fastNLO Toolkit

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

Karlsruhe Institute of Technology

The fastNLO concept

Use interpolation kernel

- Introduce set of n discrete x-nodes, x_i's being equidistant in a function f(x)
- Take set of Eigenfunctions $E_i(x)$ around nodes x_i

→ Interpolation kernels

- Actually a rather old idea, see e.g.

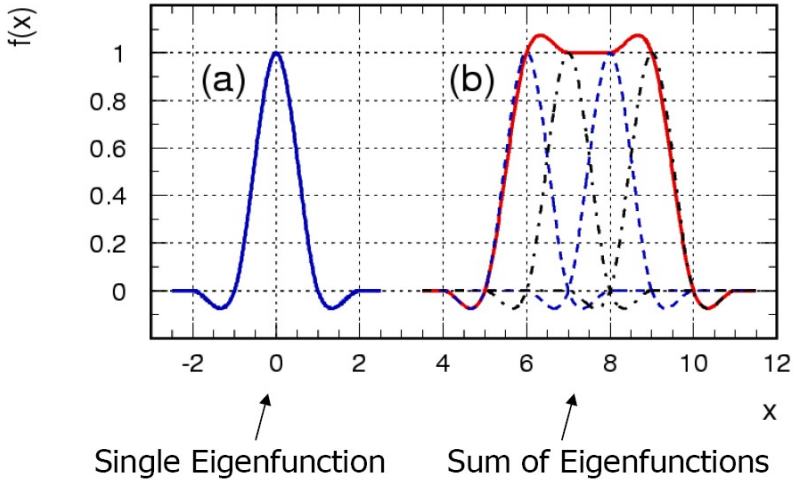
C. Pascaud, F. Zomer (Orsay, LAL), LAL-94-42

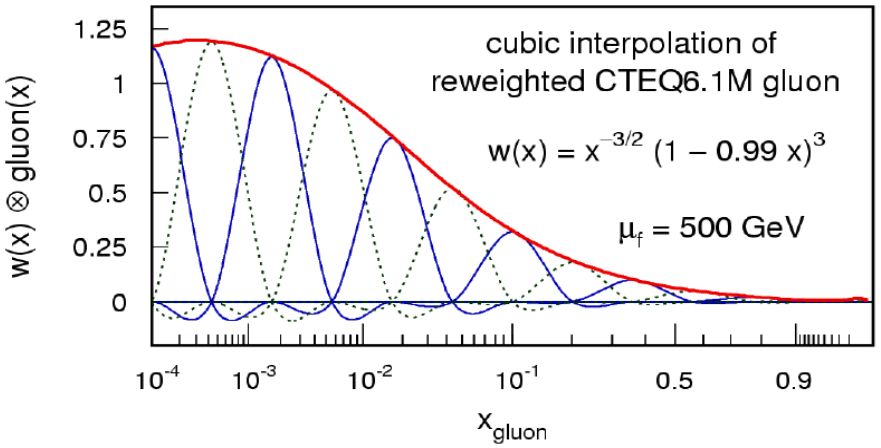
→ Single PDF is replaced by a linear combination of interpolation kernels

$$f_a(x) \cong \sum_i f_a(x_i) \cdot E^{(i)}(x)$$

→ Then the integrals are done only once

→ Afterwards only summation required to change PDF





Store a table with the convolution of the pert. coefficients with the interpolation kernel

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 (EPJ C (2010) 66: 503)]*
- Flexible-scale implementation
  - Store scale-independent weights: $\omega(\mu_R,\mu_F) = \omega_0 + \log(\mu_R)\omega_R + \log(\mu_F)\omega_F$ *[fastNLO]*

Factorization scale variations

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

Georg:
[new: also fastNLO]

Scale variations for NNLO calculations

- a-posteriori renormalization scale variations become more complicated
- NLO splitting functions are needed for factorization scale variations
  - Calculations become slow again => Not desired for fast repeated calculations

Flexible-scale implementation in NNLO

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 α_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

How to use elsewhere ?

New tool: fastNLO toolkit

What about application of fastNLO to other processes/programs ?

Hardly any theoretical limitation of fastNLO concept to pQCD or EW calculations

Why not used more frequently?

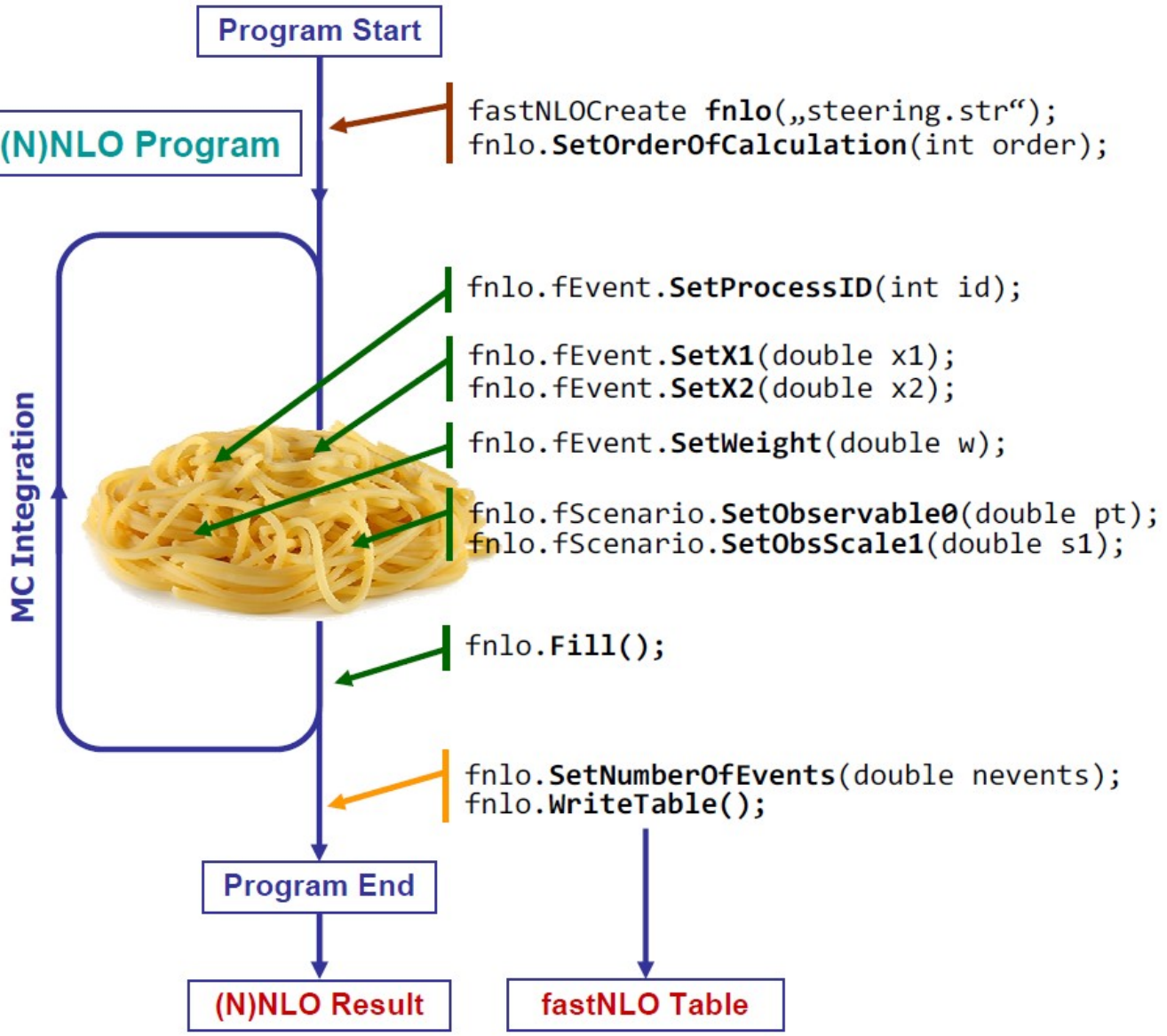
Interface of fastNLO to theory programs often very complicated…

- Theory codes are not optimized (at all) for fastNLO
- Technical difficulties are mostly limiting factor in usage

Goal:
Provide simple and flexible code to interface fastNLO to any kind of (N)NLO program

Newly developed tool: *fast*NLO Toolkit

Toolkit working concept



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

Plans from last October

- 1. Cross check old v1.4 versus new v2.1 tables … Done!
- 2. Cross check new reader code in C++ vs. Fortran … Done!
- 3. Public release of reader code as autotools tarball … Done!
- 4. Transform C++ reader code into linkable library … Done!
- 5. Transform table creation code into linkable library as independent as possible from NLOJet++!
- In progress, first test version exists.
- Make interface available for other N?LO codes.

All done by now!

Public prerelease of fastNLO Toolkit

fastNLO

fast pQCD calculations for hadron-induced processes

Home | Documentation | Scenarios | Code | Interactive (maintenance) | Links

General concept

The fastNLO project provides computer code to create and evaluate fast interpolation tables of pre-computed coefficients in perturbation theory for observables in hadron-induced processes.

This allows fast theory predictions of these observables for arbitrary parton distribution functions (of regular shape), renormalization or factorization scale choices, and/or values of alpha_s(Mz) as e.g. needed in PDF fits or in

July 17, 2014

Public prerelease of new fastNLO Toolkit

The new fastNLO Toolkit provides a library with all functionality to create, fill, read, and evaluate interpolation tables in the fastNLO format. It comes with a much improved structure that allows other programs to be interfaced to fastNLO. In addition, an example interface and code how to use NLOjet++ with this toolkit is available. Both packages can be downloaded from our code web pages.

July 16, 2014

Finally: New calculations for H1 multijets available

New fastNLO tables for the recent measurement by H1 on multijet production [arXiv:1406.4709] are available as flexible-scale tables. They are complemented by further jet production tables in DIS that were previously distributed within the HERAFitter framework. They can be downloaded here.

fastNLO code page

General concept

fastNLO provides the code to create, fill, read, and evaluate fast interpolation tables of pre-computed coefficients in perturbation theory for observables in hadron-induced processes. This allows fast theory predictions of these observables for arbitrary parton distribution functions (of regular shape), renormalization or factorization scale choices, and/or values of alpha_s(Mz) as e.g. needed in PDF fits or in systematic studies. Very time consuming complete recalculations are thus avoided.

Code

New code named the **fastNLO Toolkit (v2.3)** has been made available in the form of a prerelease (beta release). The toolkit provides a library with all functionality to create, fill, read, and evaluate interpolation tables in the fastNLO format. It comes with a much improved structure that allows other programs to be interfaced to fastNLO. The only dependency is to the **LHAPDF** package for the evaluation part. Essentially, it will replace the former fastNLO Reader (v2.1) package.

The **interface from NLOJet++ to fastNLO** is also available as a prerelease and requires a patched version of NLOJet++ and the fastNLO Toolkit. It is recommended to use it in connection with **FastJet** for the various jet algorithms. Interfaces for **DiffTop** and the **Herwig++ MatchBox** are in preparation.

fastNLO code v23 pages

Installation

Installation of distribution package:

Via GNU autotools setup (NOT required for installation), in unpacking directory of the *.tar.gz file do:
./configure --prefix=your_local_directory
(should contain other required packages, otherwise specify separate path via
--with-package=path_to_package; see also ./configure --help)
make -j number_of_cores_to_use
make install
make check (not yet implemented)

For more information see the README files of the fastNLO Toolkit and the fastNLO Interface to NLOjet++.

fastnlo_toolkit_2.3.1 (pre)releases			
pre-1854	ReleaseNotes	ChangeLog	First public release, presented at DIS Warschau and CMS PDF Forum

fastnlo_interface_nlojet-2.3.1 (pre)releases			
pre-1855	ReleaseNotes	ChangeLog	First public release, presented at DIS Warschau and CMS PDF Forum

External packages		
LHAPDF-5.9.1	Les Houches Accord Parton Distribution Functions	Necessary for table evaluation with PDF sets
NLOJet-4.1.3-patched	NLOJet++ program with patches to work with fastNLO	Necessary for table filling with 2- and 3-jet NLO jet production in DIS and pp(bar)
FastJet-3.0.6	FastJet library of jet algorithms	Recommended for running jet algorithms

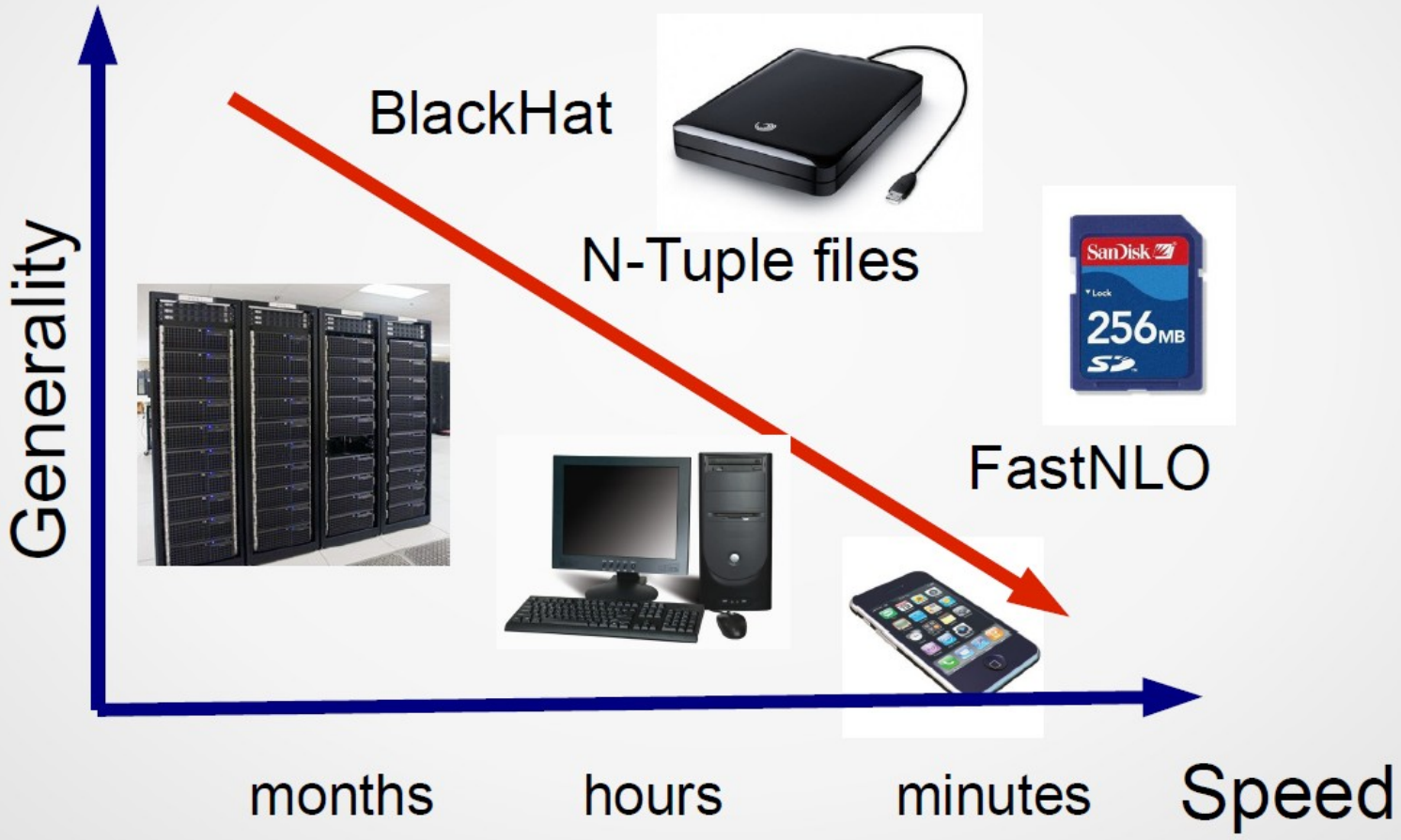
Toolkit in use

New tables from Daniel for recent H1 multi-jet study!

Note: All HERA tables are flexible-scale tables ==> The C++ reader versions must be used.			
HERA: ep @ sqrt(s) = 319 GeV			
fnh5001_l1301218	H1 inclusive jet HERA-II (kt and anti-kt); LO, NLO		
	inSPIRE	no HepData yet	no RIVET analysis available
fnh5002_l1301218	H1 dijet HERA-II (kt and anti-kt); LO, NLO		
	inSPIRE	no HepData yet	no RIVET analysis available
fnh5003kt_l1301218	H1 dijet HERA-II (kt); LO, NLO		
	inSPIRE	no HepData yet	no RIVET analysis available
fnh5003ak_l1301218	H1 dijet HERA-II (anti-kt); LO, NLO		
	inSPIRE	no HepData yet	no RIVET analysis available
fnh4002_l875006	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_l838435	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_l818707	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_l753951	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_l724050	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_l593409	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.)		

Toolkit in use

Speed vs Generality



Slide from Daniel Maitre

Loops and Legs 2014, Weimar, 1th May

Fill BlackHat parton events (N-tuple files) into fastNLO table

Toolkit in use

Toolkit interfaces:

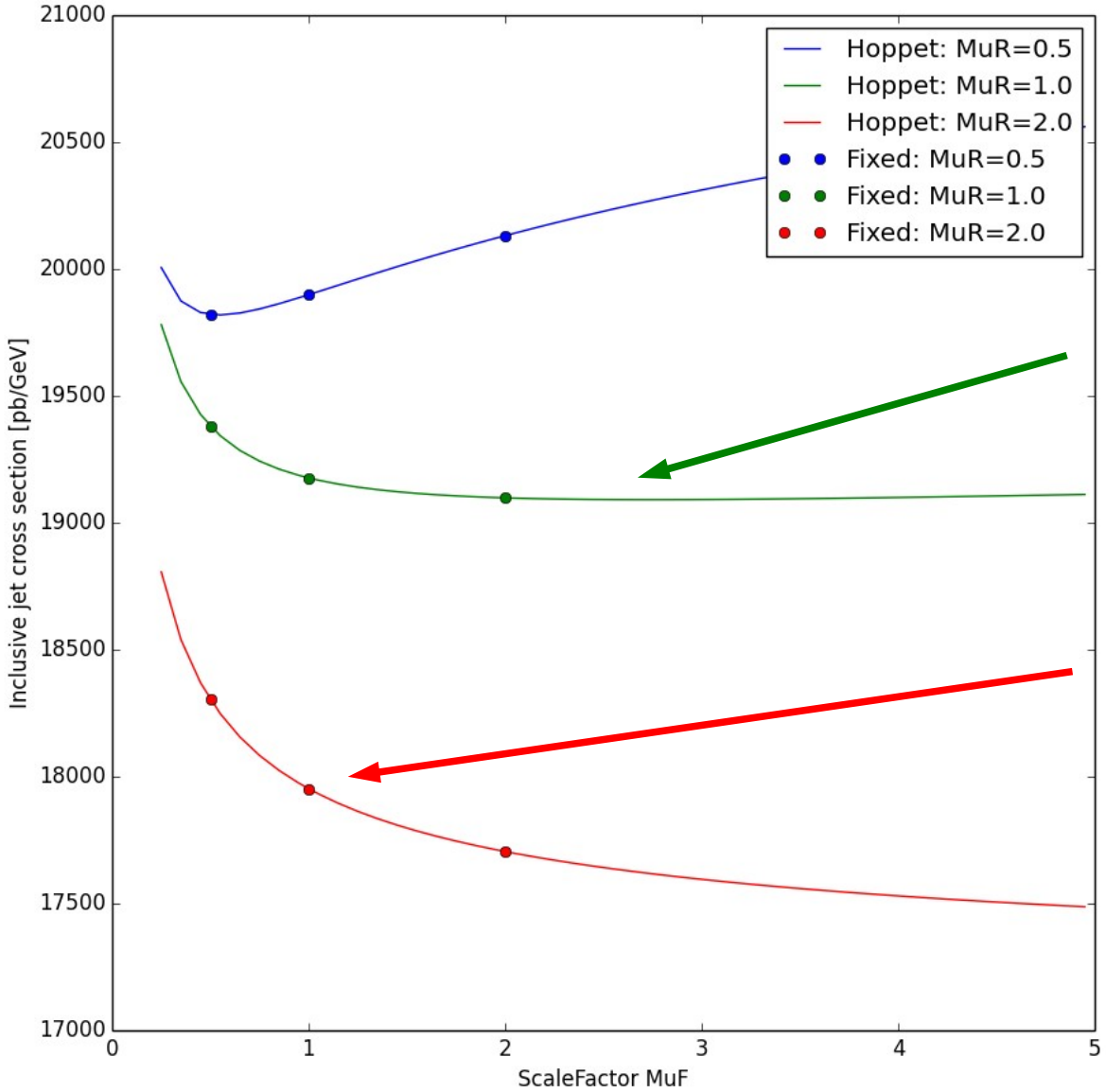
- 1. Prerelease for NLOJet++ public since TODAY!
- 2. Interface to DiffTop exists, see next talk by Marco Guzzi
- 3. Interface to the Herwig++ MatchBox with many additional processes, e.g. via MadGraph, Njet, ... is in progress
- 4. Would be interesting to test usage in aMCfast, which is possible according to Juan

Other work in progress

Use of HOPPET for mu_f variation

Not used in fastNLO up to now, but in APPLGRID.

Needs a bit less storage space but requires additional CPU time for each Table evaluation. More difficult at NNLO.



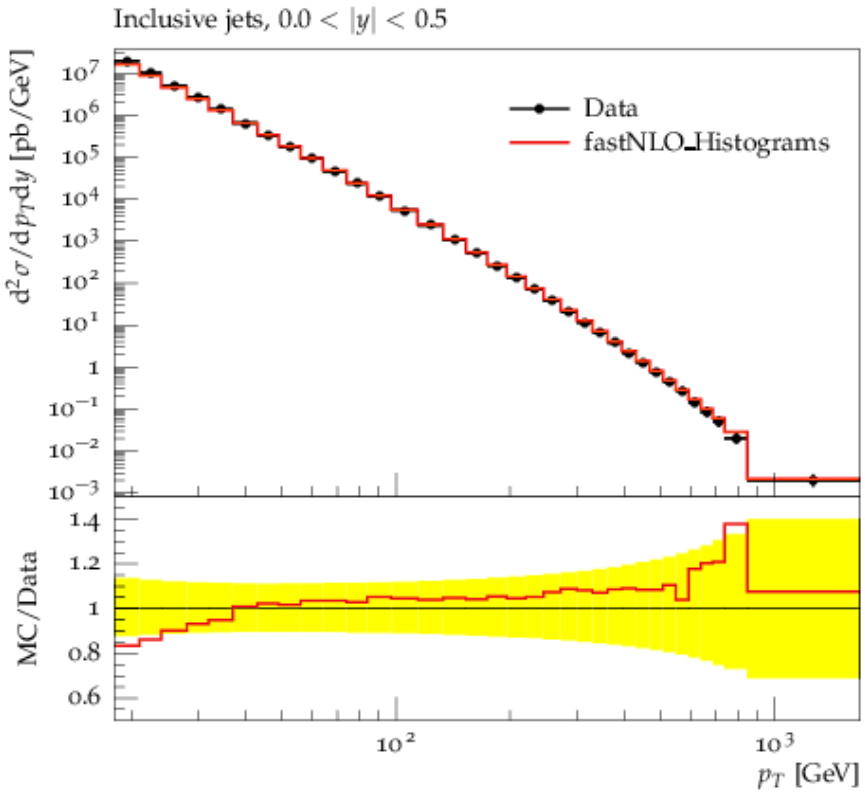
Lines derived with HOPPET

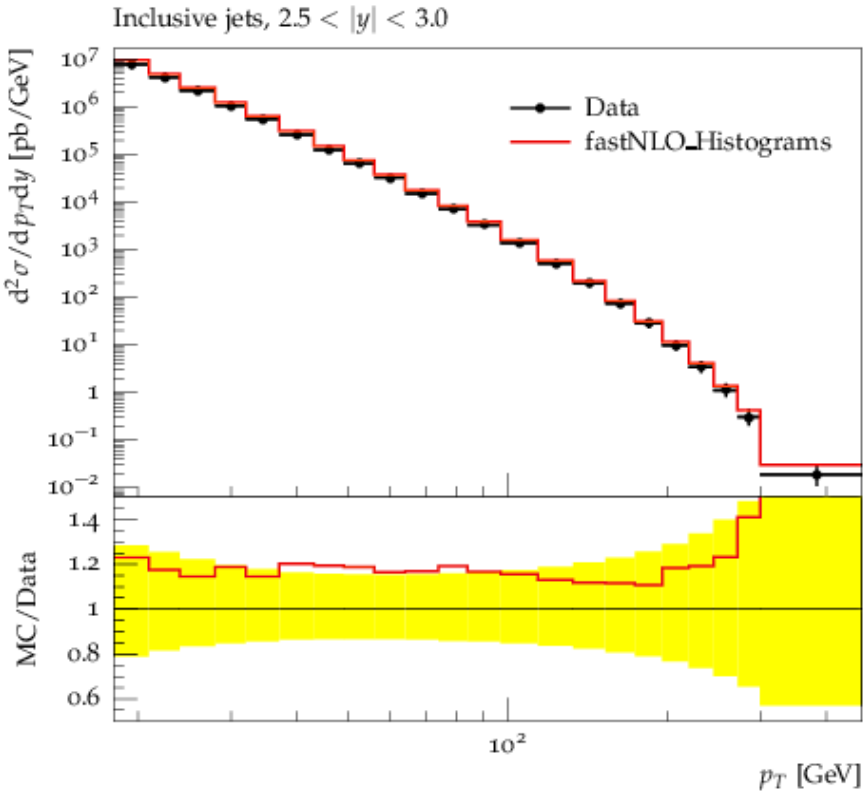
Points from fastNLO fixed-scale tables

Other work in progress

Summer student project (Stefanos Tyros) with Peter Skands:

- 1. Provide YODA formatted output for fastNLO tables Works!
- 2. Compare histograms with data using Rivet analyses … OK!
- 3. Implement fastNLO tab into development version of MCPlots page … in progress





Outlook

- A prerelease of the fastNLO Toolkit is publicly available since TODAY!
- The toolkit provides full capability to create, fill, read, and evaluate fast interpolation tables in the fastNLO format
- Other theory programs can be interfaced
- Works at NNLO ==> well prepared for jets at NNLO
- An interface to NLOJet++ is publically available as well
  ==> Do your own NLO tables!
- In case of question, problems, or requests don't hesitate to contact us!
- Much more additional integration work and ideas are in progress ...

Your feedback is welcome