the APPLgrid project: Progress and plans

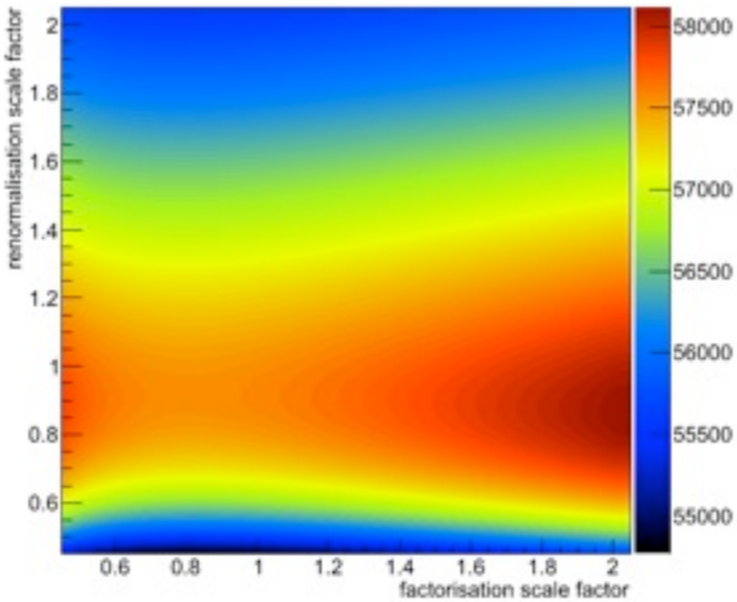
Mark Sutton
University of Sussex

On behalf of the APPLgrid developers,
Tancredi Carli, CERN,
Pavel Starovoitov, DESY, MS & many others

19th February

What is APPLgrid?

- APPLgrid is a fully open source package to build a library of C++ utility classes for performing fast (N)NLO convolutions with PDFs (with an optional FORTRAN interface)
- It uses a customised grid storage for reduced memory usage
- Can be used for fast cross section production with pre-existing grids
  - Arbitrary renormalisation and factorisation scale variation
  - Arbitrary beam-energy rescaling
  - Different PDF sets
  - Any number of multiplicative corrections can be stored and applie in the the grid
  - Allows different input PDFs for each incoming hadron
- Can be used by the user to generate custom grids for different cross sections and processes



- Currently available interfaces for ...
  - NLOjet++ for jet production
  - All processes in MCFM: Electroweak boson production, heavy flavour production, boson + jet production (plus generic interface for all remaining processes)
  - All fixed order NLO processes in aMC@NLO, using the aMCfast interface, arXiv:1406.7693
  - Sherpa for fixed NLO processes using MCgrid, arxiv:1312.4460 - in addition an independent native APPLgrid - Sherpa interface is available
- Currently working on an interface to DYNNLO for EW boson production at NNLO
- In the past we have also toyed with interfaces to JETRAD (NLO) and Vrap (NNLO)

applgrid.hepforge.org

APPLgrid – Hepforge

https://applgrid.hepforge.org

APPLgrid is hosted by Hepforge, IPPP Durham

the APPLgrid project

Home | Downloads | Documentation | Subversion | News | Links

Convolution Code Download
Current version applgrid-1.4.70
Basic example code here
The hoppet code version 1.1.5 for QCD evolution from Gavin Salam and Juan Rojo.

Calculation Code
MCFM: (use standard mcfm code)
mcfm-patch (mcfm applgrid patch 0.0.8)
mcfm-bridge (version 0.0.35 - for mcfm-6.8)
NLOjet++:
nlojet++ 4.0.1 (applgrid version 0.0.2)
nlojet++ lhapdf wrapper 1.0.0 (applgrid version 0.0.2)
nlojet++ user module (applgrid version 0.0.2)

Grid Download
Full details on the Downloads page:

Grids for sqrt(s)=2.76 TeV:
ATLAS inclusive jets (2011 - 0.2pb^{-1}) grids

Grids for sqrt(s)=7 TeV:
ATLAS inclusive jets (2010 - 17nb^{-1}) grids
ATLAS inclusive jets (2010 - 37pb^{-1}) grids
ATLAS inclusive dijets (2010 - 37pb^{-1}) grids
ATLAS inclusive dijets (2011 - 4.5fb^{-1}) grids
ATLAS W+, W- data (2010 - 35pb^{-1}) grids
ATLAS Z0 data (2010 - 35pb^{-1}) grids

Quick Start Guide
how to run the APPLgrid code

Citation
Please cite the APPLgrid as
Eur Phys J C 66 (2010) 503

Welcome

News! New versions of applgrid and the mcfm interface released. See the News page for more details.
The APPLgrid project provides a fast and flexible way to reproduce the results of full NLO calculations with any input parton distribution set in only a few milliseconds rather than the weeks normally required to gain adequate statistics.

Written in C++ (although a fortran interface is included) it can be used for the calculation of any process where the hard subprocess weights from the convolution with the PDF are available from the calculation.

The user can use existing grids simply to obtain the fast cross sections, as with fastNLO, but the complete project is pubically available should the user wish to generate the grids themselves for new cross sections. In this case, the user interfaces the grid code with NLO calculation, and after running the NLO calculation to achieve the required statistical precision once, the results of the calculation with any different parton distribution set can be calculated, typically in around 1 to 100ms, depending on the size of the grid.

At present, examples exist for MCFM and nlojet++. Since the user code that needs to run to extract the weights and create the grid may require changes to the NLO calculation code or may be dependent on specific versions of the code, the specific versions of both MCFM and nlojet++ are included here

An interface to fastNLO (version 1) grids is included.

The code is under continuous development so please come back soon for more information.

Mark Sutton, Pavel Starovoitov, Tancredi Carli and Gavin Salam - send mail to the authors: applgrid @ projects.hepforge.org: Last updated Tue 17 Feb 2015 15:04:19 CET

downloads.htm

Grid Downloads

Here you can download grids for APPLgrid version 1.4.70 for fast cross section evaluation. These grids are fully differential and should require no additional scaling. Each should include the non-perturbative (and any additional) bin-by-bin corrections or K-factors, the application of which is discussed in the relevant papers - see the Documentation page for more information.

Please note that these grids should be based on the grids that were used for the relevant papers, and were not necessarily created by the the applgrid authors.

LHC: pp @ sqrt(s) = 2.76TeV

ATLAS inclusive jets (2011 data 0.2pb^{-1})

arXiv:1304.4739v1 Inclusive jets anti-kT: Tables 4-10 (R=0.4) and 11-17 (R=0.6)

hepdata

grid tarball
contains

```
atlas-incldijets-arxiv-1304.4739/r04/atlas-incljets-eta[1-7].root (R=0.4)
atlas-incldijets-arxiv-1304.4739/r06/atlas-incljets-eta[1-7].root (R=0.6)
```

LHC: pp @ sqrt(s) = 7TeV

ATLAS inclusive jets (2010 data 17nb^{-1})

arXiv:1009.5908v2 Inclusive jets anti-kT: Tables 1-3 (R=0.4) and 4-6 (R=0.6)

hepdata

grid tarball
contains

```
atlas-incljets-arxiv-1009.5908v2/r04/atlas-incljets-eta[1-5].root (R=0.4)
atlas-incljets-arxiv-1009.5908v2/r06/atlas-incljets-eta[1-5].root (R=0.6)
```

ATLAS inclusive jets (2010 data 37pb^{-1})

arXiv:1112.6297 Inclusive jets anti-kT: Tables 5-11 (R=0.4) and 12-18 (R=0.6)

hepdata

grid tarball
contains

```
atlas-incljets-arxiv-1112.6297/r04/atlas-incljets-eta[1-7].root (R=0.4)
atlas-incljets-arxiv-1112.6297/r06/atlas-incljets-eta[1-7].root (R=0.6)
```

ATLAS inclusive dijets (2010 data 37pb^{-1})

arXiv:1112.6297 Inclusive dijets anti-kT: Tables 19-27 (R=0.4) and 28-36 (R=0.6)

hepdata

grid tarball
contains

```
atlas-incldijets-arxiv-1112.6297/r04/atlas-incldijets-eta[1-9].root (R=0.4)
atlas-incldijets-arxiv-1112.6297/r06/atlas-incldijets-eta[1-9].root (R=0.6)
```

Code Download

Various code for download is available:

- You can download the latest version (1.4.70) of the standard APPLgrid convolution code here .
- If you wish to use arbitrary factorisation scale variation, you will need to have hoppet installed first, which you can download from here .
- There are some simple examples which you can download from here . These examples requires that you have LHAPDF installed which you can find here .
- In addition, APPLgrid uses root files for storage, although not internally, so you should also install root which you can find here .

Calculation Code

If you wish to generate your own grids, we currently have NLOjet++ available for jet production at NLO and MCFM for most other processes.

- For **NLOjet++** we have our own custom version 4.0.1 which can be downloaded from here .
  This requires the nlojet pdf module that you can download from here .
  The user module can be downloaded from here
- For **MCFM**, you should download the standard version of MCFM (6.7 or later). To link with applgrid, before installing MCFM, you should download and install the mcfm-bridge code, which you install with the usual ./configure ; make ; make install etc.

  After running this you will need to patch the standard installation using the patch file which you should untar in the MCFM base directory and then remove the file

  ```
  src/User/gridwrap.f
  ```

  then just build using make. To link with applgrid, before running make, you should set the LDFLAGS enviroment variable to the output from

  ```
  mcfm-bridge --ldflags
  ```

  NB: you should **not** use the standard install script that comes with MCFM.

Other interfaces are being developed and will be released when available.

Downloads Archive

The old downloads directory can be found here (Warning, not for the faint of heart)

Open Source

the APPLgrid project

powered by trac

Search

Login | Help/Guide | About Trac | Preferences

Wiki | Timeline | Roadmap | Browse Source | View Tickets | Search

Last Change | Revision Log

source:

View revision: View diff against:

Name	Size	Rev	Age	Author	Last Change
amcatnlo-2.0.0		443	20 months	rojo	trivial edit
amcatnlo-bridge		1234	7 months	sutt	fix autotools
amcfast		1252	7 months	bertone	amcfast-shower executable added
applgrid		1294	9 days	tcarli	corrected spelling mistake in comment
dynnlo		387	22 months	sutt	create directory structure
example		1282	5 months	sutt	fixes for pdfschool
jetmod		591	19 months	sutt	rename
lhpdf-1.0.0		593	19 months	sutt	rename
mcfm-6.0		902	18 months	sutt	fix cmd line arguments
mcfm-6.6		1079	15 months	sutt	fix broken frixione cone epsilon setting
mcfm-bridge		1296	12 hours	sutt	fix runmcfm
mcfm-patch		1298	12 hours	sutt	update makefile
nlojet-4.0.1		589	19 months	sutt	rename
obsolete		367	23 months	sutt	move tags to obsolete directory
sherpa		1268	5 months	tcarli	added steering for ATLAS ttbar+jets
utils		1219	8 months	sutt	improve tagging and preparation
web		1182	9 months	sutt	modify news bg colours

View changes...

Note: See TracBrowser for help on using the repository browser.

trac | Powered by Trac 1.0.1 By Edgewall Software.

Visit the Trac open source project at http://trac.edgewall.org/

MCFM

- Current version 6.8 tested with W, Z production, QQ production (bbar, ttbar) W, Z + jets, W+c, prompt photon production
- Since W and Z production in APPLgrid has been around for a good while and is well known - won't discuss here

- Due largely to the efforts of Pavel Starovoitov and Alberto Guffanti since version MCFM 6.7, modifications required to interface with APPLgrid have been included in the standard MCFM - no longer any need for a custom version

- This is partly possible because of a new paradigm for the interface with MCFM
    - Non obtrusive hook functions based on null function pointers that can be left compiled into the standard MCFM code
    - New bridge package that includes a setup function which initialises the hook functions to external filling routines ...
        - APPLgrid interface included, or excluded simply by linking against the APPLgrid libraries, without requiring any recompilation ...
        - If you don't link against the APPLgrid libraries you get standard MCFM, no MCFM code needs to be compiled

- You do however, need to patch your initial MCFM installation with files from a small patch archive before compilation
- This additionally improves on the Makefile from the standard MCFM installation
- In addition, includes a patch from John Campbell to treat dynamic scale for top production

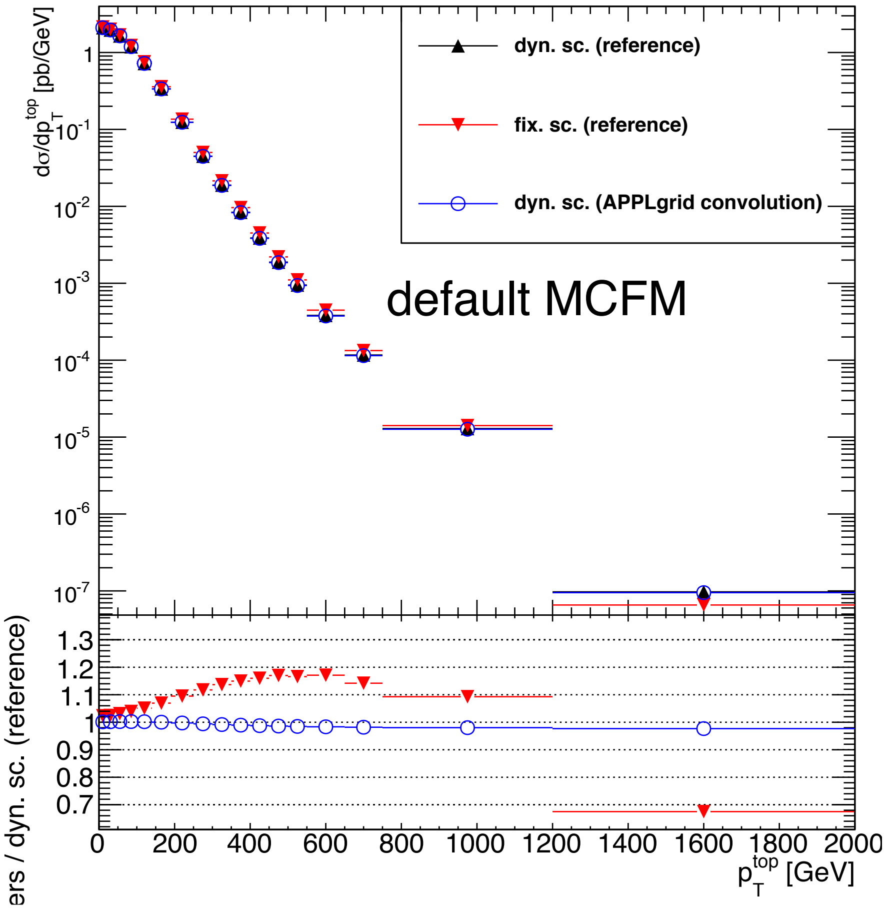
- (Incidentally, this paradigm is also used to great success in the aMCfast interface for aMC@NLO)

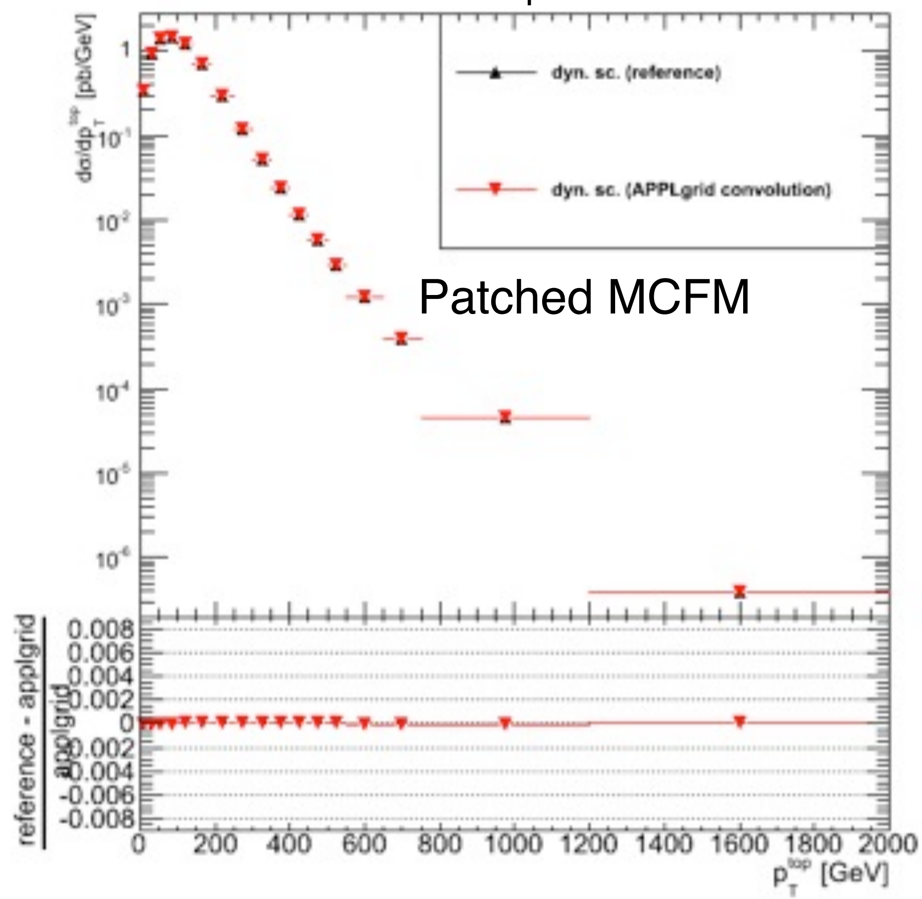
Current MCFM status ...

- All MCFM processes, in principle now included
- Those that we have (personally) verified ...
  - EW Boson production ...
    - Inclusive $W^{\pm}$ production (processes 1, 6)
    - $W^{\pm}$ + jet production (processes 11, 15)
    - Inclusive Z production (process 31)
    - Z + jet production (processes 41-43)
    - Prompt photon production (processes 280 - 286) - including fragmentation
  - Heavy flavour ...
    - ttbar production (semi-leptonic decay) with/without spin correlations (proc. 141, 142, 144, 145)
    - ttbar production (hadronic decay), with radiative corrections in t/W decays etc (proc. 146 - 151)
    - Inclusive ttbar production (157)
    - Inclusive bbar production (158)
    - Inclusive ccbar production (159)
  - Combined EW+heavy flavour …
    - $W^{\pm}$ + c production (processes 13-16)

MCFM top - dynamic scale patch

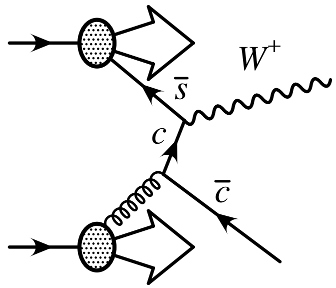
plots from Peter Berta



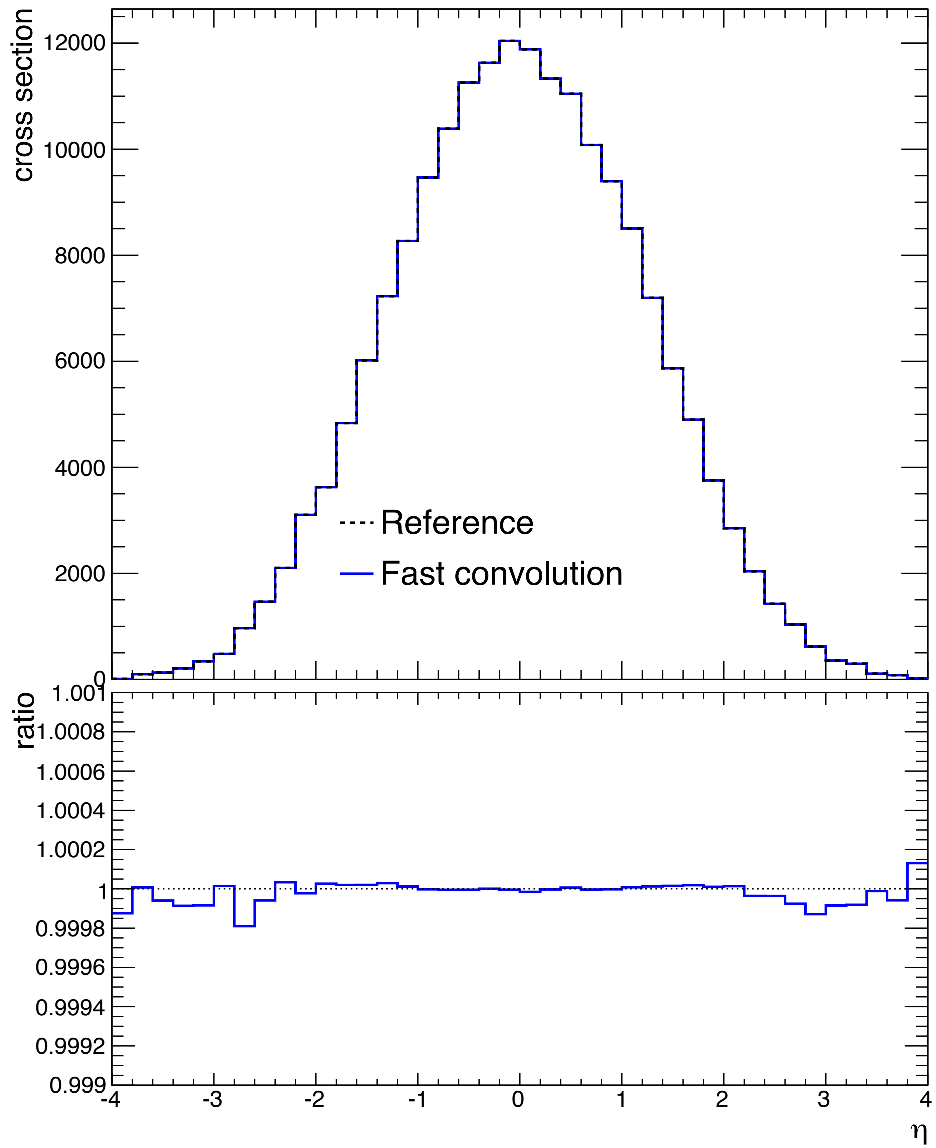


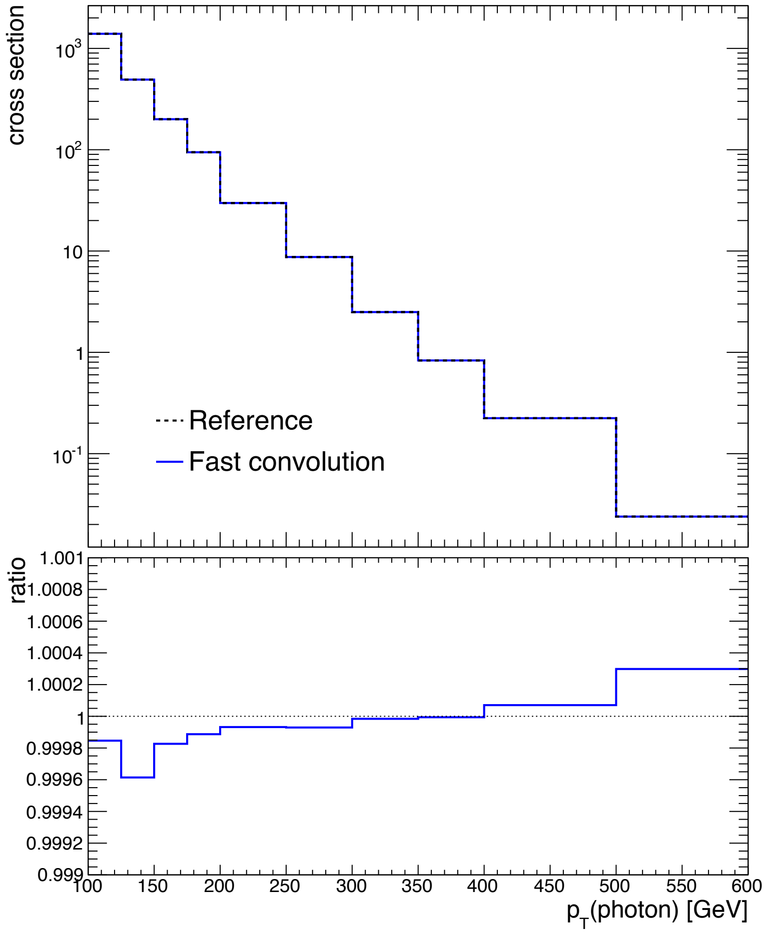
- Without the patch, small ~ 2% non-closure of the top cross section at high pT
- With the patch closes to better than 0.1%

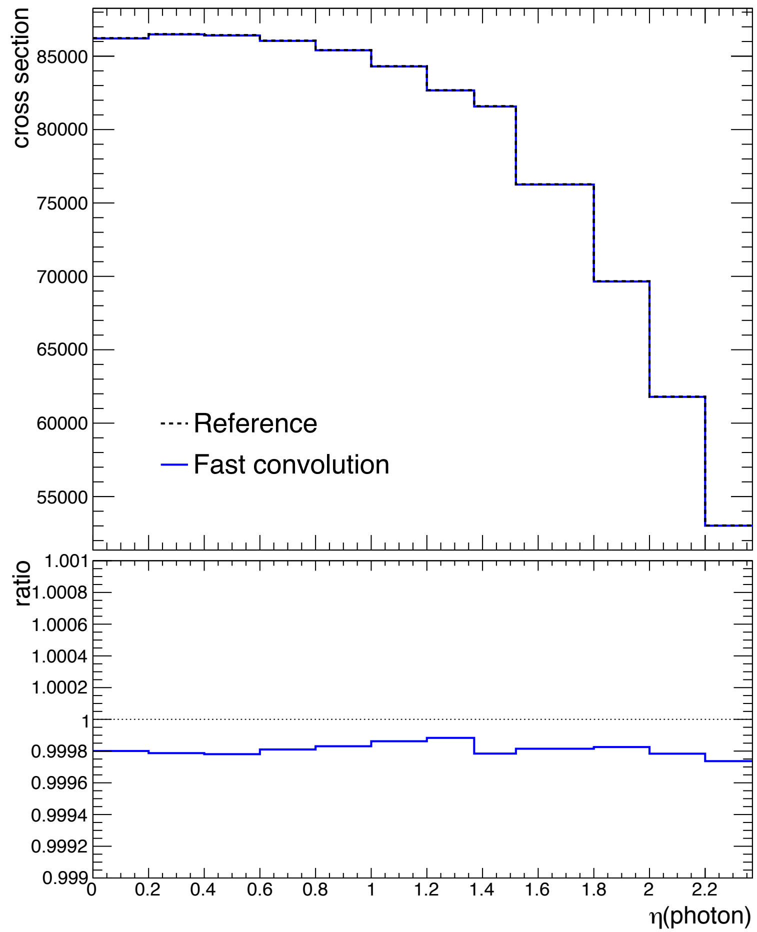
Just a selection of interesting processes ...

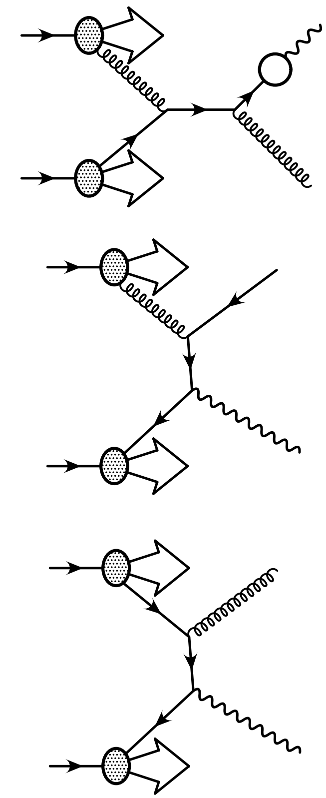


- W+charm production implemented and well tested
- Bare charm - for realistic comparison with experimental fiducial data some model of hadronisation is required









Prompt photon production

- Prompt photon production at NLO + LO fragmentation process
- Small non-closure at the level of 0.02%
- Some validation underway: NLO+NLO fragmentation contribution available from JetPhox - calculating LO → NLO fragmentation k-factors

Generic subprocess

- For any new subprocess, reducing the number of contributing processes from the most general (non-top) 11 × 11 processes is non trivial
    - Needs knowledge of which terms in the matrix elements are truly independent

- Gradually moving to a generic implementation of all PDF combinations which can be specified by simple configuration file, eg for W- production can have 6 different processes

$$
\begin{array}{llllllll}
1 & 6 & (d,\bar{c}) & (d,\bar{u}) & (s,\bar{c}) & (s,\bar{u}) & (b,\bar{c}) & (b,\bar{u}) \\
2 & 6 & (\bar{c},d) & (\bar{u},d) & (\bar{c},s) & (\bar{u},s) & (\bar{c},b) & (\bar{u},b) \\
3 & 5 & (d,g) & (u,g) & (s,g) & (c,g) & (b,g) & \\
4 & 5 & (\bar{d},g) & (\bar{u},g) & (\bar{s},g) & (\bar{c},g) & (\bar{b},g) & \\
5 & 5 & (g,d) & (g,u) & (g,s) & (g,c) & (g,b) & \\
6 & 5 & (g,\bar{d}) & (g,\bar{u}) & (g,\bar{s}) & (g,\bar{c}) & (g,\bar{b}) &
\end{array}
$$

- This can be represented in the configuration file simply as

```
-1
 0  6   1 -4   1 -2   3 -4   3 -2   5 -4   5 -2
 1  6  -4  1  -2  1  -4  3  -2  3  -4  5  -2  5
 2  5   1  0   2  0   3  0   4  0   5  0
 3  5  -1  0  -2  0  -3  0  -4  0  -5  0
 4  5   0  1   0  2   0  3   0  4   0  5
 5  5   0 -1   0 -2   0 -3   0 -4   0 -5
```

- Once a grid has been constructed using such a file, the information is encoded in the grid itself

- This information is produced **automatically** for aMC@NLO, so these tables are created in memory and used during grid construction so that the equivalent files need never be written

Generic subprocesses

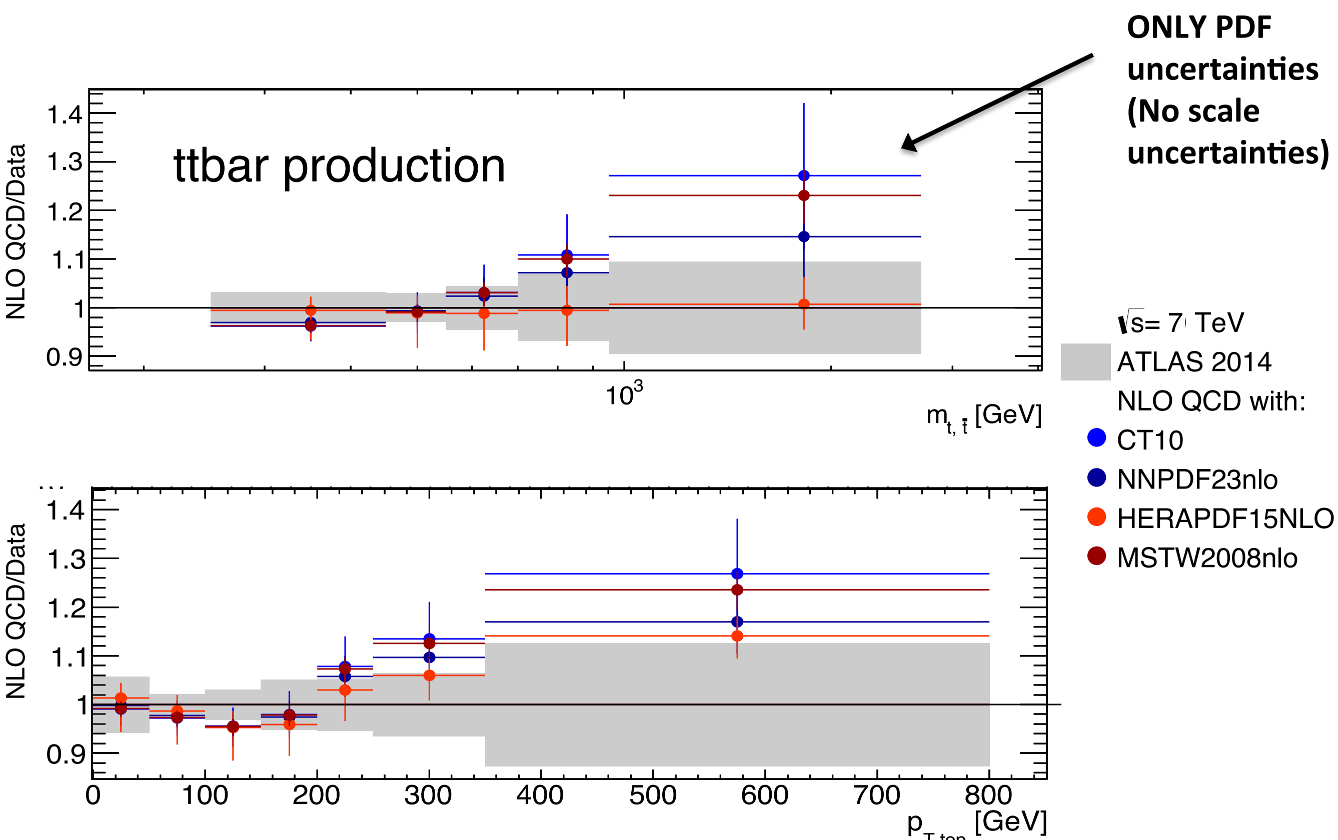
- For generic calculations, can use a representative **basic** configuration containing 11 × 11 (non-top) subprocesses
- This is rather **too large** for general use, so that we have a feature of the grids that can reduce this to the smallest number of unique subprocesses
- For example, for inclusive photon production with 121 processes, these can be reduced to 6 subprocesses for LO and 33 for NLO

```
0 6   -5 0  -3 0  -1 0  1 0  3 0  5 0
1 6   -5 5  -3 3  -1 1  1 -1  3 -3  5 -5
2 4   -4 0  -2 0  2 0  4 0
3 4   -4 4  -2 2  2 -2  4 -4
4 6   0 -5  0 -3  0 -1  0 1  0 3  0 5
5 4   0 -4  0 -2  0 2  0 4
```

```
0 3   -5 -5  -3 -3  -1 -1
1 6   -5 -4  -5 -2  -3 -4  -3 -2  -1 -4  -1 -2
2 6   -5 -3  -5 -1  -3 -5  -3 -1  -1 -5  -1 -3
3 3   -5 0  -3 0  -1 0
4 6   -5 1  -5 3  -3 1  -3 5  -1 3  -1 5
5 6   -5 2  -5 4  -3 2  -3 4  -1 2  -1 4
6 3   -5 5  -3 3  -1 1
7 6   -4 -5  -4 -3  -4 -1  -2 -5  -2 -3  -2 -1
8 2   -4 -4  -2 -2
9 2   -4 -2  -2 -4
10 2   -4 0  -2 0
11 6   -4 1  -4 3  -4 5  -2 1  -2 3  -2 5
12 2   -4 2  -2 4
13 2   -4 4  -2 2
14 3   0 -5  0 -3  0 -1
15 2   0 -4  0 -2
16 1   0 0
17 3   0 1  0 3  0 5
18 2   0 2  0 4
19 6   1 -5  1 -3  3 -5  3 -1  5 -3  5 -1
20 6   1 -4  1 -2  3 -4  3 -2  5 -4  5 -2
21 3   1 -1  3 -3  5 -5
22 3   1 0  3 0  5 0
23 3   1 1  3 3  5 5
24 6   1 2  1 4  3 2  3 4  5 2  5 4
25 6   1 3  1 5  3 1  3 5  5 1  5 3
26 6   2 -5  2 -3  2 -1  4 -5  4 -3  4 -1
27 2   2 -4  4 -2
28 2   2 -2  4 -4
29 2   2 0  4 0
30 6   2 1  2 3  2 5  4 1  4 3  4 5
31 2   2 2  4 4
32_2   2 4  4 2
```

- NB: This **may not** be the minimal number for which symmetries of the matrix elements could be used, but it is a significant reduction none-the-less

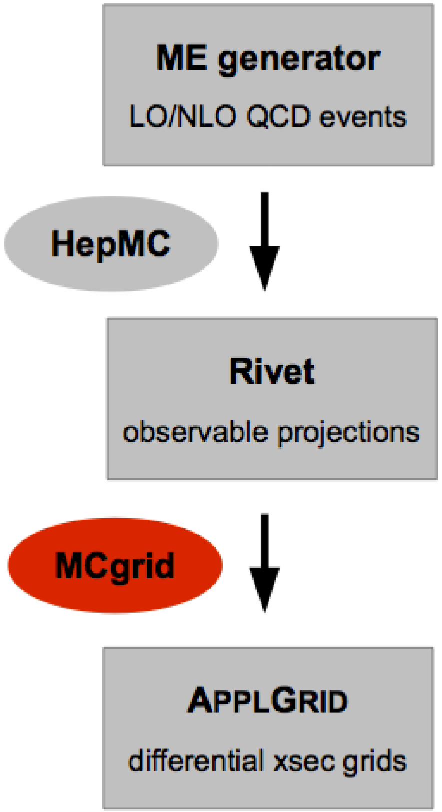
Sherpa - Native APPLgrid - Sherpa interface



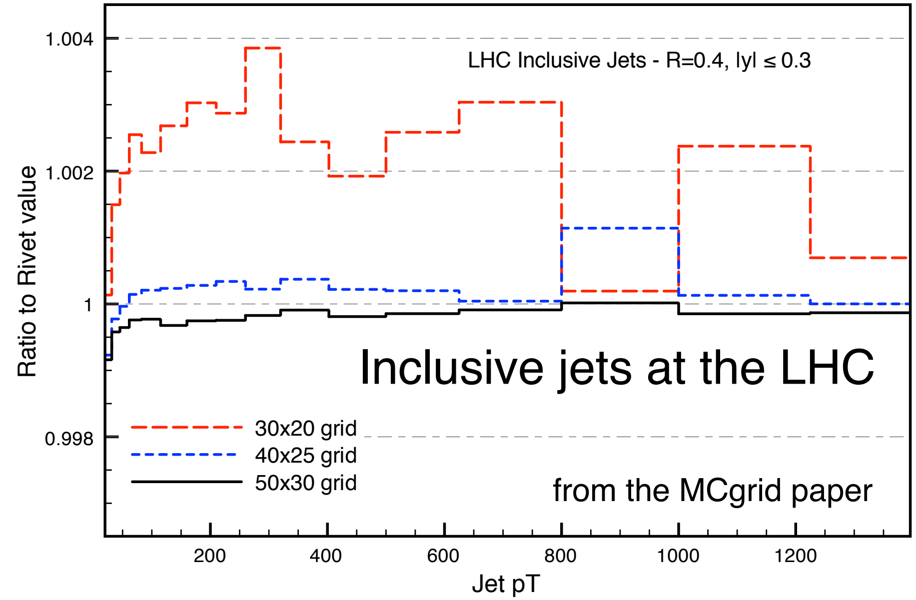
- Uses the native Sherpa output ntuple: Tancredi Carli, Cameron Embree MS
- Uses the generic process decomposition
- Can be used for any fixed order NLO sherpa process
- **Caveat:** when generating weights for individual parton-parton subprocesses, in order to get agreement with reference at low statistics, need to use the full 121 parton-parton luminosities
  - Imposing known symmetries in the matrix element to fill the grid with fewer subprocesses provides more precise grids, but at the cost of better than statistical agreement with the reference histogram at low statistics

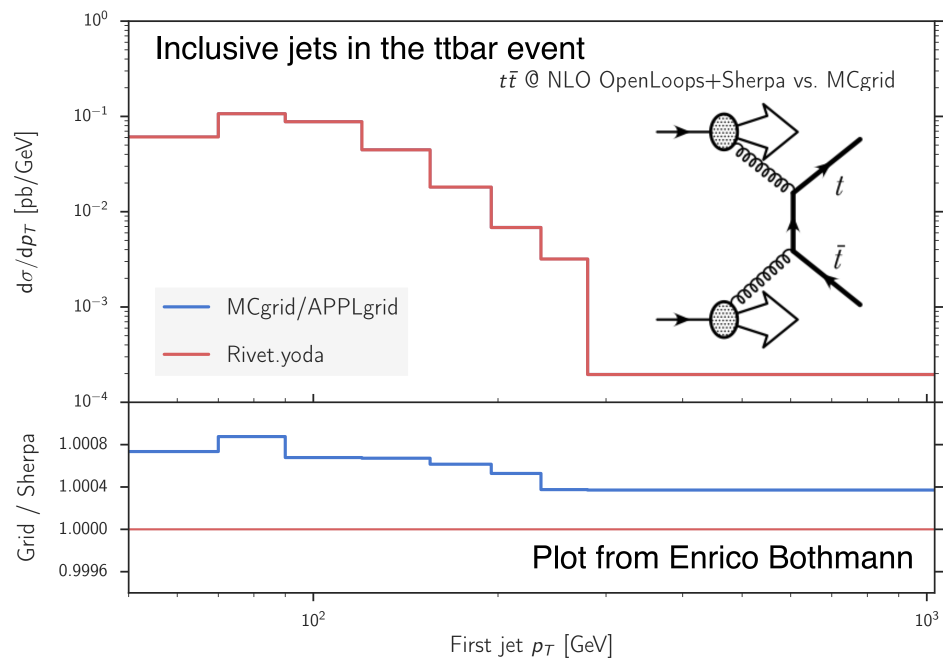
MCgrid Sherpa interface - Del Debbio et al

- Written by Del Debbio, Hartland and Schumann arXiv:1312.4460. http://mcgrid.hepforge.org
- Available for all fixed order NLO processes using the generic PDF decomposition



- MCgrid works as a Rivet plugin using the HepMC event record





aMCfast - A fast interface between MadGraph5_aMC@NLO and APPLgrid

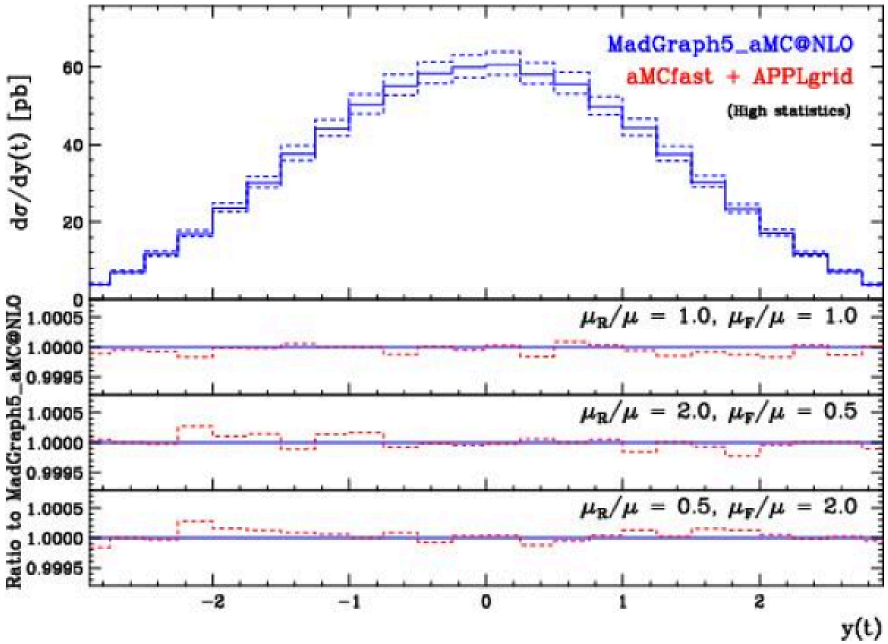
- aMCfast Home
- References
- Download and installation
- Instructions to run the code
- Analysis file examples
- Talks
- Contact

aMCfast [arXiv:1406.7693] is an *automated interface* which bridges the automated cross section calculator **MadGraph5_aMC@NLO** [arXiv:1405.0301] with the fast interpolator **APPLgrid** [arXiv:0911.2985].

The chain **MadGraph5_aMC@NLO – aMCfast – APPLgrid** will allow one to include, in a straighforward manner, any present or future LHC measurement in an NLO global PDF analysis.

The basic idea behind the use of these three codes is that of computing user-defined observables relevant to arbitrary processes, and to represent them in terms of look-up grids, which can be accessed at later times, and used to obtain predictions for such observables with any PDFs. This a-posteriori computation is both accurate and very fast. Contrary to other **APPLgrid** application, factorisation scale variation can be performed without linking to any third-party code.

The following representative figures show the rapidity of the top quark in top-pair production and the lepton-pair invariant mass in dilepton production in association with one jet at the 14 TeV LHC. For more details, please refer to the original publication.



- Valerio Bertone, Rik Frederix, Stefano Frixione, Juan Rojo, MS

aMCFast interface MC@NLO interface

- Implementation of aMC@NLO standalone calculations uses 4 sets of weights - one at tree level and three for the NLO part

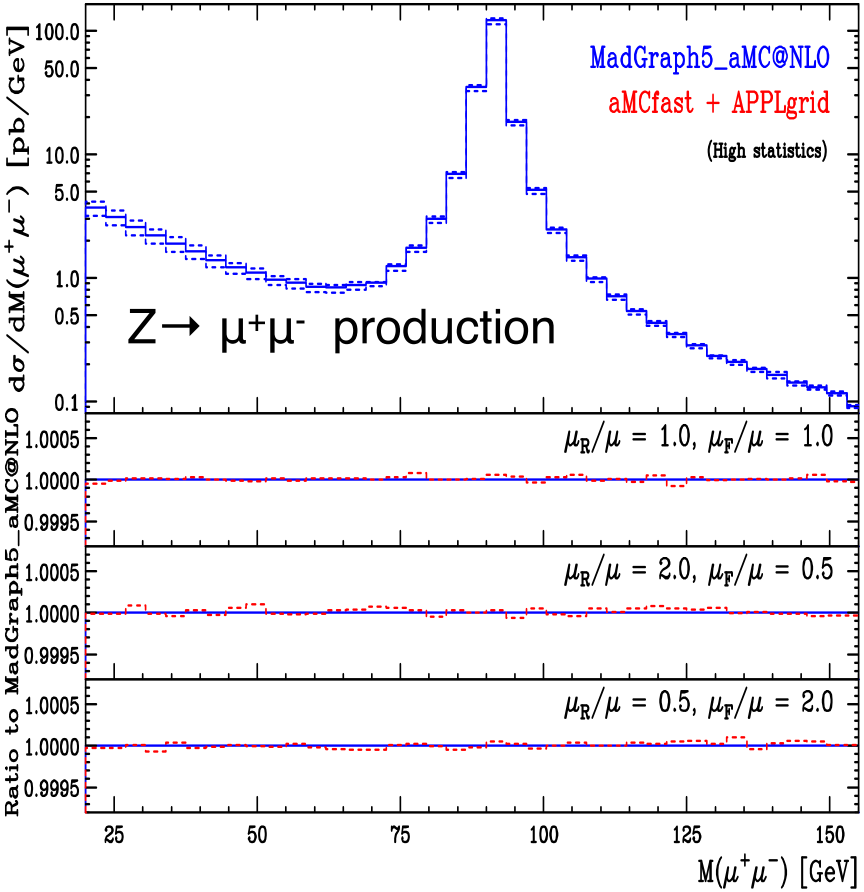
$$
\begin{aligned}
d\sigma \quad = \quad \alpha_s(Q)^n \Bigg[W^B + \\
\alpha_s(Q) \left[W^0 + W^R \log\frac{\mu_R}{Q} + W^F \log\frac{\mu_F}{Q} \right] \Bigg]
\end{aligned}
$$

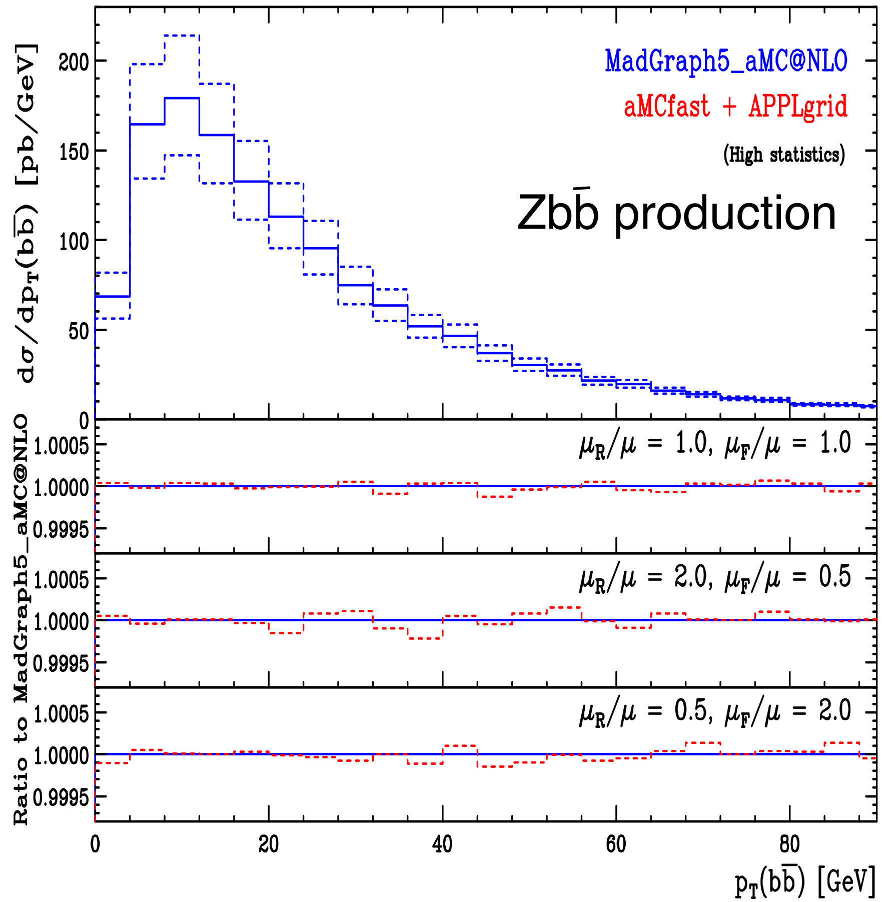
- Minor internal modifications to the grid class to make use of additional weight grids

- Implements all fixed order NLO calculations within aMC@NLO
- Using generic subprocesses automatically configured using the information provided by aMC@NLO and the new "MCFM" interface paradigm discussed earlier

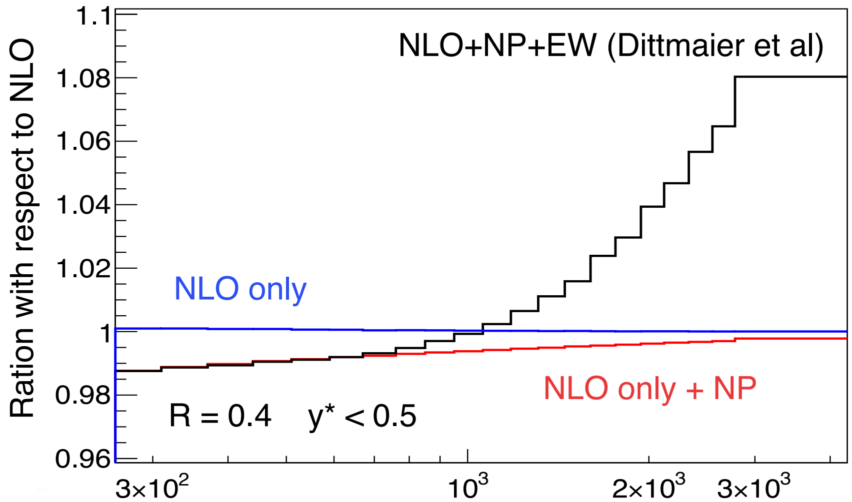
- Makes extensive use of the grid combination utilities to combine grids after generation

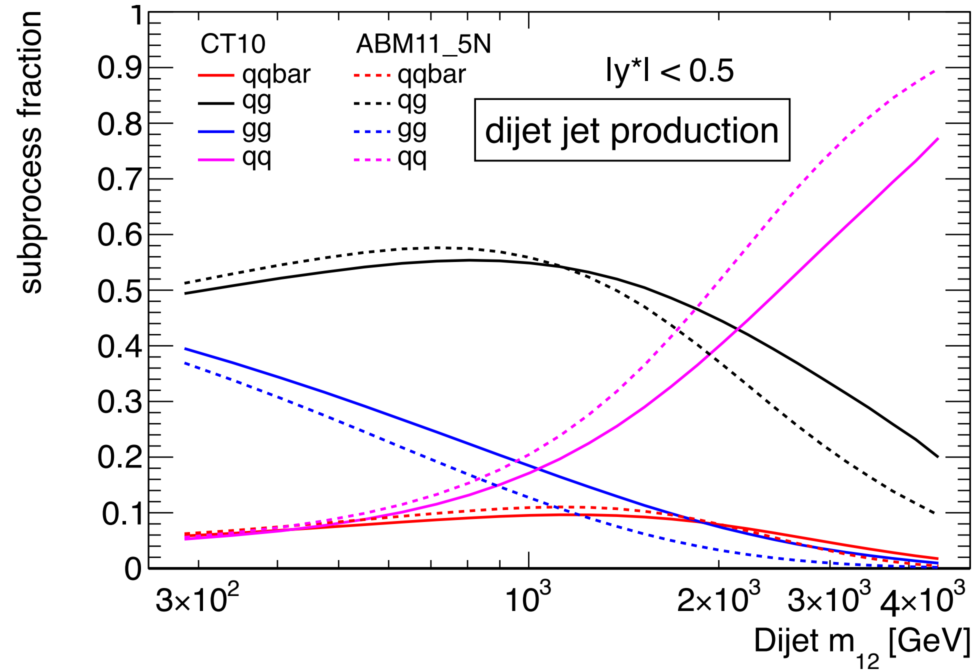
aMCfast interface to aMC@NLO

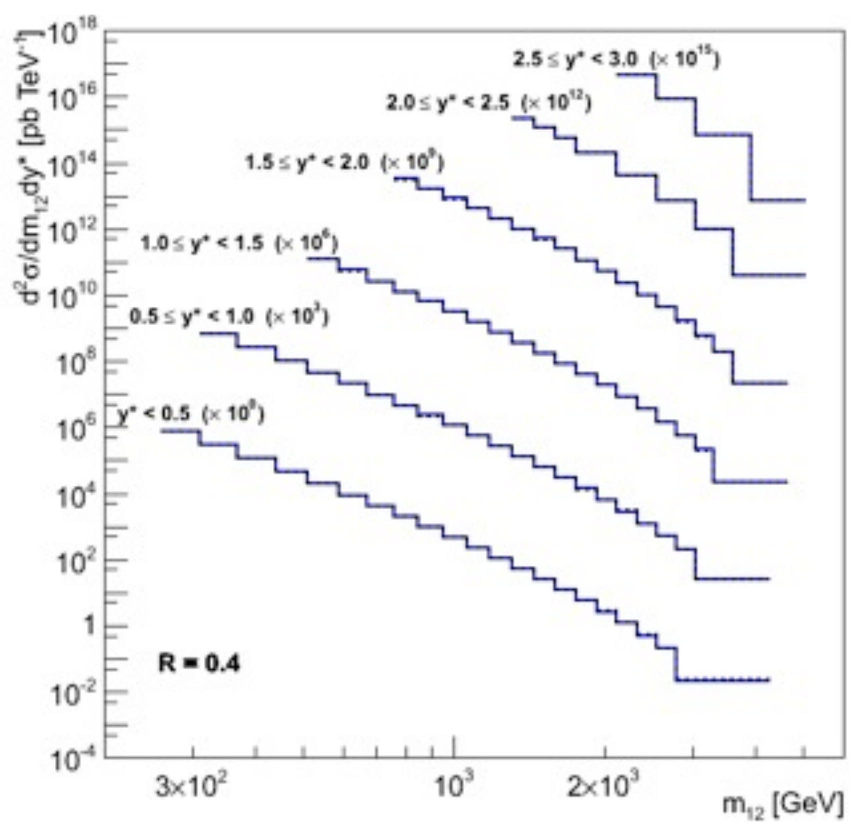




- More details in presentation from Valerio

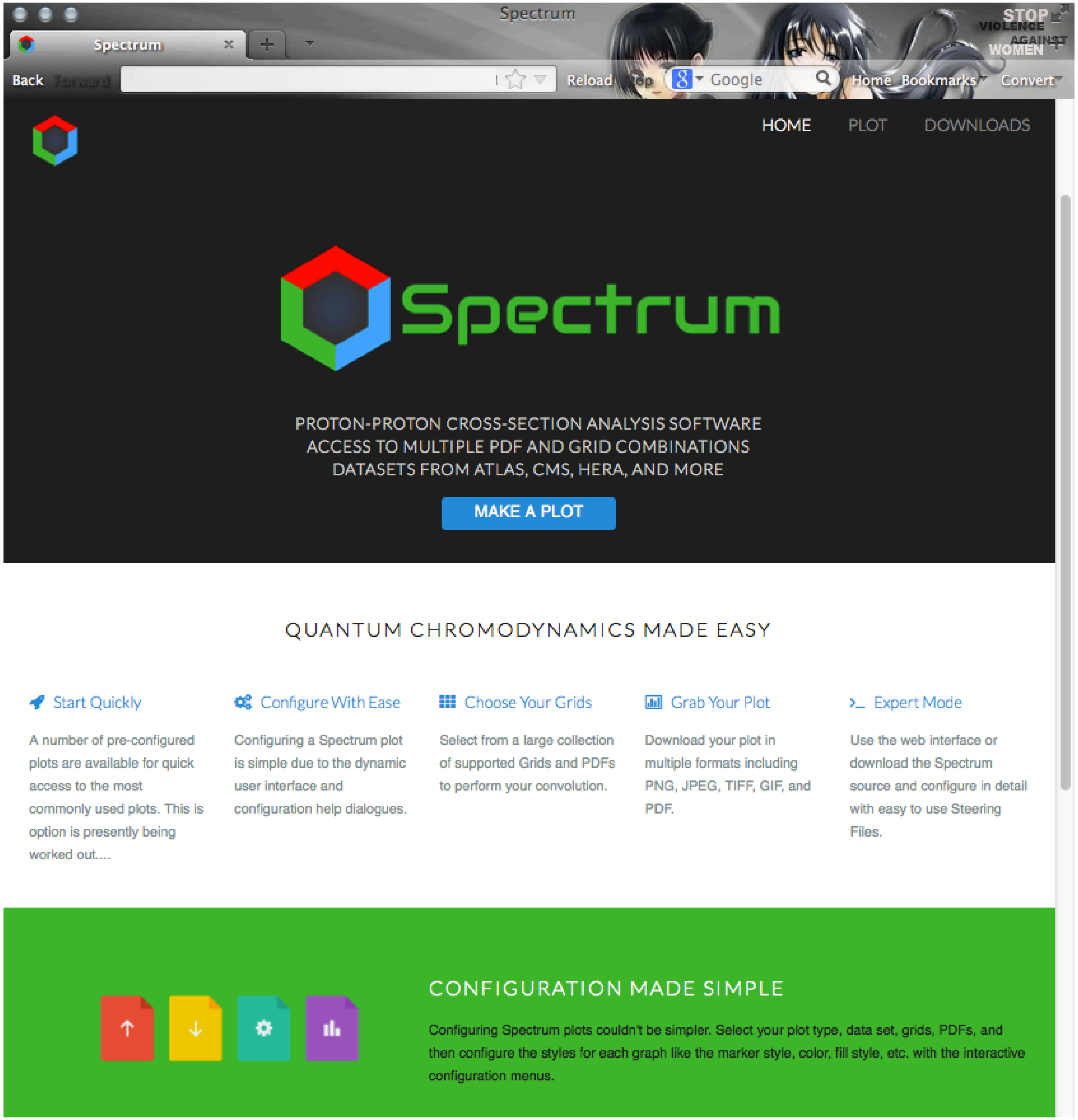






Multiplicative corrections: Example dijet production

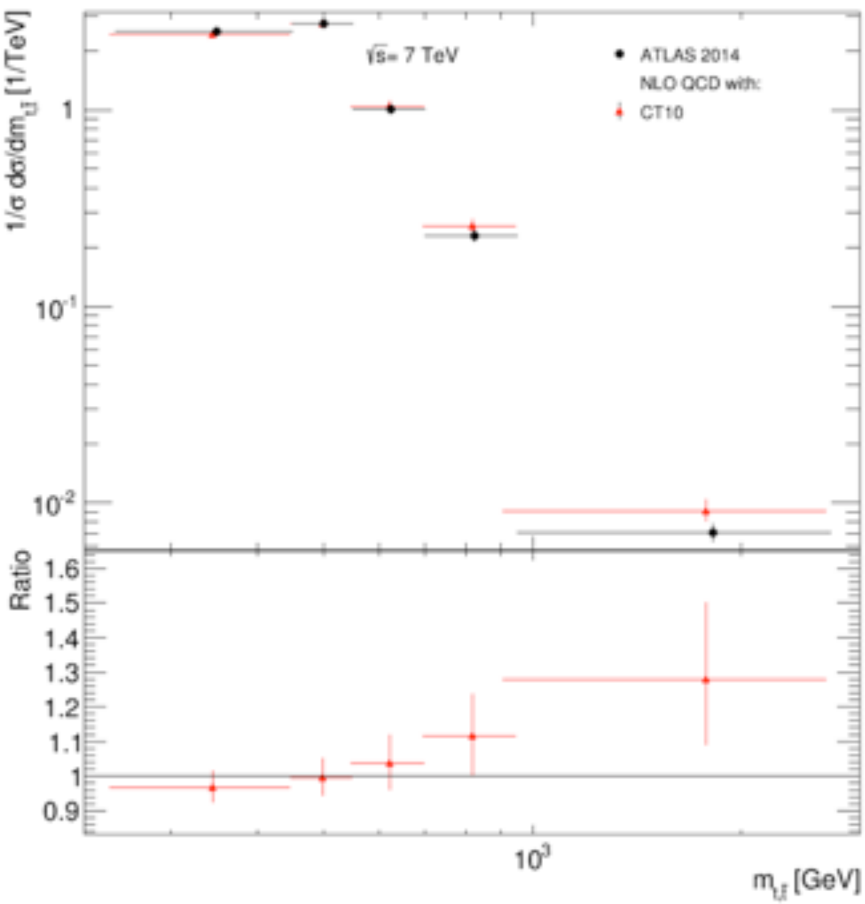
- Grids can include eg non-perturbative and EW corrections
- Corrections can be applied independently, or in any combination
- Single multiplicative terms for any give correction is not ideal
    - Move towards subprocess dependent corrections
    - Although not ideal - better dependence than current single k-factor approach

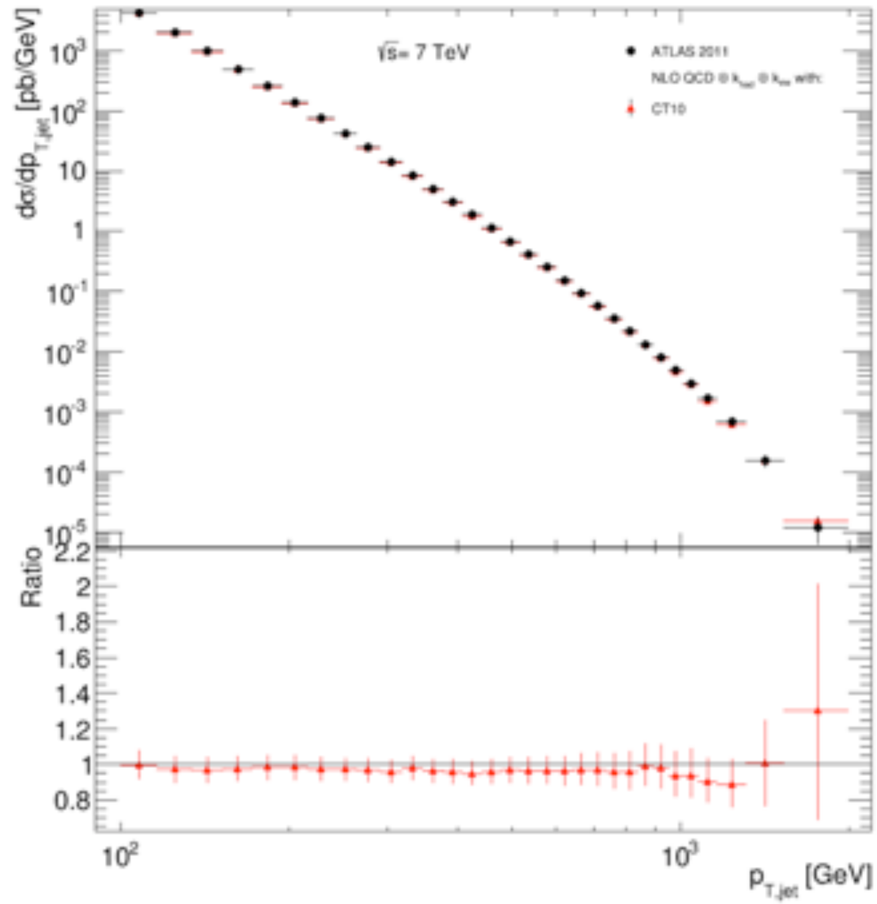


- New plotting web utility for simpler production of comparisons of data with calculations code

- Still under development but should be available soon
- From Tancredi Carli and Joe Gibson

Spectrum plotter





- Several sets of cross section data and grids are available
  - Compare data and calculation with any choice of PDF, with or with out hadronisation, EW corrections etc
  - Aim is to try to collect all available processes and grids - grids will be downloadable, with appropriate authorship attribution etc, so please donate your grids!

The Holy Grail - implementing parton showers

- Clearly it would be beneficial to be able to include the effects of parton showers into the grids
  - Routinely use NLO + parton shower (aMC@NLO, Sherpa etc) for model predictions at the LHC
  - Including the effects of hadronisation would be **extremely useful** for including eg tagged W+charm data without the inherant limitations of applying bin-by-bin hadronisation corrections

- Final state showers should cause no problems - independent of the incoming PDF
- The issue lies in the initial state shower - since both ends of the shower are fixed, replace the Sudakov probability for no emission in the backwards evolution

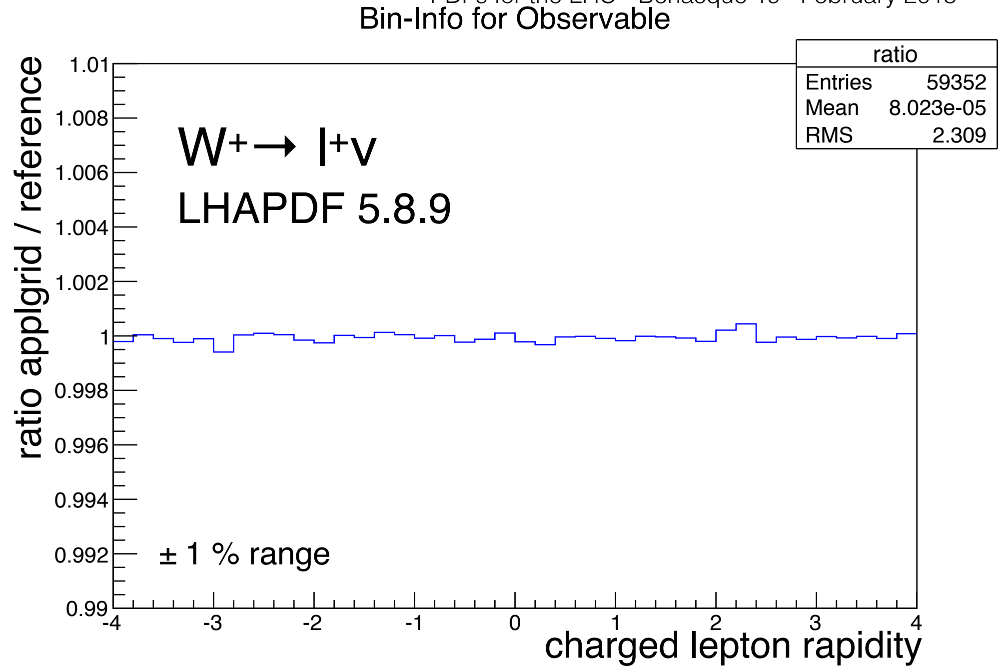
$$\Delta_i(Q^2, q^2) \rightarrow \frac{\Delta_i(Q^2, q^2)}{f_i(x, q^2)}$$

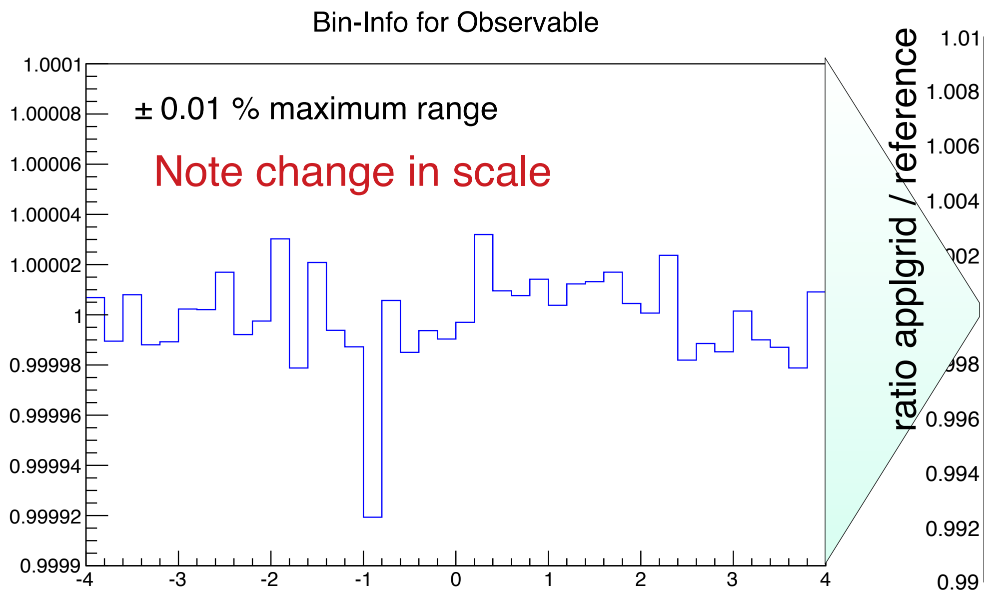
  - This introduces an explicit dependence of the PDF in the initial state parton shower - if you change the PDF, you change the probability for emission, so that when filling the grid, the number of branches that might have taken place
- Often the PDF used for (leading log) shower is fixed, and not the same as for the cross section calculation
  - This case should be easy to reproduce

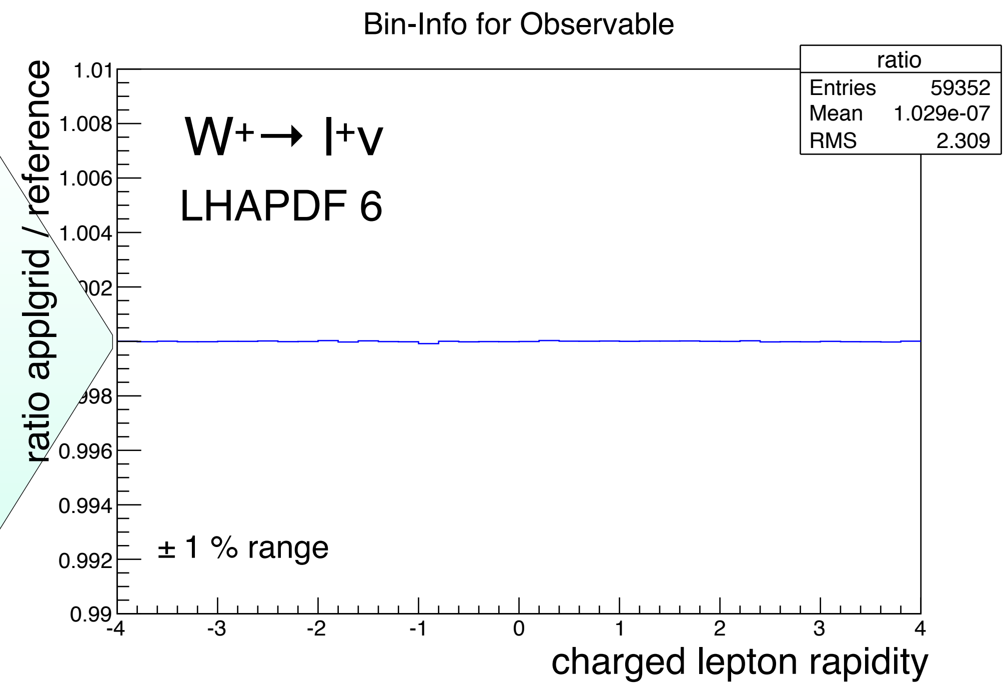
- The Holy Grail of the shower grid implementation is to accurately reproduce the calculation when the PDF is changed in the shower
- Efforts ongoing in both MCgrid and aMCfast

A brief diversion on using LHAPDF 6

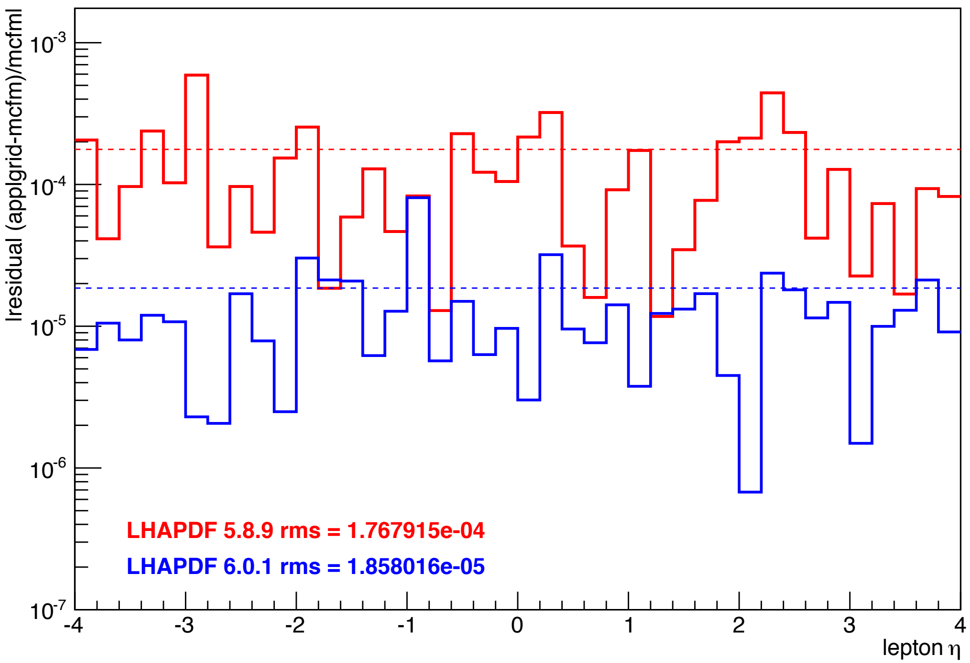
- Developing an interface for grids to native LHAPDF 6
- During testing, compare convolution with old and new version of LHAPDF
    - Convolutions with LHAPDF 6 have significantly smaller interpolation errors, generally better than ~ 0.003 % (cf 0.03% with 5.8.9)
    - Size of fluctuation previously attributed to APPLgrid interpolation may in fact have been from LHAPDF itself !







How much better is the convolution using LHAPDF 6?



- Take the absolute value of the residual, calculate the rms of each convolution
- LHAPDF 6 about **10 times** smaller residual than 5.8.9

- Sadly, because of the lack of a true interface for the LHAPDF 6 PDF.h class, the native C++ PDF classes can't be used
  - Still need to use the lhaglue FORTRAN based interface routines

Towards a native LHAPDF 6 interface

- A native interface to LHAPDF 6 would be extremely useful
    - Would allow fitting with a photon density using the new interface etc - at the moment the LHAPDF 5 based interface makes extensions of this nature somewhat non-trivial

- The PDF.h header defines a de-factor standard for the new PDF interface
- Unfortunately this is not an interface class, not does it inherit from an interface class
    - Includes code and dependencies for the full PDF constructor, _loadInfo(const std::string& mempath) etc, none of which are needed, or desirable in an interface
    - Using the existing class as an interface, forces the user of PDF.h to link against the full LHAPDF library, even if they do not want to use the LHAPDF code directly - perhaps they want to define their own PDF class for use in a PDF fit and don't want the LHAPDF constructor baggage

- It would be trivial, and have **absolutely no consequences** for any existing code, for PDF.h to inherit from a virtual abstract interface, say, iPDF.h, containing pure virtual interface functions for the routines needed for the PDF sampling
- This would enable other classes to use the same interface without needing to implement all the internals of the PDF class, without having to link to LHAPDF etc.
- Without this is is largely impossible to use the de-facto LHAPDF 6 interface for anything over than calling a PDF from LHAPDF

- I propose a new interface package where we can collect such interfaces for interchangeability of different tools to avoid explicit dependencies on specific tools

Forwards ...

- The APPLgrid convolution interface has been growing for some time
- It is a very rich interface, but is somewhat unwieldy
- We are currently discussing a redesign of the convolution interface to greatly simplify the calling structure
    - Current thoughts are with allowing the passing of a simple Configuration object

```
appl::grid g("grid.root");
TH2D* xsec = g.Convolute( pdf, appl::config( nloops, muf, mur ) );
```

    - The current plethora of different calling routines being replaced by different derived instances of this basic configuration class
- Intend to retain the, very desirable, **PDF-agnostic** state of the grids

- Currently, we have a lightweight filling interface …

```
// update grid with one set of event weights
void fill(const double x1, const double x2, const double Q2,
          const double obs,
          const double* weight, const int iorder);


void fill_phasespace(const double x1, const double x2, const double Q2,
                     const double obs,
                     const double* weight, const int iorder);
```

- We would like to extend this interface by using a more generic weight structure ...

```
void fill( const grigInfo& gi );
```

- In an ideal world, this new *gridInfo* structure would be part of a wider scheme for simplifying the implementation of new calculation interfaces

Forwards - standardised interface weight class

- Life cycle of implementing a calculation in a grid …
    1. Reverse engineer the internal convolution code fragment from any calculation
    2. Create of expose the native internal storage structures used by this code fragment
    3. Create hook functions which book and fill the grid
    4. in the hook functions, extract the information from the exposed native internal storage and call the grid filling routines

- The grid filling routines are typically trivial - usually only ~ 5 functions that need to be called in total
- Essentially all the effort is in understanding the internal workings of the calculation code and, extracting the information from the internal storage structures

- So far the issue in the bridge classes is that the internal storage needs to be exposed and the custom hook functions need to be implemented
    - These are generally, by definition unique to the calculation code, but the information needed by the grid is in a standard form

- Having a **genuinely standard form** for the exchange of the weight information would greatly simplify the implementation of any calculation, and replace the above stages to
    - Calculation author, can additionally fill the **standard form** structure

- This would be essentially the same as the approach using HEPMC in MCgrid, and the native APPLgrid interface implementation using the Sherpa ntuple, but with a more standard structure

Minimal elements of any standard structure

- As a basis such a structure need only store a single weight for a single sub process …
    - subprocess id (ie 0 .. 121, or some smaller number using eg the generic PDF combination as a mapping)
    - order in alphas
    - index of grid to fill
    - kinematic variables, x1, x2, Q2, …
    - final state quantities - either variables themselves (or 4 vectors of outgoing hadrons, partons etc)

- Other bookeeping information..
    - values of the PDFs used for this specific weight
    - alphas value, scales etc
    - …

- Such an interface would of course require an equivalent FORTAN interface

- I would like to suggest that, as a community, we should try to define such a light weight data structure to facilitate the more automatic implementation of new and existing calculations within the grid tools that we have available

Outlook

- The APPLgrid project is now reasonably mature with an increasingly extensive portfolio of cross section processes available
- An increasingly large user base is developing for both the fast convolution code and grid generation
- Developments are underway for interfaces with new calculation code
  - Cross community discussions might be useful for more extensive and rapid progress
- We are always open to new suggestions or requests - particularly if people want to contribute to the project