Recent Developments

Daniel Britzger, Klaus Rabbertz, Georg Sieber, Fred Stober, Markus Wobisch (DESY, KIT, KIT, Uni Hamburg, Louisiana Tech University)
Use of HOPPET for $\mu_f$ variation

- fastNLO uses 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
  - Continuous fast variation at NLO
  - Same method as used in APPLgrid

Points pre-calculated (fixed-scale table @ $\mu_f = 0.5, 1.0, 2.0$)

Lines derived using HOPPET

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 $n\beta_0 \ln(c_r)$
- Flexible-scale implementation
  - Store scale-independent weights:

Factorization scale variations
- Calculate LO DGLAP splitting functions using HOPPET
- Store coefficients for desired scale factors
- Flexible-scale implementation

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
    \[
    \omega(\mu_R, \mu_F) = \omega_0 + \log(\mu_R^2)\omega_R + \log(\mu_F^2)\omega_F + \log^2(\mu_R^2)\omega_{RR} + \log^2(\mu_F^2)\omega_{FF} + \log(\mu_R^2)\log(\mu_F^2)\omega_{RF}
    \]
  - Log’s for NLO: \(\omega_0, \omega_R, \omega_F\)\r
  - Additional log’s in NNLO: \(\omega_{RR}, \omega_{FF}, \omega_{RF}\)
    - 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 \(p_{T,\text{max}}\)
    - 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

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 $p_T_Z$.

### Tables from H1 multi-jet study use $\sqrt{Q^2}$ and $p_T$

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
<th>HepData</th>
<th>RIVET Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>fnh5001_l1301218</td>
<td>H1 inclusive jet HERA-II (kt and anti-kt); LO, NLO</td>
<td>InSPiRE</td>
<td>No RIVET analysis available</td>
</tr>
<tr>
<td>fnh5002_l1301218</td>
<td>H1 dijet HERA-II (kt and anti-kt); LO, NLO</td>
<td>InSPiRE</td>
<td>No RIVET analysis available</td>
</tr>
<tr>
<td>fnh5003kt_l1301218</td>
<td>H1 trijet HERA-II (kt); LO, NLO</td>
<td>InSPiRE</td>
<td>No RIVET analysis available</td>
</tr>
<tr>
<td>fnh5003ak_l1301218</td>
<td>H1 trijet HERA-II (anti-kt); LO, NLO</td>
<td>InSPiRE</td>
<td>No RIVET analysis available</td>
</tr>
<tr>
<td>fnh4002_l1875006</td>
<td>ZEUS inclusive dijet HERA-II (kt); LO, NLO</td>
<td>InSPiRE</td>
<td>No RIVET analysis available</td>
</tr>
<tr>
<td>fnh5201_l838435</td>
<td>H1 inclusive jets at low $Q^2$ HERA-I (kt); LO, NLO</td>
<td>InSPiRE</td>
<td>No RIVET analysis available</td>
</tr>
<tr>
<td>fnh5401_l818707</td>
<td>H1 inclusive jets at high $Q^2$ HERA-I (kt); LO, NLO</td>
<td>InSPiRE</td>
<td>No RIVET analysis available</td>
</tr>
<tr>
<td>fnh5101_l753951</td>
<td>H1 inclusive jets HERA-I (kt); LO, NLO</td>
<td>InSPiRE</td>
<td>No RIVET analysis available</td>
</tr>
<tr>
<td>fnh4401_l724050</td>
<td>ZEUS inclusive jets HERA-I (kt); LO, NLO</td>
<td>InSPiRE</td>
<td>No RIVET analysis available</td>
</tr>
<tr>
<td>fnh4301_l593409</td>
<td>ZEUS inclusive jets HERA (kt); LO, NLO</td>
<td>InSPiRE</td>
<td>No RIVET analysis available</td>
</tr>
</tbody>
</table>

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

HERA: $ep \@ \sqrt{s}(s) = 319$ GeV

HERA: $ep \@ \sqrt{s}(s) = 300$ GeV

fastnlo @ HepForge
### Flexible-scale tables in DIS

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 $p_T^Z$.

#### Tables from H1 multi-jet study use $\sqrt{Q^2}$ and $p_T$

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>4L, NLO</th>
<th>InSPIRE</th>
<th>HepData</th>
<th>RIVET Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>fhh5001_i1301218</td>
<td>H1 inclusive jet HERA-II (kt and anti-kt): LO, NLO</td>
<td>no HepData</td>
<td>no RIVET analysis available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fhh5002_i1301218</td>
<td>H1 dijet HERA-II (kt and anti-kt): LO, NLO</td>
<td>no HepData</td>
<td>no RIVET analysis available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fhh5003kt_i1301218</td>
<td>H1 trijet HERA-II (kt): LO, NLO</td>
<td>no HepData</td>
<td>no RIVET analysis available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fhh5003ak_i1301218</td>
<td>H1 trijet HERA-II (anti-kt): LO, NLO</td>
<td>no HepData</td>
<td>no RIVET analysis available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fhh4002_i1875006</td>
<td>ZEUS inclusive dijet HERA; (Note: This table only works with the new fastnlo_toolkit reader, but not yet with the old fastnlo_reader.)</td>
<td>no HepData</td>
<td>no RIVET analysis available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fhh5201_i838435</td>
<td>H1 inclusive jets at low $Q^2$ HERA-I (kt): LO, NLO</td>
<td>no HepData</td>
<td>no RIVET analysis available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fhh5401_i818707</td>
<td>H1 inclusive jets at high $Q^2$ HERA-I (kt): LO, NLO</td>
<td>no HepData</td>
<td>no RIVET analysis available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fhh5101_i753951</td>
<td>H1 inclusive jets HERA-I (kt): LO, NLO</td>
<td>no HepData</td>
<td>no RIVET analysis available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fhh4401_i724050</td>
<td>ZEUS inclusive jets HERA-I (kt): LO, NLO</td>
<td>no HepData</td>
<td>no RIVET analysis available</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### HERA: $ep @ \sqrt{s} = 319$ GeV

- fhh4301_i593409 | ZEUS inclusive jets HERA (kt): LO, NLO | no HepData | no RIVET analysis available |

---

Additional information:

- Note: All HERA tables are flexible-scale tables $\implies$ The C++ reader versions must be used.
- 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 $p_T^Z$.
Following a discussion I had with Frank Krauss and a follow-up at the Benasque PDF Workshop, tables can now be stored with data.
Use with BlackHat N-Tuples

Speed vs Generality

- BlackHat
- N-Tuple files
- FastNLO

Slide from Daniel Maitre

Loops and Legs 2014, Weimar, 1th May
Differential $t\bar{t}$ in approx. NNLO: $d\sigma/dp_T$, $d\sigma/dy$

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

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

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

1. Prepared Toolkit library for creating & evaluating fastNLO interpolation tables
   - Independent of any generator
2. Facilitated use with extensible steering files
3. Being asked at DIS we put together an example of Fortran-based access to the C++ library
4. Interface even more theory programs …
   - NLO for higher multiplicities
   - NNLO e.g. Z+jet, jets
   - …
**Simple example for use of Toolkit**

- **Initialize fastNLO class(es)**
  ```cpp
define_steering_str = steering.str
fnlo.Create()
fnlo.SetOrderOfCalculation(int order);
```

- **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 Sherpa2 & MCgrid2

In collaboration with Enrico Bothmann & Steffen Schumann

✓ fastNLO Toolkit access implemented:

- Events generated with Sherpa 2.1.1/2.2.0
- 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 ...

GoSam, G. Cullen et al., EPJC72, 2012.
NJET, S. Badger et al., CPC184, 2013.
...
Snippets of Rivet+MCgrid analysis

```cpp
#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 {
        ...

        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

Book & config grid and histos

Fill events in event loop.

Final check out, normalize, write table.
Test with inclusive Jets

Previously:
Drell-Yan Z rapidity @ Tevatron
- 1M (phase space) / 10M (fill) events
- Constant scale → interpolation in x only
- Agreement at sub-permille level

NEW HERE:
Inclusive Jets @ LHC
- 100M (NLOJet++ ph. sp.) / 4M (fill) events
- Dynamic scale → interpolation in x & Q
- Problem in interface MCgrid-fastNLO fixed
- Agreement at sub-permille level
Comparison to NLOJet++

Sherpa/BlackHat inclusive Jets:
- 100M (NLOJet++ ph. sp.) / 4M fill events
- ~ only some hours on 4 cores of my Laptop!

NLOJet++ inclusive Jets:
- 100M (phase space) / 4G fill events
- ~ 3000h on cluster using 400 nodes
- Large stat. fluctuations, but no apparent trend!

Note: Both calculations use an event-wise dynamical scale, pTmax, jet-wise scales not possible currently with Sherpa-MCgrid.


Klaus Rabbertz  London, UK, 02.09.2015  QCD@LHC 2015
Comparison to Data

Agreement with NLOJet++ and Data except at low $p_T$ since NP corrections not included here (Stat. fluctuations still there, of course.)
Use with Rivet 2 & YODA Format

Started as a CERN Summer student (S. Tyros) project last year with Peter Skands.

Can be used to provide NLO histograms with uncertainty to MCPLOTS web site.

NLO with scale uncertainty / Data

NLO with PDF uncertainty / Data

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

- The toolkit provides simple access to full capability of fastNLO
- Creating, filling, reading, and evaluating fast interpolation tables in the fastNLO format
- A simplified interface to NLOJet++ is publically available
- Flexible-scale table format ideally suited for NNLO
- Tested at (approx.) NNLO with DiffTop and by BlackHat
  => first applications @ NNLO
- Other theory programs can be/have been interfaced
- Demonstrated new application with MCgrid and Sherpa
- Will be synchronized with new release of MCgrid
- Progress with further theory interfaces …!
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/...]

QCDNUM, M. Botje, CPC182, 2011.
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 {
  "|y|"
  "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!
Demo plot using Python extension

✔ Python extension available

… [--enable-pyext]

✔ Easy example plotting 2D scale dependence:

```python
#! /usr/bin/env python2
from fastnlo import fastNL0LHAPDF
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import axes3d
import numpy as np

fnlo = fastNL0LHAPDF('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]

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
```

Setup Python with fastNLO

Select table, PDF & mem.

Define $\mu_r$, $\mu_f$ ranges

Loop over $\mu_r$, $\mu_f$

Plot
Extra slide: CMS dijet mass

Central scale: $\mu = <p_T_{1,2}>$

Outer $|y_{max}|$ bin!

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

Derived from one fastNLO flexible-scale table
**Extra slide: ATLAS dijet mass**

Central scale: $\mu = p_T^{\text{max}}$

![Graph 1](image1)

Positive deviation: +80k

Negative deviation: -80k

Outer $y^*$ bin!

Central scale: $\mu = p_T^{\text{max}} \cdot \exp(0.3 \cdot y^*)$

![Graph 2](image2)

Positive deviation: +80k

Positive deviation: +55k

Derived from one fastNLO flexible-scale table