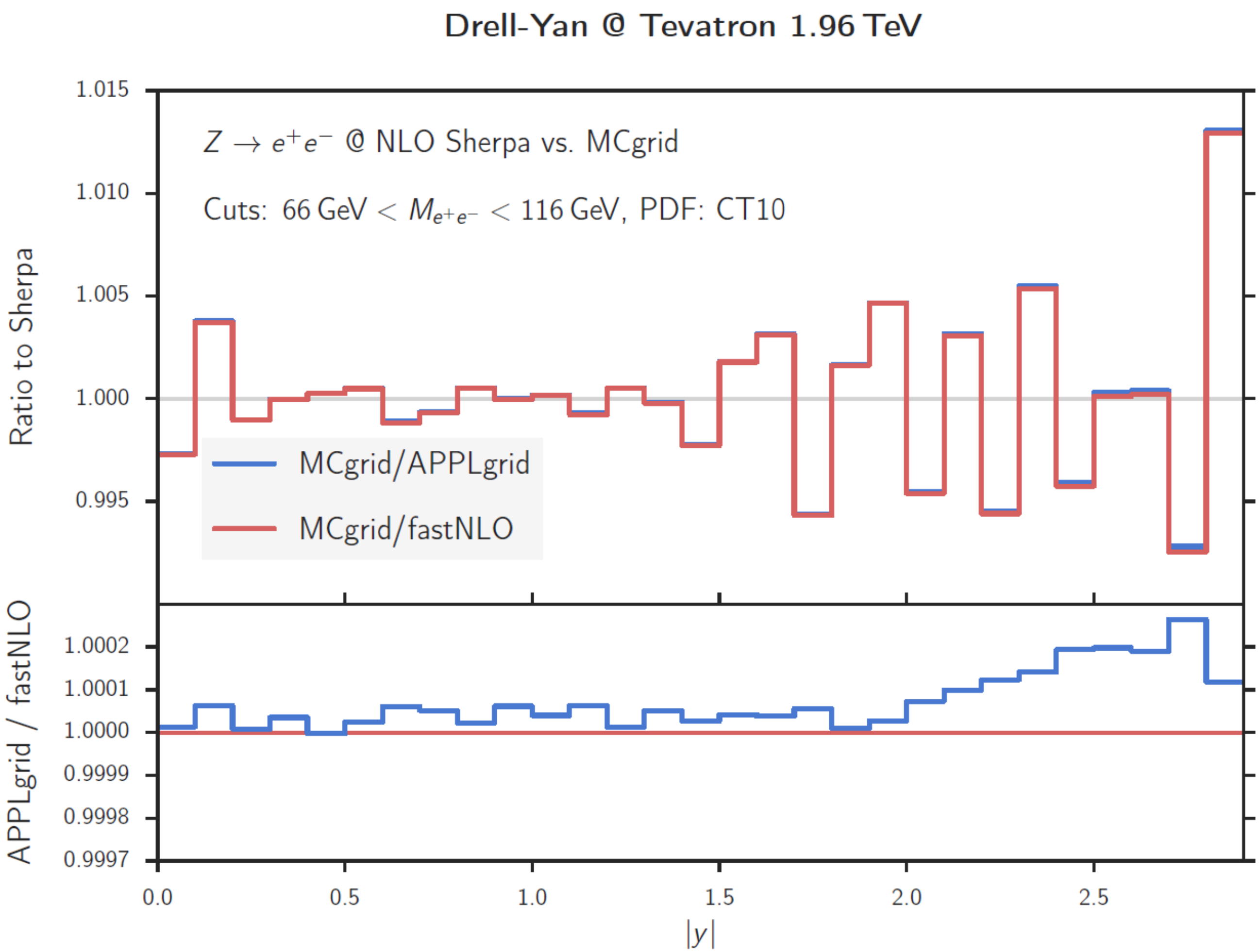
# Excerpt of steering.str

```
ScenarioName fnl2342b_I902309_v23_flex         # Name and describe scenario
ScenarioDescription {
    "d2sigma-jet_dpT_dy_[pb_GeV]"
…
JetAlgo                          2              # fastjet jet algorithm: 0,1,2=kT,CA,anti-kT
Rjet                             0.5            # Jet size parameter: Required for all jets
ptjmin                          18.             # Minimal jet pT
yjmin                            0.0            # Minimal jet rapidity
yjmax                            3.0            # Maximal jet rapidity
… extensible
LeadingOrder                       2            # Number of jets for the LO process
DifferentialDimension              2            # Dimensionality of binning
DimensionLabels {                               # Labels (symbol and unit) for dimensions
   "|y|"                                        # Defines the observables to be calculated!
   "pT_[GeV]"
}

FlexibleScaleTable               true           # Create table fully flexible in mu_f
ScaleDescriptionScale1           "pT_jet_[GeV]" # This defines the scale to be used
ScaleDescriptionScale2           "pT_max_[GeV]" # Specify 2nd scale name and unit

DoubleDifferentialBinning {{
  1stDimLo    1stDimUp    "----- Array of bin-grid for 2nd dimension -----"
    0.0         0.5          18.   21.   24.   28.   32.   37.   43.   49.   56. …
…
}}
```

**Running any other scenario can be as simple as adapting some kinematical cuts & binning, often not even a recompile necessary!**

# DY with reduced flavour-basis

- Store identical subprocesses into same “process bin” instead of full 121 flavour basis
  - Dramatic reduction in required storage space!




- ✖ But not exactly an apple-to-apple comparison anymore
  - Statistical variations visible between exclusive Sherpa events and interpolated results
- Anyway still ok at sub-percent level