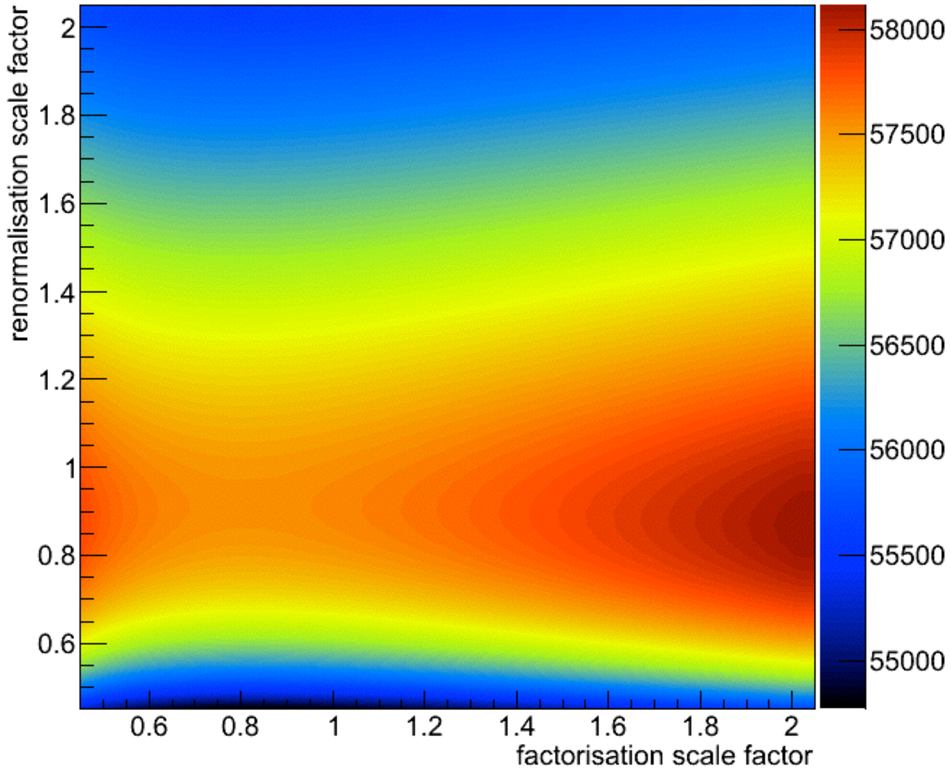
Recent developments with *APPLgrid*

Mark Sutton
The University of Sussex

Stefano Carrazza
CERN

What is APPLgrid ?

- APPLgrid is a fully open source package to build a library of C++ utility classes for performing fast convolutions with PDFs up to NNLO
- It uses a customised grid storage for reduced memory foot print
- Interfaced to a range of calculations - NLOjet++, MCFM, Sherpa, aMC@NLO, NNLOJET …
  - Perform the full calculation once - store the weights, performa the convolution with PDFs afterwards in only a few milliseconds
- Can be used for fast cross section production with pre-existing grids
  - Arbitrary renormalisation and factorisation scale variation
  - **Arbitrary beam-energy rescaling**
  - **Different PDF set for each target hadron**
  - Any number of multiplicative corrections can be stored and applied in the the grid
  - Allows different input PDFs for each incoming hadron
  - **Can include photon density**



- **Current full featured development version**
  - **applgrid-1.5.30**
  - **Download from https://applgrid.hepforge.org/downloads/applgrid-1.5.30.tgz**

Recap of the Numerical Technique

- For a calculation of a cross section from $m = 1 \ldots N$ weights, w_m , from a Monte Carlo integration with momentum fraction x_m, form the product

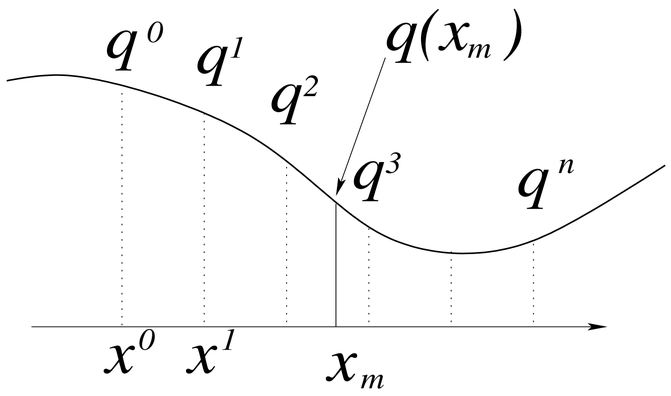
$$\sum_m w(x_m) q(x_m)$$

- Can interpolate the function $q(x_m)$ ...

$$q(x_m) \approx \sum_i q^{(i)} I^{(i)}(x_m - x^{(i)})$$

- such that

$$\begin{aligned} \sum_m w(x_m) q(x_m) &\approx \sum_i q^{(i)} \sum_m w(x_m) I^{(i)}(x_m - x^{(i)}) \\ &\approx \sum_i q^{(i)} W^{(i)} \end{aligned}$$



- For a calculation of a cross section with m = 1 .. N weights, from a Monte Carlo integration with momentum transfer Q^2

$$\begin{aligned} d\sigma &= \sum_p \sum_{m=1}^{N} w_m^{(p)} \left(\frac{\alpha_s(Q_m^2)}{2\pi} \right)^p q(x_m, Q_m^2) \\ &= \sum_p \sum_{ij} q(x_{(i)}, Q_{(j)}^2) \left(\frac{\alpha_s(Q_{(j)}^2)}{2\pi} \right)^p \sum_m^N w_m^{(p)} I_i^x(x_m) I_j^{Q^2}(Q_m^2) \\ &= \sum_p \sum_{ij} q(x_{(i)}, Q_{(j)}^2) \left(\frac{\alpha_s(Q_{(j)}^2)}{2\pi} \right)^p W_{ij}^{(p)} \end{aligned}$$

proton-proton Collisions

- For pp collisions need an extra dimension for the PDF of the second colliding hadron

$$d\sigma = \sum_p \sum_{m=1}^{N} w_m^{(p)} \left(\frac{\alpha_s(Q_m^2)}{2\pi} \right)^p q_1(x_{1m}, Q_m^2)\ q_2(x_{2m}, Q_m^2)$$

- But there is an implicit summation over parton flavours. Make use of symmetries in the matrix elements to use a vector of $k = 1 \ldots M$ independent weights such that

$$\sum_{ij=q,\bar{q},g} w_{ij} q_{1i}(x_1) q_{2j}(x_2) = \sum_{k=1}^{M} w^{(k)} F^{(k)}(x_1, x_2)$$

- such that

$$d\sigma = \sum_p \sum_{k=1}^{M} \sum_{m=1}^{N} w_m^{(p)(k)} \left(\frac{\alpha_s(Q_m^2)}{2\pi} \right)^p F_m^{(k)}(x_{1m}, x_{2m}, Q_m^2)$$

- Which can be placed on a grid in the same way as for DIS

- So from the summation, everything is down to the quality of the interpolation of the pdf at the grid nodes
  - It is s pure quadrature technique and is not, in principle subject to statistical fluctuation, or put another way …
  - … each individual weight gets added to the grid, and should be well approximated individually

applgrid.hepforge.org

XXVII International Workshop | LHAPDF: PDF sets – Hepforge | applgrid – Hepforge

Secure https://applgrid.hepforge.org

applgrid is hosted by Hepforge, IPPP Durham

the APPLgrid project

Home | Downloads | Documentation | Subversion | Nightlies | News | Links

Convolution Code Download
Current version applgrid-1.5.15
Basic example code here
Standalone convolution utility here
The hoppet code version 1.2.0 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 inclusive jets (2011 - 4.5fb^{-1}) grids
ATLAS W+, W- data (2010 - 35pb^{-1}) grids
ATLAS Z0 data (2010 - 35pb^{-1}) grids
ATLAS W+, W- data (2011 - 4.6fb^{-1}) grids
ATLAS Z0 data (2011 - 4.6fb^{-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

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, Claire Gwenlan and Gavin Salam - send mail to the authors: applgrid @ projects.hepforge.org: Last updated Sun 9 Dec 2018 16:53:21 CET

Photon induced processes

- Modifications to allow the photon to be treated like any other parton in the proton
- Currently use a modification of the lhaglue type LHAPDF interface …

```
extern "C" void evolvepdf(const double& x, const double& Q, double* xfx) {
   xfx += 6;
   for (int i = -6; i < 7; i++) xfx[i] = pdf->xfxQ(i, x, Q);
   xfx[7] = pdf->xfxQ(22, x, Q);
}
```

- Use native LHAPDF 6 calls with call to access photon density, in lhaglue type wrapper

- APPLgrid intrinsic `lumi_pdf` class allows additional parton - 7 or 22 for the photon, for example …

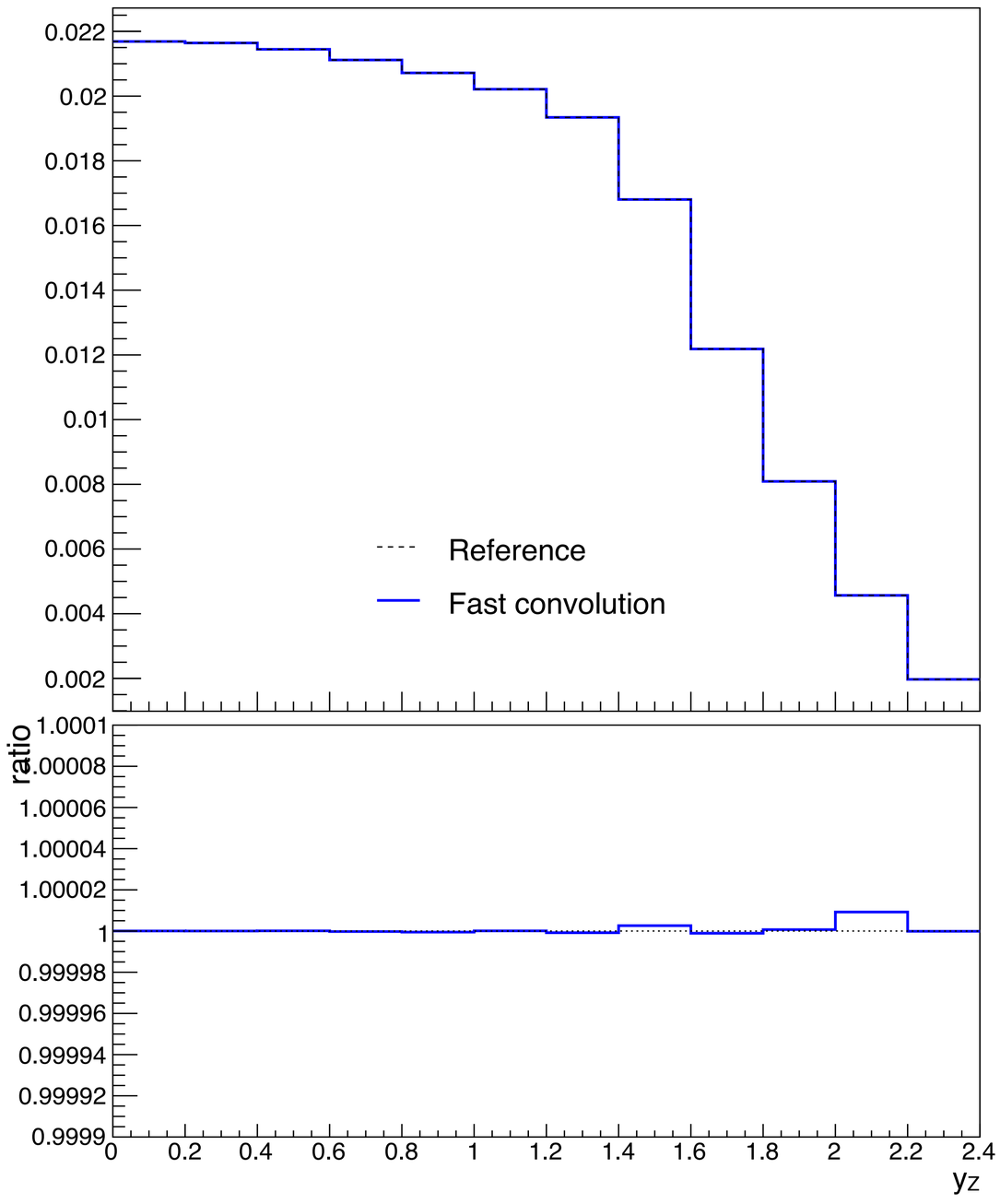
```
 0
 0 2    2 -2   4 -4
 1 2    0 -4   0 -2
 2 2    2 0   4 0
 3 3    1 -1   3 -3   5 -5
 4 3    0 -5   0 -3   0 -1
 5 3    1 0   3 0   5 0
 6 2    -4 4   -2 2
 7 2    0 2   0 4
 8 2    -4 0   -2 0
 9 3    -5 5   -3 3   -1 1
10 3    0 1   0 3   0 5
11 3    -5 0   -3 0   -1 0
12 1    7 7
```

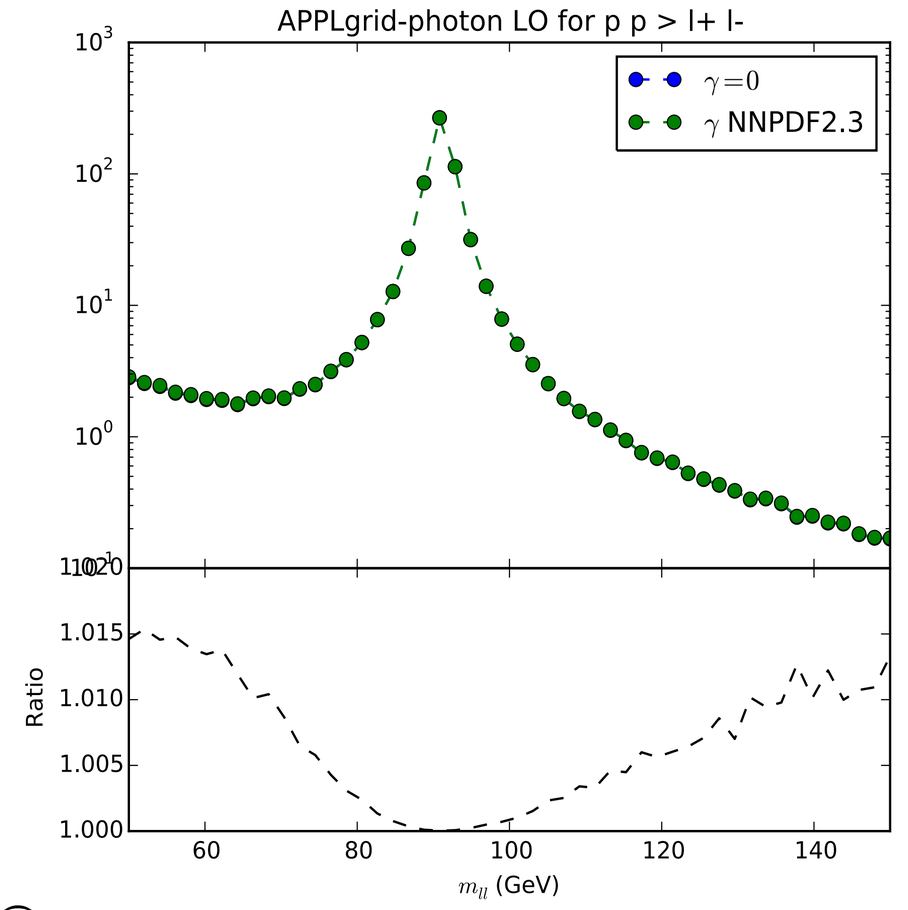
pdf : config nprocesses: 13

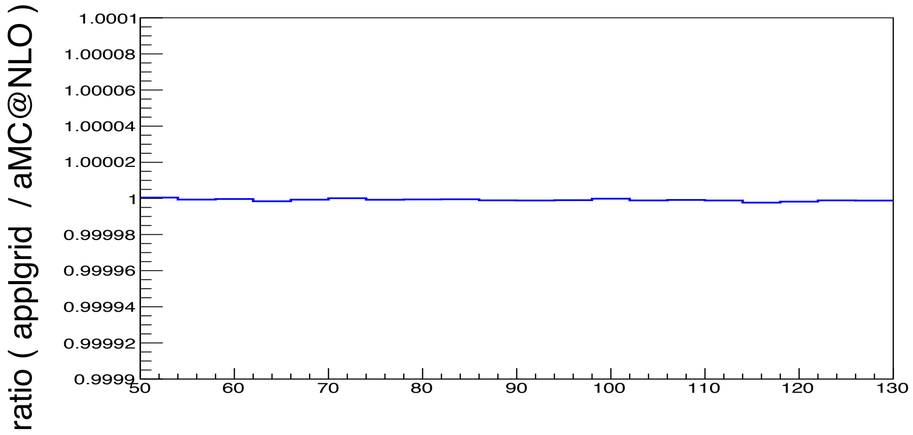
0	0	$(u, \bar{u}) + (c, \bar{c})$
1	1	$(g, \bar{c}) + (g, \bar{u})$
2	2	$(u, g) + (c, g)$
3	3	$(d, \bar{d}) + (s, \bar{s}) + (b, \bar{b})$
4	4	$(g, \bar{b}) + (g, \bar{s}) + (g, \bar{d})$
5	5	$(d, g) + (s, g) + (b, g)$
6	6	$(\bar{c}, c) + (\bar{u}, u)$
7	7	$(g, u) + (g, c)$
8	8	$(\bar{c}, g) + (\bar{u}, g)$
9	9	$(\bar{b}, b) + (\bar{s}, s) + (\bar{d}, d)$
10	10	$(g, d) + (g, s) + (g, b)$
11	11	$(\bar{b}, g) + (\bar{s}, g) + (\bar{d}, g)$
12	12	(γ, γ)

- Subsequent plots, treat the photon induced as a separate process with it's own grid containing just the PI contribution

Photon induced processes

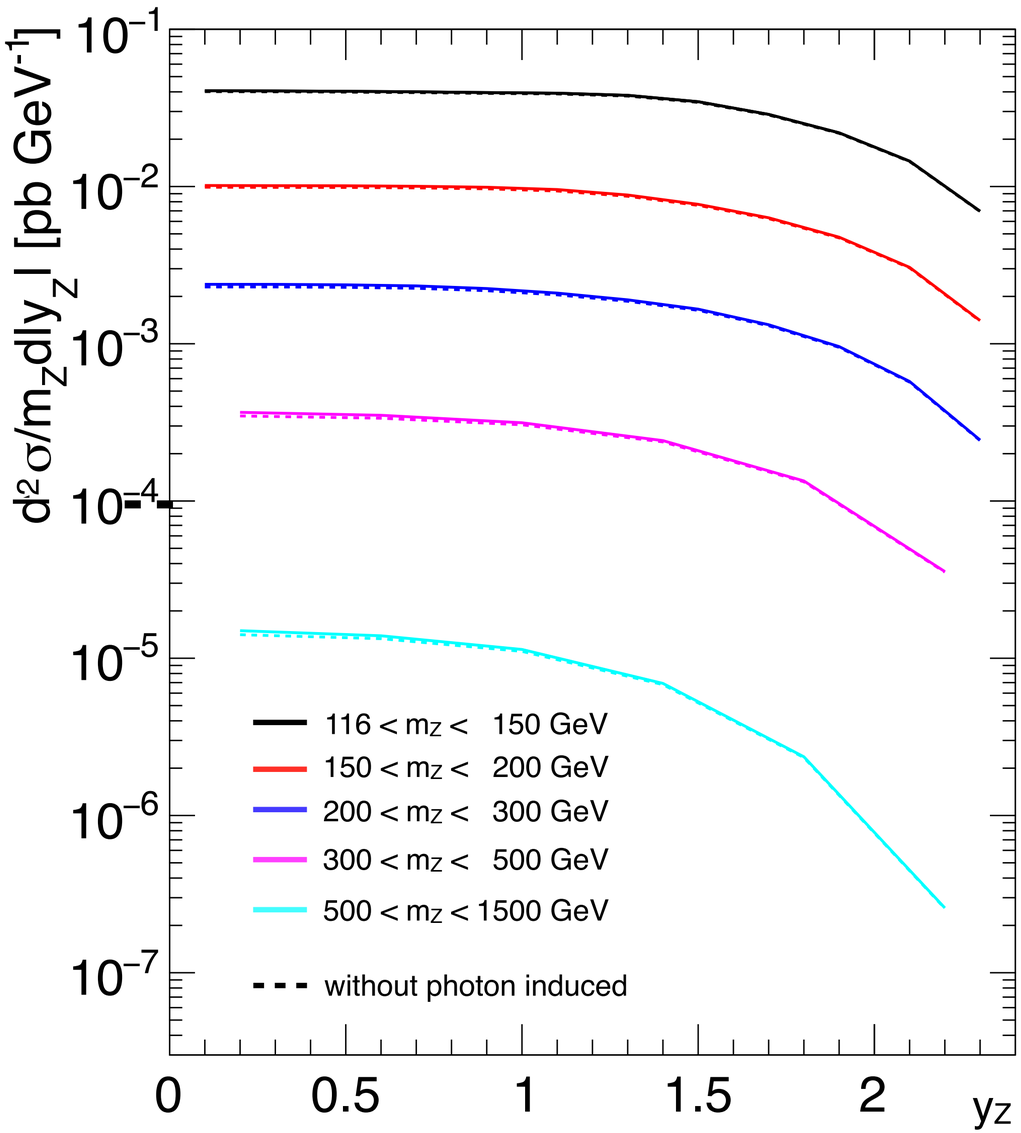


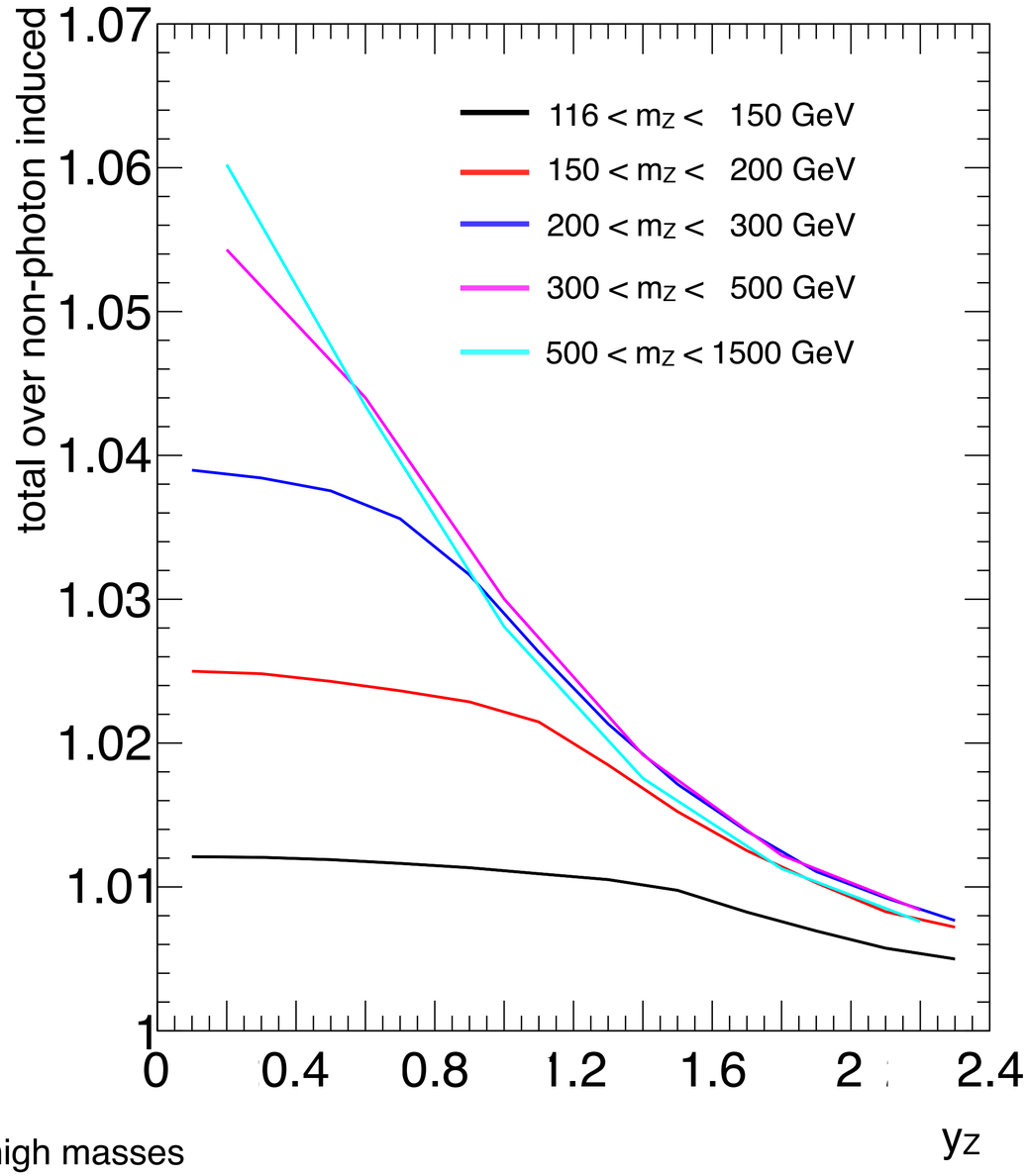




- Photon contribution to the proton
- Original implementation in applgrid private release, some time ago
  - Now code fully implemented into main development branch …

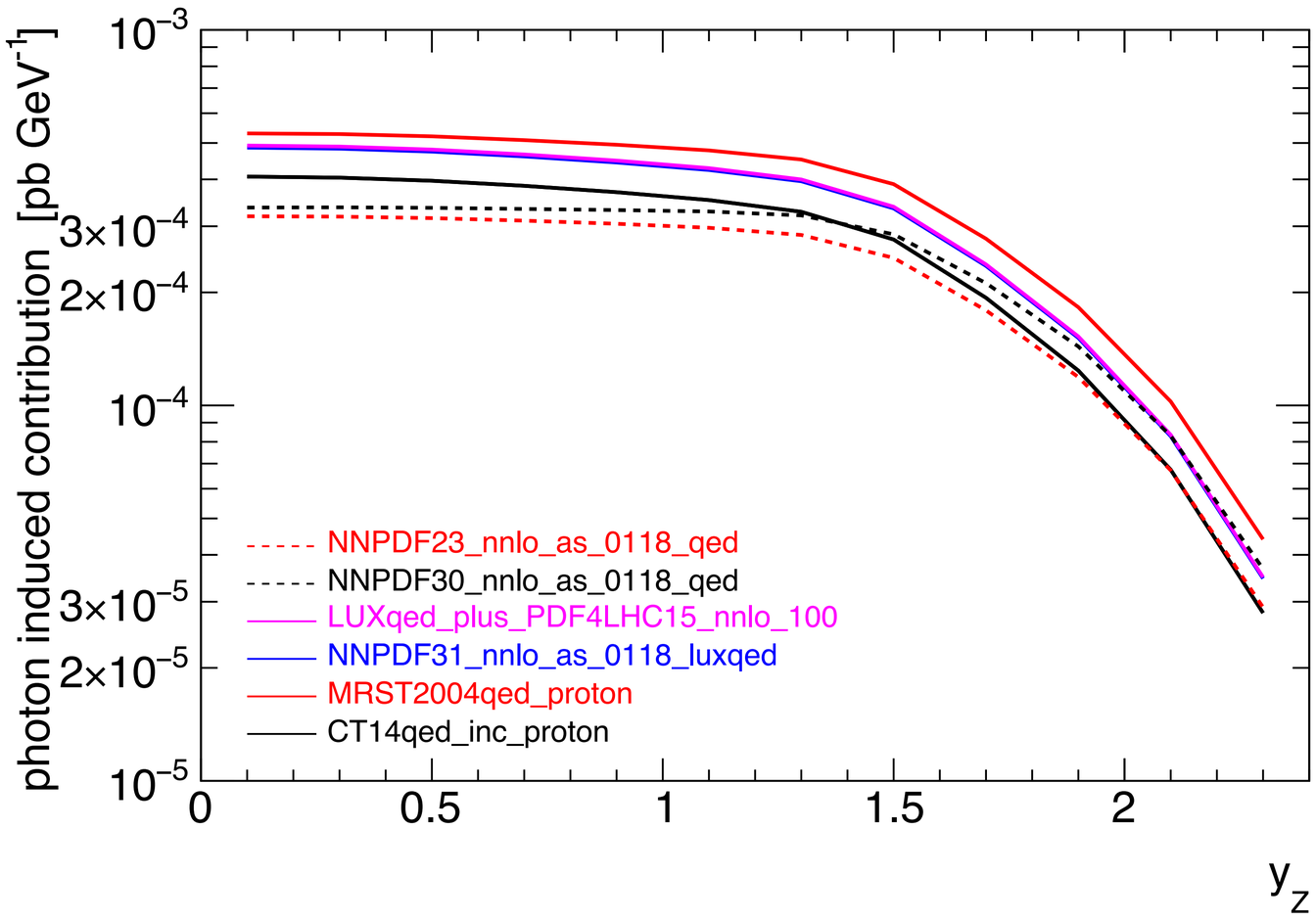
Photon induced contributions

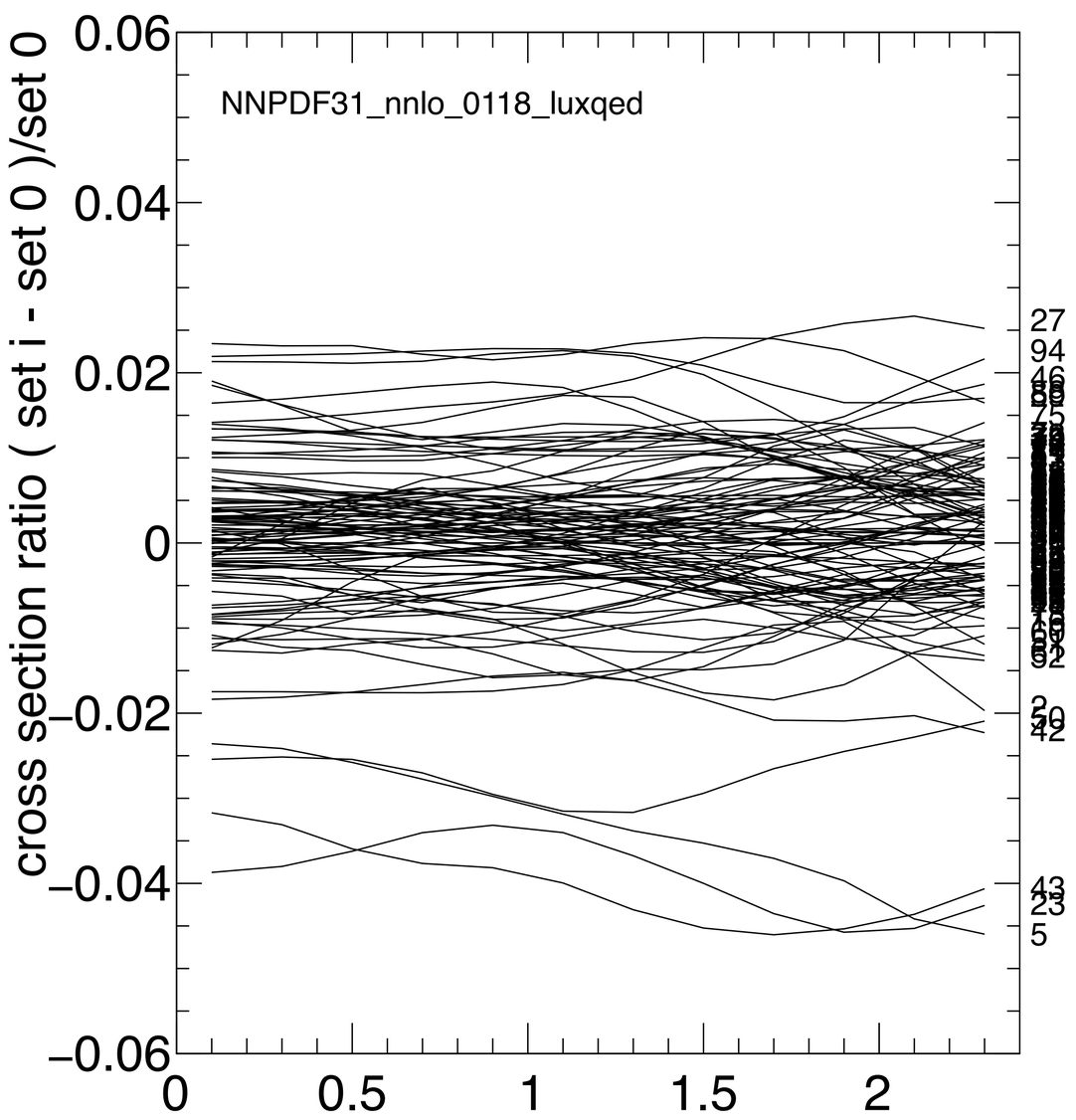




- Photon induced contribution is small, but can be several percent at high masses

Example with different PDFs …



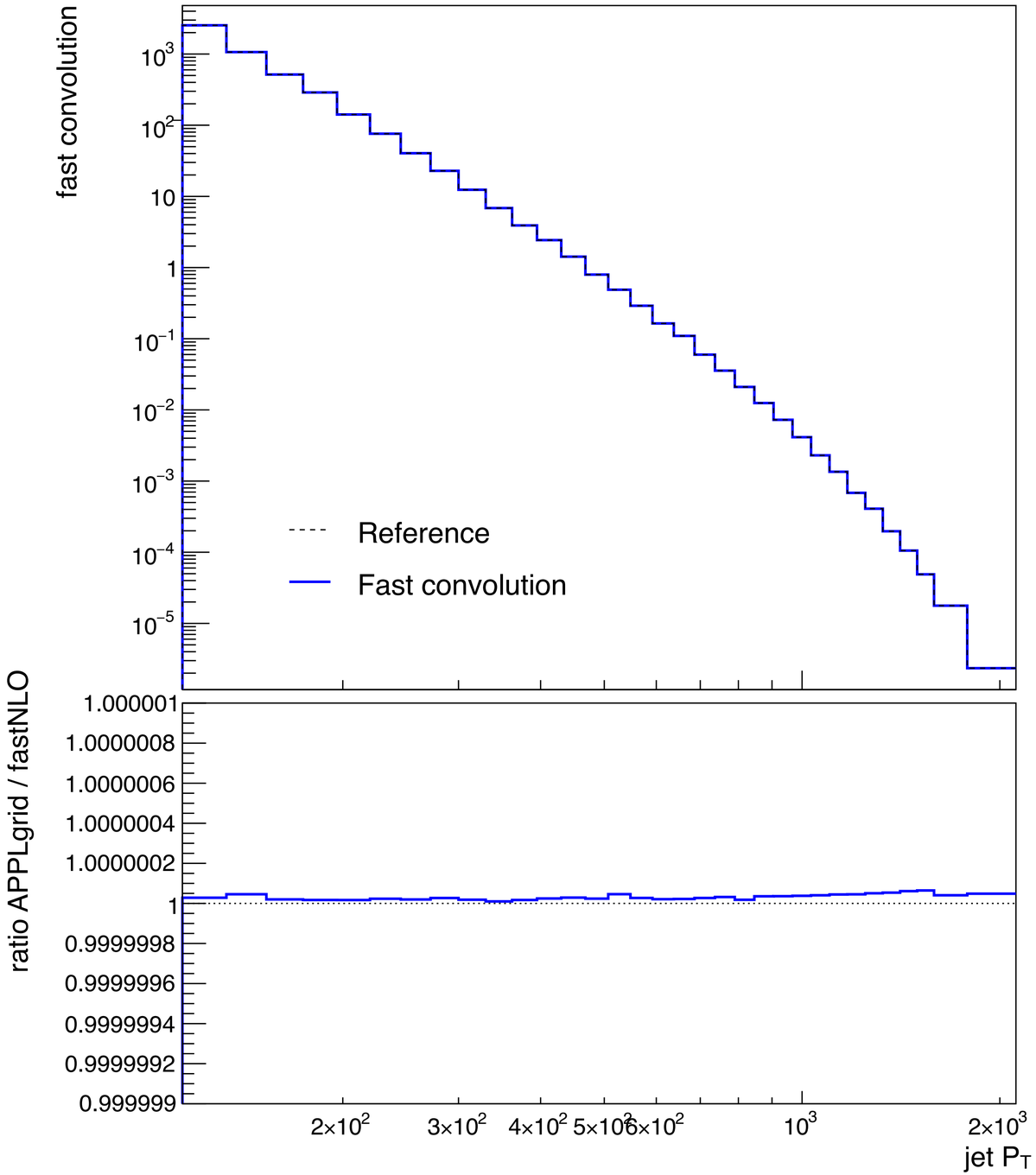


- Fast reproduction of the cross section with different PDFs

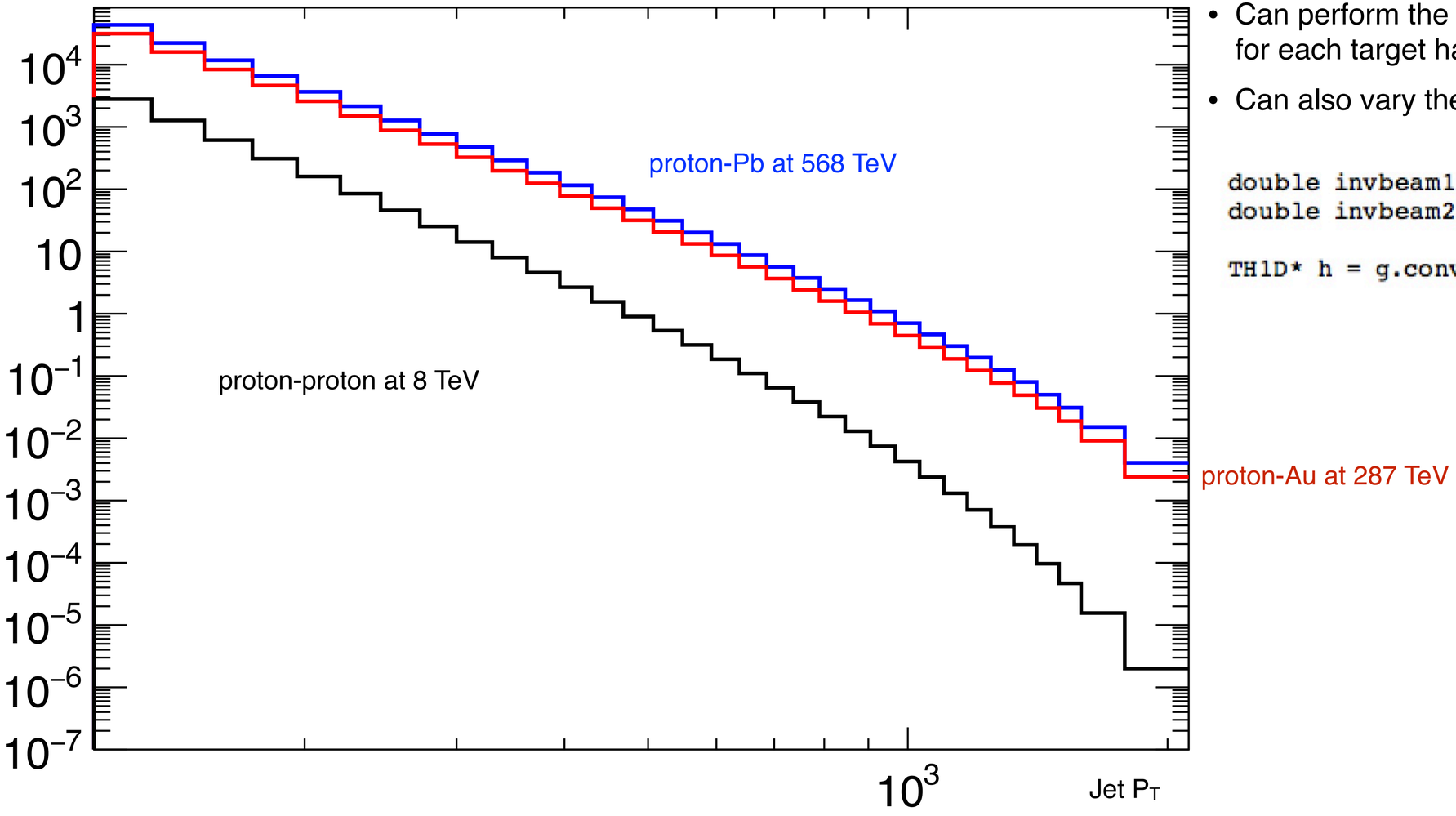
Grid remapping

- In principle can treat the weights on a grid at specific grid nodes exactly as if these were weights from the Vegas integration of an independent calculation
  - Can fill the weights onto a new grid with different parameters - fewer, or more, grid nodes, different transform for x, and Q2, different scale etc
  - Can be used to **reduce size of grids**, a posteriori determine the smallest grid size for any given required precision etc, also to rigorously combine bins, …
  - Caveat: remapping will accumulate interpolation errors - but note that interpolation errors are usually small - see later

- Implemented code to serialise fastNLO grids and remap to native APPLgrid format grids …
  - Significant saving in grid file size and memory footprint due to custom APPLgrid sparse structure etc



Use for Heavy Ion collisions …

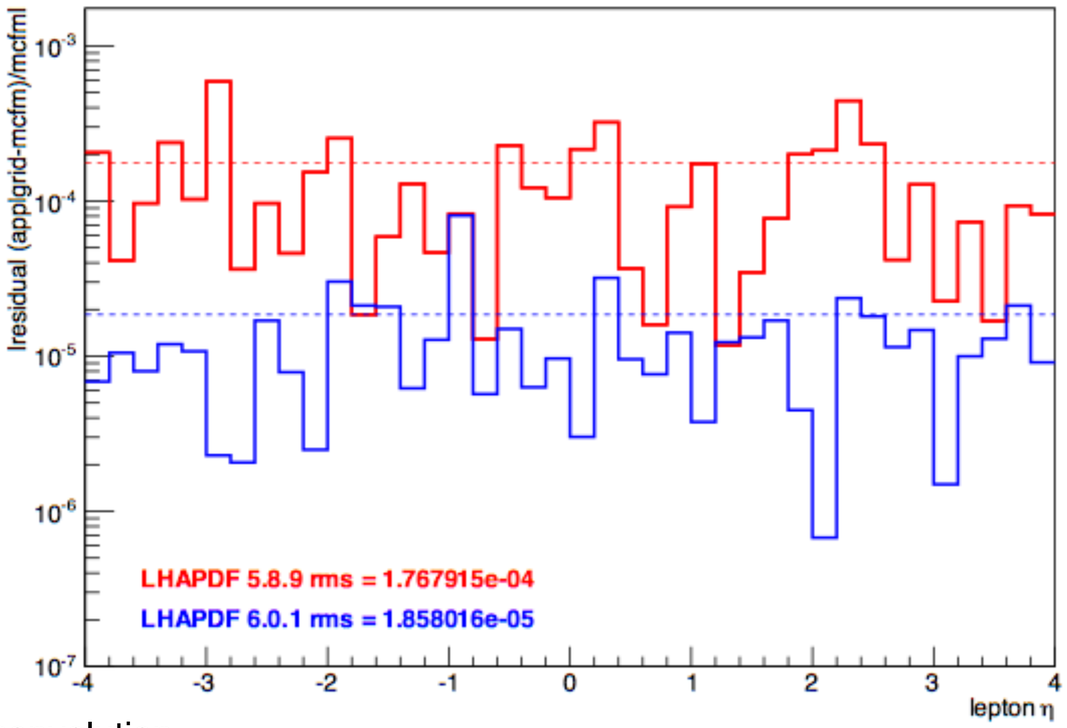


- Can perform the a posteriori convolution using a different PDF for each target hadron
- Can also vary the beam energy independently for each beam

```
double invbeam1 = 1;
double invbeam2 = 8/287;

TH1D* h = g.convolute( evolvepdf, evolvepdfi, alphaspdf,
                       nloop, rscale, fscale,
                       invbeam1, invbeam2 );
```

A brief diversion on LHAPDF version 6



- Reminder that LHAPDF version 6 has significantly better performance than LHAPDF 5.8.9
  - The a posteriori convolution is an **order of magnitude** more precise with LHAPDF version 6
- Currently there is no pure interface class for LHAPDF 6 format PDFs to be used in grid convolution
  - The grid convolution interface is still mostly generic lhaglue type interface …

```
extern "C" void evolvepdf(const double& x, const double& Q, double* xfx) {  e& Q, double* xf);
  xfx += 6;
  for (int i = -6; i < 7; i++) xfx[i] = pdf->xfxQ(i, x, Q);
  xfx[7] = pdf->xfxQ(22, x, Q);
}
```

  - New native LHAPDF 6 style interface would better integrate treatment of photon density and other LHAPDF 6 features
- In discussion with the LHAPDF developers, consensus that now might be the time to develop a generic PDF interface class
  - In the first instance should be based on LHAPDF::PDF but with all implementation details removed
  - Standardise treatment of photon density in fits and calculations ?
- Developers to work on this in conjunction with LHAPDF developers, or indeed on other aspects of LHAPDF functionality would be welcome - possibility for MCNET studentships

Towards the future …

- Have been discussing a standard interface for grid filling …
- Ultimate aim to have a single interface for all standard client interaction with grids
  - Filling, convolution etc …

```
/// production run  filling interface

/// fill the grid with a single weight for a single parton luminosity

virtual int fill( const double& x1, const double& x2, const double& Q2,
                  const int&    order,
                  const int&    process_index,
                  const double& observable,
                  double weight ) = 0;


/// fill the grid with a single weight for a vector of weights - one
/// entry for each parton luminosity

virtual int fill( const double& x1, const double& x2, const double& Q2,
                  const int&    order,
                  const double& observable,
                  const std::vector<double>& weights ) = 0;
```

- APPLfast combines several aspects of the grid filling for APPLgrid and fastNLO interface with NNLOJET, but still implement their respective native APIs
  - Hoping to work towards a common interface for fast interpolation grid interaction …
  - Both for the filling and for the subsequent a posteriori convolution
    - Client code could then be agnostic about which format of fast grid they were using

- These new interfaces could include native interface LHAPDF 6 type PDFs
  - Collect different package independent interfaces into a separate common utility package
  - Developers of other packages implement their code to operate with these standard interfaces, so that code elements could be easily exchanged
- Hopefully this should simplify code needed for interfacing with additional tools and any new code in the future

Outlook

- The APPLgrid project is now increasingly mature
  - Emphasis now mostly looking towards developments for use with NNLO cross section and interface with NNLOJET and fastNLO
  - However, desirable features are still being added
- In discussion with developers from other packages, hoping to move towards more standardised interfaces with other code
- We are always open to new suggestions or requests - particularly if people want to contribute to the project