QCDNUM Overview

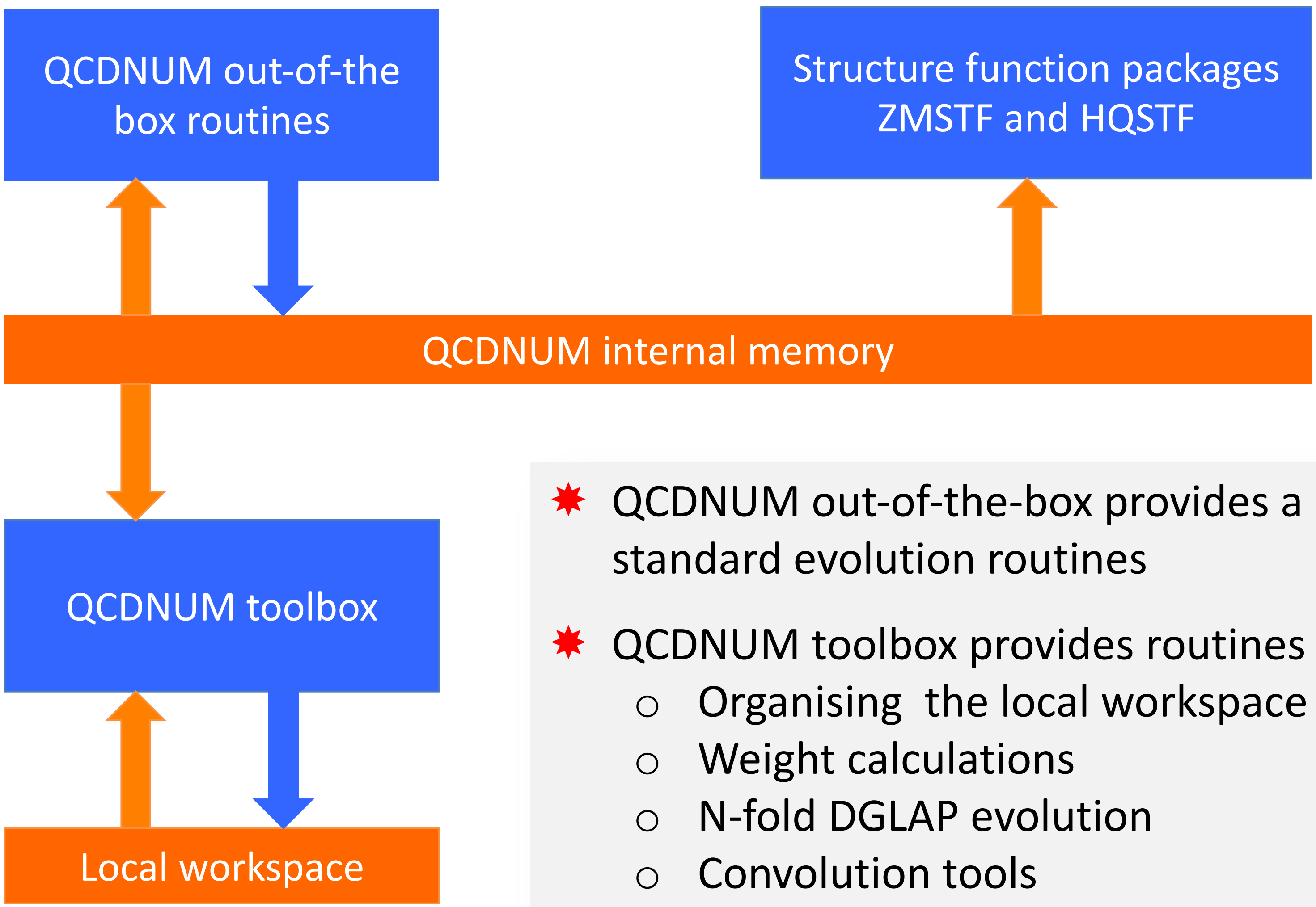
Michiel Botje

Nikhef, Amsterdam

HERAFitter users meeting

Heidelberg May 13, 2015

QCDNUM program structure



- QCDNUM out-of-the-box provides a set of standard evolution routines
- QCDNUM toolbox provides routines for
  - Organising the local workspace
  - Weight calculations
  - N-fold DGLAP evolution
  - Convolution tools

What out-of-the-box QCDNUM can do for you

- Evolution of α_s
- Unpolarised pdf evolution (NNLO)
- Polarised pdf evolution (NLO)
- Timelike evolution of fragmentation functions (NLO)
- Evolution in FFNS or VFNS (or mixed scheme)
- Variation of the renormalisation scale
- Many ways to access (linear combinations of) the pdfs
- Import external pdf sets into QCDNUM memory
- Structure functions (ZMSTF and HQSTF add-on packages) with variation of the factorisation scale

Out-of-the-box NNLO evolution

```
..
call QCINIT( lun, ' ' )
call GXMAKE( xmi, wt, 5, 100, nx, 3 )
call GQMAKE( qlims, wt, 2, 60, nq )
call FILLWT( itype, id1, id2, nw )
call SETORD( 3 )
call SETCBT( nffns, iqc, iqb, iqt )
call SETALF( alfas, ascale )
call EVOLFG( itype, func, def, iq0, eps )
call PDFTAB( itype, def, x, nx, q, nq, pdftab, 0 )
..
```

Initialisation and definition of evolution parameters

Evolution and generation of pdf tables

Only takes nine lines of code

New in version 17-01: datacards

- Here is a datacard file to run a QCDNUM evolution:

```
' GXMAKE   3  100  1  1.D-4               '
' GQMAKE        60  2  2.     1.D4        '
' FILLWT   ../weights/unpolarised.wgts '
' SETORD   3                              '
' SETALF   0.364  2.                      '
' SETCBT   0       3.  25.    1.D11       '
' EVOLVE   1       2.5                    '
' PDFOUT   ../pdfs/unpolarised.pdfs       '
```

Predefined keys

User defined keys

- Calls to QCDNUM routines can now be replaced by a datacard read

```
  ..
  call QCARDS( USUB, 'example.dcards', 0 )
  ..
```

User provided subroutine to process the user keys
When it sees a user keyword, QCARDS calls USUB

It is easy to add your own keys

- Step 1: Book your keys

```
call QCBOOK ( 'Add', 'MYKEY1' )
call QCBOOK ( 'Add', 'MYKEY2' )
```

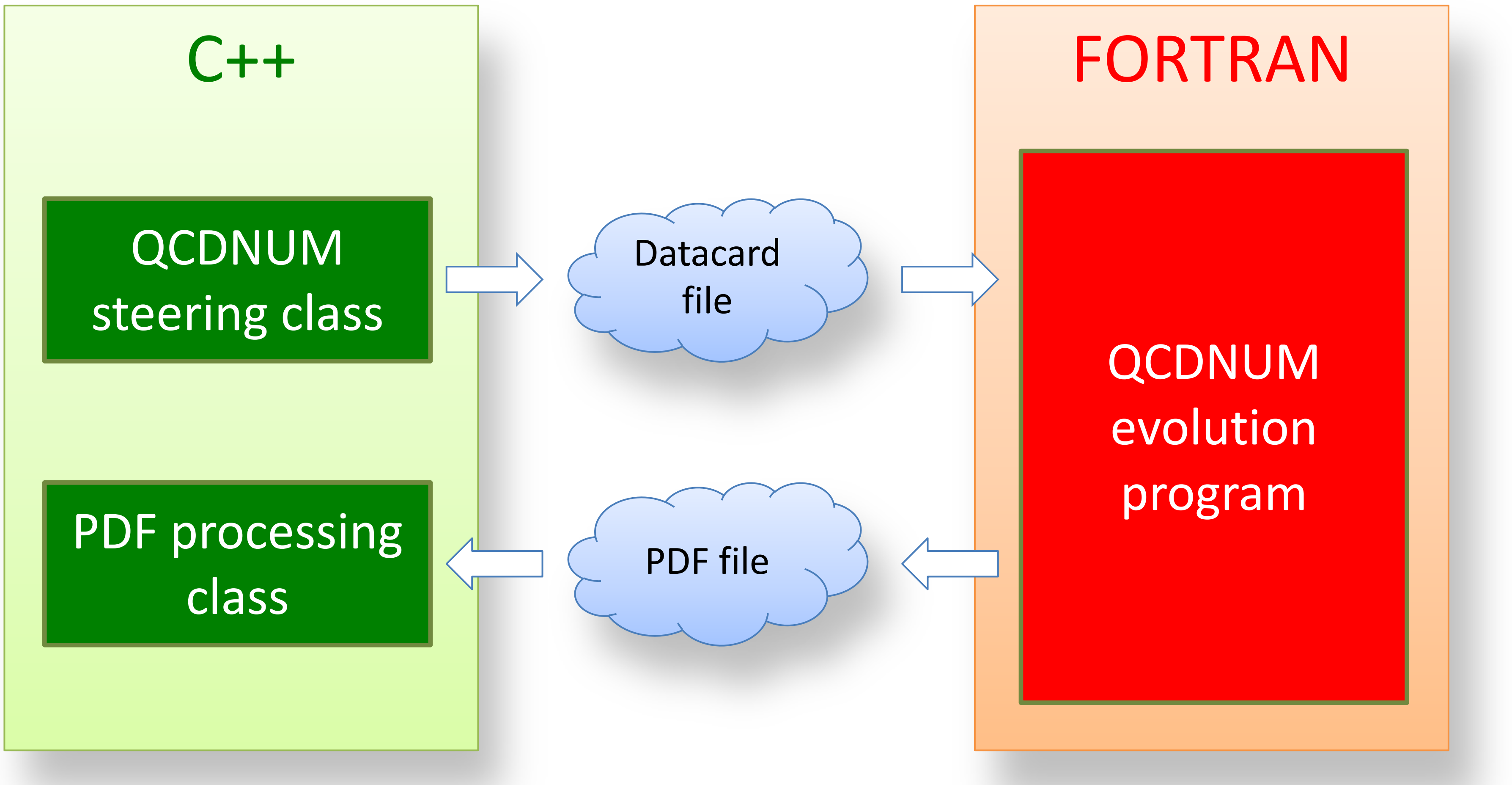
- Step 2: Write code to define key action (can be anything you like)

```
subroutine MYCARDS ( key, .., parameters, .. )
..
if     (key .eq. 'MYKEY1') then
  do something
elseif(key .eq. 'MYKEY2') then
  do something else
else
..
```

- Step 3: Off you go …

```
call QCARDS( MYCARDS, 'myrun.dcards', 0 )
```

Application: Mickey Mouse QCDNUM++



Remark: The QCDNUM distribution does not provide C++ classes

New in 17-01: Allow for intrinsic charm

- VFNS in QCDNUM 17-00
  - Starting scale must be below the charm threshold
  - All heavy quarks are generated dynamically through DGLAP

- VFNS in QCDNUM 17-01
  - Starting scale can be above the charm threshold
  - This allows for intrinsic charm (or bottom, top)
  - You cannot evolve backward over the threshold to n_f - 1

What the Toolbox can do for you

- The QCDNUM toolbox is a set of routines that operate on a local workspace (this is a large linear array that you declare in your application program)
- With these routines you can
  - Partition the workspace into (sets of) tables
  - Fill weight tables, combine weight tables
  - Simultaneous n-fold DGLAP evolution
  - Pdf interpolation, pdf transfer to QCDNUM internal memory
  - Convolution tools, fast convolution engine
- Supports generalised mass (GM) convolutions

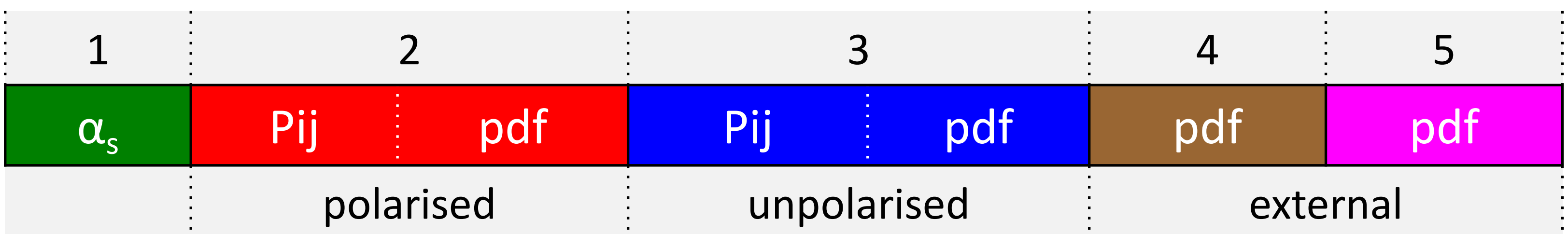
Partition the local workspace

- Can declare six types of tables

type-1	$\text{wgt}(x)$	type-4	$\text{wgt}(x, \mu^2, n_f)$
type-2	$\text{wgt}(x, n_f)$	type-5	$\text{pdf}(x, \mu^2)$
type-3	$\text{wgt}(x, \mu^2)$	type-6	$\text{alfa}(\mu^2)$

New in QCDNUM 17-01

- It is useful (and easy) to organise tables into sets, e.g.:



- With such a layout, a single routine can evolve both polarised and unpolarised pdfs just by changing the table set identifier

Convolution kernels in QCDNUM

$$f \otimes C \equiv x \int_{\chi}^{1} \frac{\mathrm{d}z}{z} f(z,\mu^2)\, C\left(\frac{\chi}{z}, \mu^2, Q^2, m_{\mathrm{h}}^2\right)$$

- Uses rescaling variable like $\chi \equiv ax = \left(1 + \frac{\mu_h^2}{Q^2}\right) x$
- For splitting functions $\chi = x, a = 1$
- C(…) and a(…) must be supplied as Fortran functions
- You can generate weight tables at initialisation and then enjoy very fast convolution as weighted sum
- Can also combine weight tables e.g.: $W_3 = W_1 \otimes W_2$

- Weight routines can handle singularities

$$C = A + [B]_+ + R[S]_+ + D\delta(1-x)$$

- Each term has its own makewt routine

Local workspace

Convolution kernel

```
call MAKEWTA(w,id,afun,achi)
call MAKEWTB(w,id,bfun,achi,0)
call MAKEWRS(w,id,rfun,sfun,achi,0)
call MAKEWTD(w,id,dfun,achi)
```

Table identifier

Defines rescaling variable

New in V17-01: n-fold DGLAP evolution

$$\frac{\partial f_i(x,\mu^2)}{\partial \ln \mu^2} = \sum_{j=1}^{n} [P_{ij} \otimes f_j](x,\mu^2)$$

- Suite of routines for coupled n×n evolution
  - Fill tables of expansion coefficients
  - Coupled DGLAP evolution
  - Pdf interpolation
  - Export pdfs (gluon and quarks only) to internal memory
- The splitting function weight tables must be filled beforehand with the makewt routines

Fast convolution engine

- Can calculate all kind of convolutions, such as

$$f \otimes K, \quad f \otimes K_a \otimes K_b, \quad f_a \otimes f_b, \quad f_a \otimes f_b \otimes K$$

- The idea behind the fast convolution engine:

  ① First, you pass a list of interpolation points

  ② The engine then sets-up an interpolation mesh and restricts the subsequent calculations to only these mesh points

  ③ Next, copy pdfs from memory or local workspace into the engine

  ④ Do your convolutions or other calculations (e.g. multiply by α_s^n, etc.)

  ⑤ Finally, let the engine return the list of interpolated results

- This usually leads to very fast code

- Note that the structure function packages ZMSTF and HQSTF are entirely based on the fast convolution engine

Is QCDNUM fast?

- Mimic a fit by 1000 evolutions and 10^6 F_2 and F_L calculations
  - VFNS NNLO on a 5-fold 100×60 x-Q^2 grid in the HERA kinematic range
  - MacBook Pro (2012) with 2.5 GHz Intel core i5 and 8 GB RAM
  - Code compiled with O2 optimisation and w/o array boundary check

10^3 evolutions	2×10^6 structure functions	total
3.7 secs	4.0 secs	7.7 secs

- Heavy quark contributions to $F_{2,L}$ with HQSTF take about the same CPU as the light quark structure functions with ZMSTF

Yes, QCDNUM is pretty fast

New in V17-01: toolbox tutorial

- Step-by-step building of your own evolution code (in LO)

  E.1 How to partition a workspace

  E.2 How to calculate weight tables

  E.3 How to fill the α_s table

  E.4 Singlet/gluon and non-singlet evolution

  E.5 How to construct the singlet/non-singlet basis pdfs

  E.6 Your own interpolation routine

  E.7 How to compute a structure function

  E.8 Make it robust and user-friendly

- The last two sections are still missing
- Fortran code in the testjobs directory and also on the web

The QCDNUM Home Page

QCDNUM-17-00/06

Download

How to install

Write-up

Talks

Example jobs

Tutorial

Release history

Next release

Contact

Stable Release	qcdnum-17-00/06	Write-up
Beta Release	qcdnum-17-01/0f	Write-up

- QCDNUM 17-01 is still in its beta release
- The beta release is almost fully functional so there is no problem to use it
- Please visit the QCDNUM website to check for updates and the stable release 17-01
- Or ask me to be put on the mailing list

`https://www.nikhef.nl/~h24/qcdnum`