LHAPDF 6: developments and plans

Andy Buckley
University of Glasgow

Parton distributions for the LHC, Benasque, 19 Feb 2015

Recap of LHAPDF 5 → 6

- **LHAPDF5 code had several fundamental, *nasty* problems**
  - Each PDF group's "wrapper" code declared different Fortran *un*common block working arrays: memory waste
  - Retrofitted multi-set mode multiplied this memory typically ×3
  - ☠ **∼ 2 GB memory requirements!**
  - ☠ **incompatible with Grid:** experiments' reweighting painful
  - ☠ **sometimes *incorrect* metadata values!!**
- **LHAPDF6 is a ground-up rewrite of the LHAPDF concept**
  - top-level design aims: fix memory problems, fix correctness problems, improve API, decouple set data from code releases, retain backward compatibility, minimise loads on developers/maintainers!
- I think all of these issues are now addressed – and releasing new sets is also far easier.

LHAPDF 6

Key features:

- Written in C++, with wrappers in Fortran and Python
- Memory now dynamically allocated: no static VMEM problems, no limit in concurrency
- PDF member rather than set is fundamental: increased flexibility
- Single (log-cubic in log space) interpolation routine for all sets. Q-subgrids in xf and α_s.
- Common grid data format and extensive metadata: self-documenting and not tied to code releases
- Arbitrary composite and constituent particle species, using PDG numbering scheme

Paper: http://arxiv.org/abs/1412.7420, now accepted for EPJ C

LHAPDF 6 performance

Memory:

- **2 GB → 275 kB!!!**
- Loading one PDF member ∼ 200 kB; whole set ∼ 10 MB. Memory scales with what you use.

Speed:

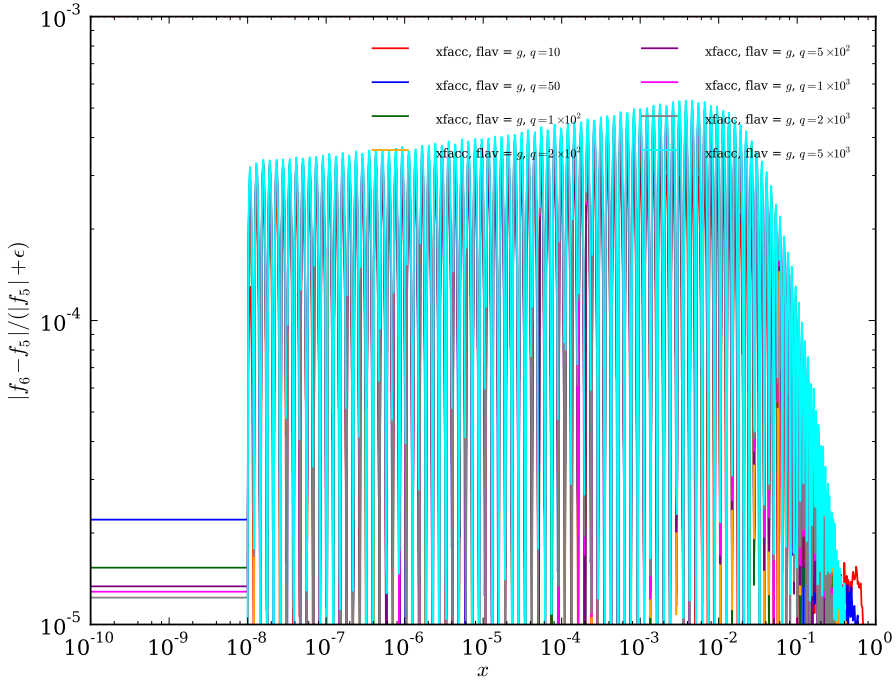
- Single member loading faster than v5. Set loading similar; faster with `zlibc` trick.
- Single-flavour interpolation is faster than LHAPDF5, e.g. in Sherpa (right)
- Possible further improvements through ipol weight caching (easy) and e.g. vectorization (hard).

Process/PDF	t_5	t_6	t_5/t_6
Cross-section integrations, 1M phase space points			
CT10			
$pp \to jj$	23′10″	9′17″	2.5
$pp \to \ell\ell$	4′12″	2′02″	2.1
$pp \to H$ (ggF)	0′20″	0′15″	1.3
NNPDF23nlo			
$pp \to jj$	54′40″	9′28″	5.8
$pp \to \ell\ell$	8′06″	2′33″	3.2
$pp \to H$ (ggF)	0′25″	0′11″	2.3
CKKW event generation, 100k $pp \leq$ 4 jet events			
CT10			
Weighted	43′02″	35′47″	1.2
Unweighted	5h04′39″	4h30′26″	1.1
NNPDF23nlo			
Weighted	47′47″	27′20″	1.7
Unweighted	6h44′47″	4h48′26″	1.4

Set migration and validation

- We set a nominal LHA5 $\rightarrow$ 6 reproduction accuracy target of per-mille (1/1000)
- Regularised deviation measure $\Delta = |f_6 - f_5|/(|f_5| + \epsilon)$

xf vs. x

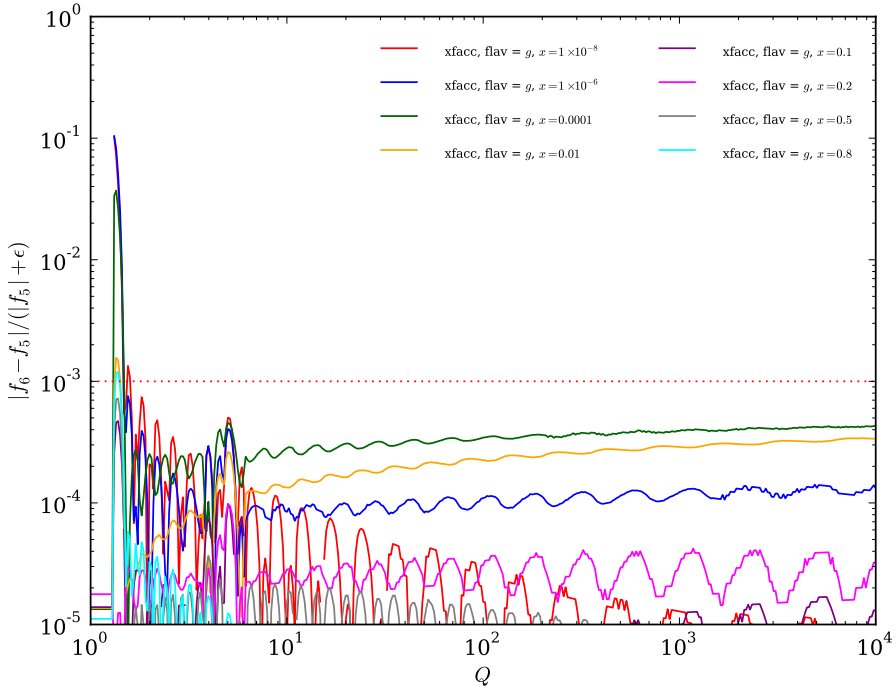


- Newest sets from each group migrated, plus requests and new submissions. Currently **368** validated sets.
- Get sets from **http://www.hepforge.org/archive/lhapdf/pdfsets/6.0/**

Set migration and validation

- We set a nominal LHA5 $\rightarrow$ 6 reproduction accuracy target of per-mille (1/1000)
- Regularised deviation measure $\Delta = |f_6 - f_5|/(|f_5| + \epsilon)$

xf vs. Q

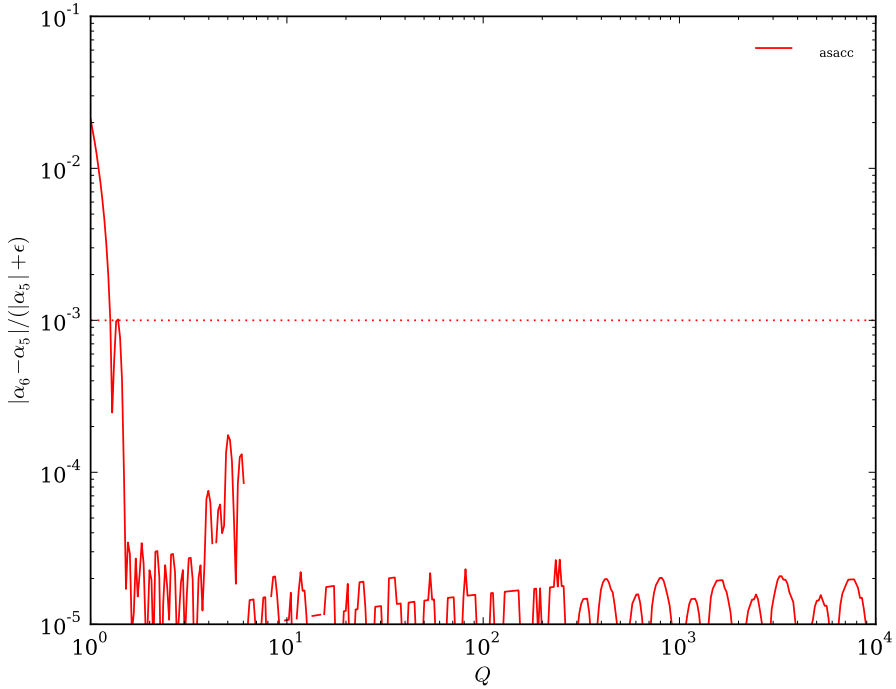


- Newest sets from each group migrated, plus requests and new submissions. Currently **368** validated sets.
- Get sets from **http://www.hepforge.org/archive/lhapdf/pdfsets/6.0/**

Set migration and validation

- We set a nominal LHA5 $\to$ 6 reproduction accuracy target of per-mille (1/1000)
- Regularised deviation measure $\Delta = |f_6 - f_5|/(|f_5| + \epsilon)$

α_s vs. Q



- Newest sets from each group migrated, plus requests and new submissions. Currently **368** validated sets.
- Get sets from **http://www.hepforge.org/archive/lhapdf/pdfsets/6.0/**

Examples: usage from C++

Single member:

```
#include "LHAPDF/LHAPDF.h"
...
LHAPDF::PDF* pdf = LHAPDF::mkPDF("CT10nlo/0");
size_t num_mems = pdf->numMembers();
// One value:
double xf_g = pdf->xfxQ(21, 1e-3, 126.0);
// Quark and gluon values:
vector<double> xfs;
pdf->xfxQ(1e-3, 126.0, xfs);
// All values (partons, photon, gluino, ...):
map<int, double> xfs = pdf->xfxQ(1e-3, 126.0);
delete pdf;
```

PDF set:

```
vector<unique_ptr<LHAPDF::PDF>> pdfs;
LHAPDF::mkPDFs("CT10nlo", pdfs);
for (const auto& p : pdfs)
  double xf_g = p->xfxQ(21, 1e-3, 126.0);
```

Examples: usage from Python

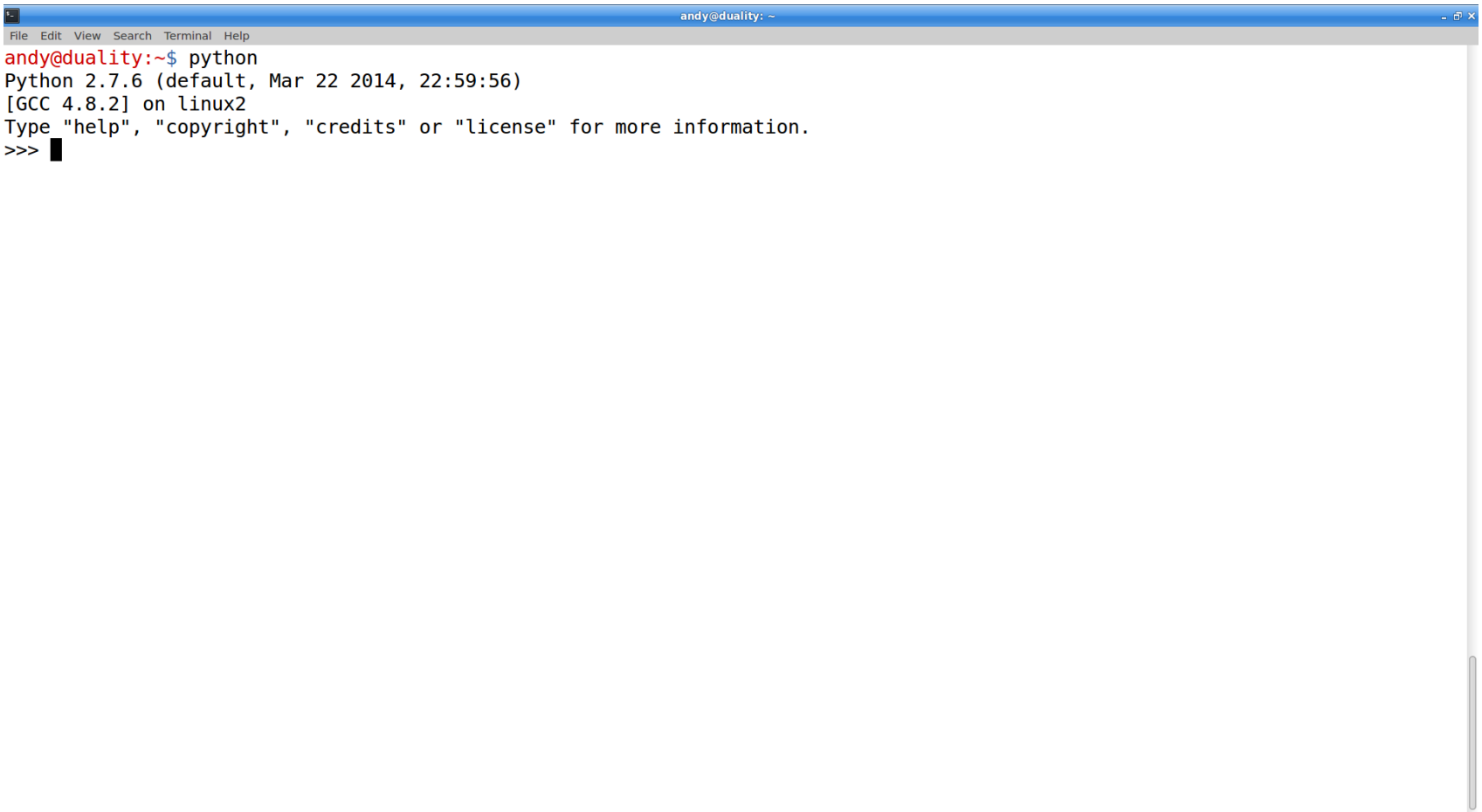
Single member:

```
>>> import lhapdf
>>> pdf0 = lhapdf.mkPDF("CT10nlo", 0)
>>> pdf0.xfxQ(21, 1e-3, 126)
31.199466144272378
```

PDF set:

```
>>> pdfs = lhapdf.mkPDFs("CT10nlo")
>>> len(pdfs)
52
>>> [pdf.xfxQ(21, 1e-3, 126) for pdf in pdfs]
[31.199466144272378, 31.10261967456719, ...
...
```

Examples: interactive usage in Python

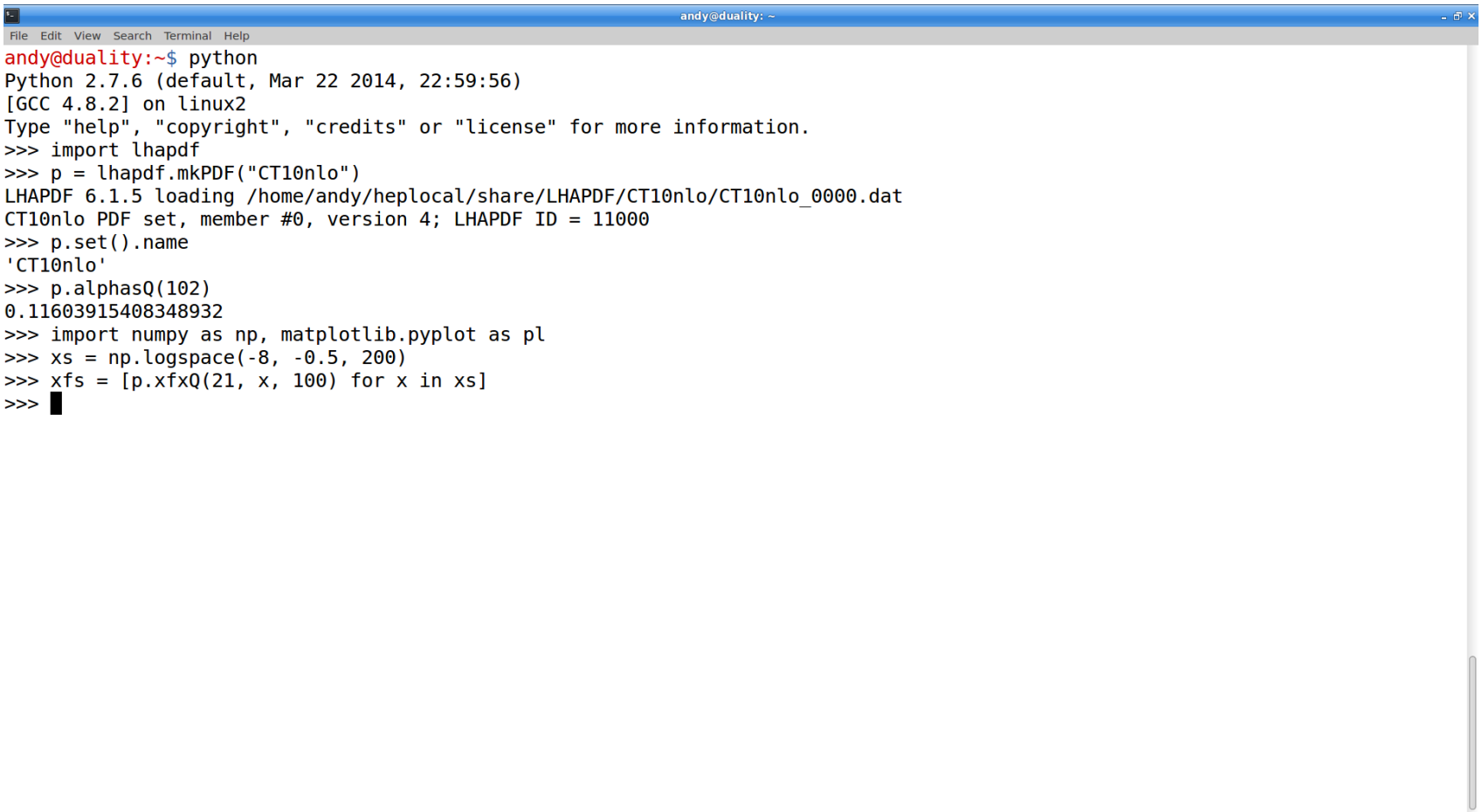


Examples: interactive usage in Python

Examples: interactive usage in Python

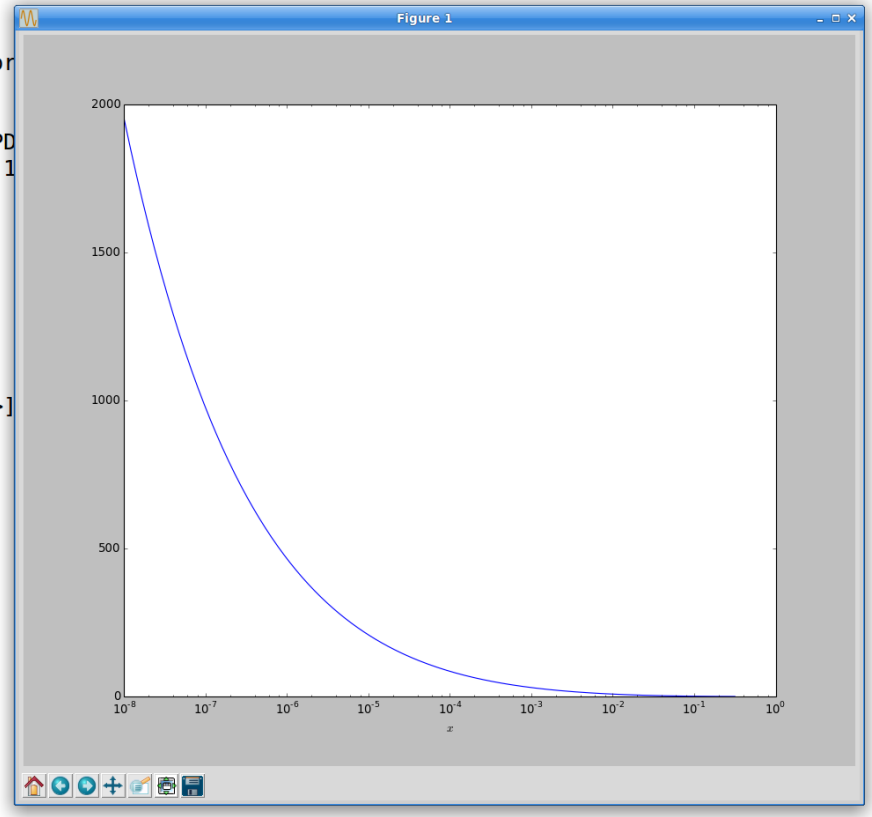
Examples: interactive usage in Python

Examples: interactive usage in Python



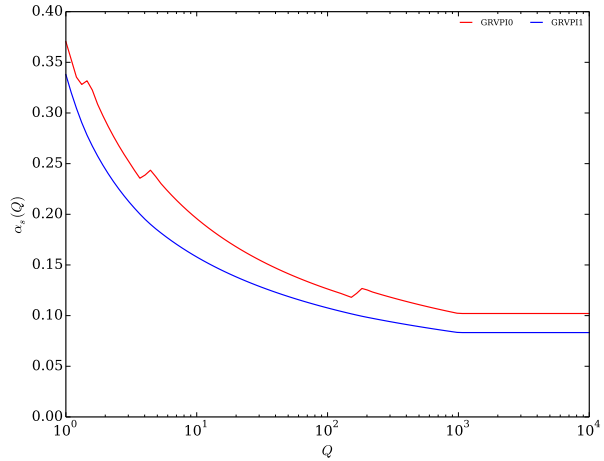
Examples: interactive usage in Python

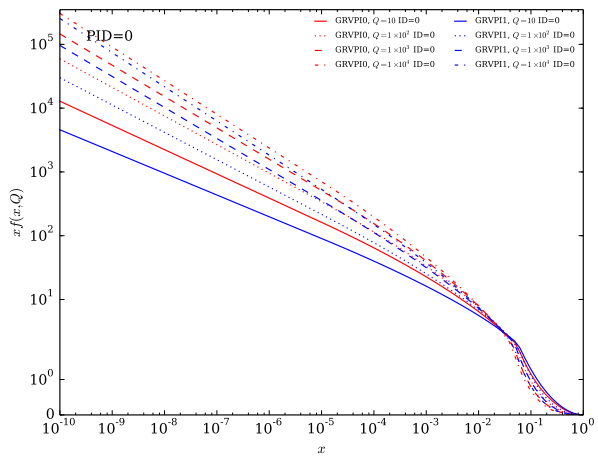
```
andy@duality: ~
File  Edit  View  Search  Terminal  Help
andy@duality:~$ python
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for
>>> import lhapdf
>>> p = lhapdf.mkPDF("CT10nlo")
LHAPDF 6.1.5 loading /home/andy/heplocal/share/LHAPD
CT10nlo PDF set, member #0, version 4; LHAPDF ID = 1
>>> p.set().name
'CT10nlo'
>>> p.alphasQ(102)
0.11603915408348932
>>> import numpy as np, matplotlib.pyplot as pl
>>> xs = np.logspace(-8, -0.5, 200)
>>> xfs = [p.xfxQ(21, x, 100) for x in xs]
>>> pl.plot(xs, xfs)
[<matplotlib.lines.Line2D object at 0x7f6c318baa50>]
>>> pl.xscale("log")
>>> pl.xlabel("$x$")
<matplotlib.text.Text object at 0x7f6c3ac26b90>
>>> pl.show()
```



Limitations, plans, etc.

- No photon PDFs, no non-interpolation PDFs (but interface allows)
- No pion PDFs yet – to be added soon (blocked only by some funny business: see α_s plot to right)
- No nuclear correction factors yet – a flexible implementation is half done
- No more nucleon PDFs will be migrated from v5, unless *really* well motivated
- Build requires Boost header library (SL6 version is ok, SL5 is too old). C++11 in future. . . when?





Limitations, plans, etc. (2)

- Interpolation caching optimisation will be done. Is there any incentive to work harder?!
- Some suggestions:
  - Put more "meta-PDF utilities" into tool collection, cf. reweighting and uncertainty functions.
  - Need a better Fortran interface? Better how?
  - Need better Q extrapolation for FCC etc.?
- Best to get input on what *you* – both fitters and users – want.
- Otherwise I hope that it just works and that not much maintenance is needed!