QCDNUM 17-01-13

Michiel Botje
Nikhef, Amsterdam

xFitter developers meeting

January 11, 2017

Fast interpolation in QCDNUM-17-01-13

- The interpolation algorithm is split in three parts:

  1. Set-up an interpolation mesh and store weights for each interpolation point
  2. Store function value for all the mesh points (take mesh overlaps into account)
  3. Interpolate by computing a weighted sum of function values over each mesh

- Efficient when more than one function is interpolated for a given point or list of points (set-up the interpolation only once for all functions)
- Efficient when meshes of a list of points do overlap (no redundant function calls in the overlaps)
- Gain speed by very fast addressing of pdf tables
- Steps 1.2.3 above are packed inside each of the present routines
- A future version of the toolbox will make steps 1.2.3 separately available for fast interpolation of any user-defined function

NB: an interpolation mesh is an $n_x \times n_q$ subgrid around the interpolation point e.g. 3×3 for quadratic interpolation

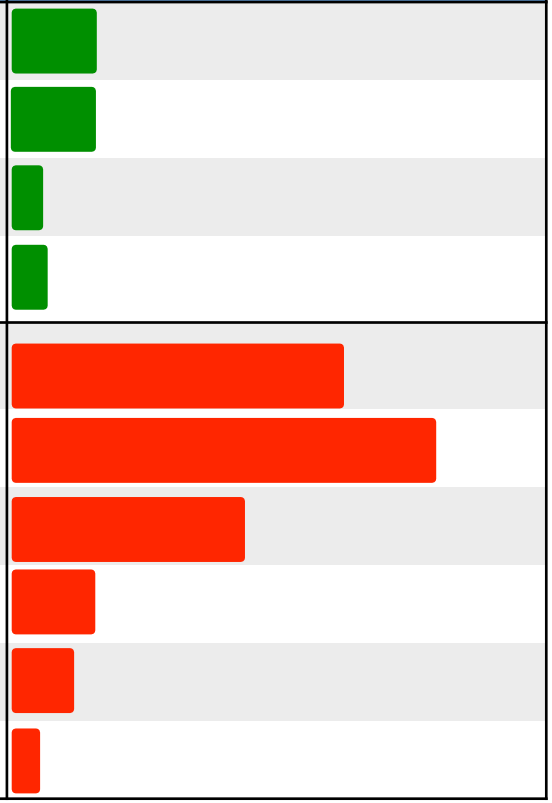
The new interpolation routines

Old Name	New Name
FSNSXQ	BVALXQ(iset, id, x, q, ichk)
FVALXQ	FVALXQ(iset, id, x, q, ichk)
FPDFXQ	ALLFXQ(iset, x, q, f, n, ichk)
FSUMXQ	SUMFXQ(iset, c, isel, x, q, ichk)
PDFLST	FFLIST(iset, c, isel, x, q, f, n, ichk)
PDFTAB	FTABLE(iset, c, isel, x, nx, q, nq, tbl, ichk)

- ☑ Most routines are (much) faster than before
- ☑ All routines can access extra pdfs (if present in the pdf set)
- ☑ Singlet-nonsinglet selection via the `isel` flag (see later)
- ☑ Speed-up by optionally switching-off checks (see later)

Calculate weighted sum $\sum_{i}^{n_f} e_i^2 x(q_i + \bar{q}_i)$ for 2400 x-Q² points

PDF accessed in one call	Subroutine	Speed gain wrt 01-12	CPU (arbitrary units)
Single basis pdf	BVALIJ	6	
Single flavour pdf	FVALIJ	9	
All flavour pdfs	ALLFIJ	6	
Linear combination	SUMFIJ	11	
Single basis pdf	BVALXQ	2	
Single flavour pdf	FVALXQ	4	
All flavour pdfs	ALLFXQ	2	
Linear combination	SUMFXQ	19	
List	FFLIST	1	
Table (here 60×40)	FTABLE	2	



NB: Timing profile depends much on what you calculate

Singlet-nonsinglet selection (1)

```
SUMFXQ( iset, c, isel, x, q, ichk )
FFLIST( iset, c, isel, x, q, f, n, ichk )
FTABLE( iset, c, isel, x, nx, q, nq, tbl, ichk )
```

- `c(-6:6)` array with quark and antiquark coefficients
- `isel` selection flag that defines what is returned

`isel` =	`0`	Gluon
	`1`	Quark linear combination defined by `c`
	`2-8`	Singlet-nonsinglet selection (next slides)
	`12+i`	Extra pdf xf_i if present in `iset`

Singlet-nonsinglet selection (2)

Reminder QCDNUM basis pdfs $q_i^{\pm} \equiv q_i \pm \bar{q}_i$

$\|e_1^+\rangle$	$=$	$\sum_{i=1}^{n_f} (q_i + \bar{q}_i)$	singlet
$\|e_1^-\rangle$	$=$	$\sum_{i=1}^{n_f} (q_i - \bar{q}_i)$	valence
$\|e_2^{\pm}\rangle$	$=$	$u^{\pm} - d^{\pm}$	$n_f \geq 3$
$\|e_3^{\pm}\rangle$	$=$	$u^{\pm} + d^{\pm} - 2s^{\pm}$	$n_f \geq 3$
$\|e_4^{\pm}\rangle$	$=$	$u^{\pm} + d^{\pm} + s^{\pm} - 3c^{\pm}$	$n_f \geq 4$
$\|e_5^{\pm}\rangle$	$=$	$u^{\pm} + d^{\pm} + s^{\pm} + c^{\pm} - 4b^{\pm}$	$n_f \geq 5$
$\|e_6^{\pm}\rangle$	$=$	$u^{\pm} + d^{\pm} + s^{\pm} + c^{\pm} + b^{\pm} - 5t^{\pm}$	$n_f \geq 6$

nonsinglet

Singlet-nonsinglet selection (3)

- The linear combination `c(-6:6)` in terms of basis pdfs:

$$|xq\rangle = \overbrace{d_1^+|xe_1^+\rangle}^{\text{si}} + \overbrace{\sum_{i=2}^{n_\text{f}} d_i^+|xe_i^+\rangle}^{\text{ns}^+} + \overbrace{d_1^-|xe_1^-\rangle}^{\text{va}} + \overbrace{\sum_{i=2}^{n_\text{f}} d_i^-|xe_i^-\rangle}^{\text{ns}^-}$$

- **`isel`:** Select combination of singlet and nonsinglet terms

$$|1\rangle = \text{all terms} \quad |2\rangle = \text{si} \quad |3\rangle = \text{all ns} \quad |4\rangle = \text{ns}^+$$
$$|5\rangle = \text{va} + \text{ns}^- \quad |6\rangle = \text{ns}^- \quad |7\rangle = \text{va} \quad |8\rangle = d_1^+|xg\rangle$$

- Useful for structure function calculations e.g.:

$$\begin{aligned} F_2^\text{LO} &= |1\rangle \\ F_2^\text{NLO} &= C_\text{g} \otimes |8\rangle + C_\text{s} \otimes |2\rangle + C_\text{ns} \otimes |3\rangle \\ F_2^\text{NNLO} &= C_\text{g} \otimes |8\rangle + C_\text{s} \otimes |2\rangle + C_+ \otimes |4\rangle + C_- \otimes |5\rangle \end{aligned}$$

Speed-up goody: switch-off checks

```
BVALXQ|IJ( iset, id, x|ix, q|iq, ichk )
FVALXQ|IJ( iset, id, x|ix, q|iq, ichk )
ALLFXQ|IJ( iset, x|ix, q|iq, f, n, ichk )
SUMFXQ|IJ( iset, c, isel, x|ix, q|iq, ichk )
```

- The `ichk` flag controls the error checking:

0	Return a `null` when `x` or `qmu2` are outside the grid
+1	Fatal error when `x` or `qmu2` are outside the grid
NEW −1	As above, but do not check the input `iset` and `id`

- Useful to speed-up loops (most effective for `BVALIJ`)

```
pdf(1)   = BVALIJ( iset, id, ix(1), iq(1), +1 )
do k = 2,n
  pdf(k) = BVALIJ( iset, id, ix(k), iq(k), -1 )
enddo
```

Restricted access to pdfs (1)

- QCDNUM can have pdf sets with different evolution parameters in memory

`iset = 0`	Current (active) parameters that steer QCDNUM
`= 1-24`	Parameters stored with a pdf set

- You can only access pdf sets evolved with the current parameters
- So you may have to first activate the parameters of your favourite pdf set
- Looks clumsy but protects against using incompatible pdfs in a calculation
- There are four routines to manage sets of evolution parameters

`USEPAR( iset )`	Activate parameter set (make them current parameters)
`KEYPAR( iset )`	Get key of parameter set (equal key = equal parameters)
`PUSHCP`	Push current parameters on a stack
`PULLCP`	Pull current parameters from a stack

Restricted access to pdfs (2)

- The routines `USEPAR` and `PULLCP` are slow because they re-initialise QCDUM
- Always calling `USEPAR` is therefore not a very good idea …

```
call USEPAR(iset)
call ALLFXQ(iset, ... )
```

- Use keys to first check if a pdf set is evolved with the current parameters
  - Parameter keys are (un)equal when parameter sets are (un)equal

```
if( KEYPAR(iset) .eq. KEYPAR(0) ) then
  call  ALLFXQ(iset, ...)  !get pdfs directly
else
  call  PUSHCP             !store current parameters
  call  USEPAR(iset)       !activate parameters of iset
  call  ALLFXQ(iset, ...)  !get pdfs
  call  PULLCP             !restore current parameters
endif
```

- If your pdf sets have the same parameters you don't need all this of course!

Progress on the C++ interface

- Valerio sent me yesterday a 17-01-13 version with a C++ interface and test-jobs that worked out-of-the-box
- Presently I am setting-up the Autotool part of the release script to handle the C++ code correctly
- For the moment there exists an interface to QCDNUM out-of-the-box routines, and the add-ons ZMSTF and HQSTF
- Waiting for Renats QCD-QED stand-alone code to turn it into an add-on package, also with a C++ interface
- Interface to the QCDNUM toolbox will come later
- Will soon start testing all existing C++ routines one-by-one

QCDNUM joblist beyond 17-01-13

- C++ interface coming soon (my thanks to Valerio Bertone)
- Cleanup code to have one evolution routine (now there are two)
- VFNS evolution starting above charm threshold (intrinsic charm)
- Re-enable cuts
- Toolbox improvements (make it more user-function driven)
- Upgrade polarised and time-like evolution to NNLO
- …

You are welcome to add to this list or make suggestions to prioritise it