QCDNUM17

Fast QCD evolution and convolution

Michiel Botje
NIKHEF
PO Box 41882 1009DB Amsterdam

PDF4LHC workshop, CERN, August 6-7, 2009

What is QCDNUM ?

- Fortran program that evolves a_s and the parton densities up to NNLO on a grid in x and Q^2
- Possibility to vary renormalisation scale with respect to the factorisation scale
- Convolution of pdfs with a Wilson coefficient in zero mass or in any generalised mass scheme
- Fast, accurate and user friendly

QCDNUM has a long history...

1988	Code by Ouraou and Virchaux (BCDMS)	CRAY vectorized Fortran
1993	NMC adaptation to low x	CRAY vectorized Fortran
1998	QCDNUM16.12 used by ZEUS	Unix Fortran77
2007	NNLO upgrade QCDNUM17	Unix Fortran77
2009	QCDNUM17-beta-05	Unix Fortran77

DGLAP evolution of PDFs

Singlet/gluon evolution

$$q_{\rm s} = \sum_{i=1}^{n_f} (q_i + \bar{q}_i)$$

$$\frac{\partial}{\partial \ln \mu^2} \begin{pmatrix} q_{\rm s} \\ g \end{pmatrix} = \begin{pmatrix} P_{\rm qq} & P_{\rm qg} \\ P_{\rm gq} & P_{\rm gg} \end{pmatrix} \otimes \begin{pmatrix} q_{\rm s} \\ g \end{pmatrix}$$

Non-singlet evolution

$$q_{ij}^{\pm} = (q_i \pm \bar{q}_i) - (q_j \pm \bar{q}_j)$$

$$q_{\rm v} = \sum_{i=1}^{n_f} (q_i - \bar{q}_i)$$

$$\frac{\partial q_{\rm ns}}{\partial \ln \mu^2} = P_{\rm ns} \otimes q_{\rm ns}$$

	LO	NLO	NNLO
q_{ij}^{+}	P_{qq}	P_{+}	P_{+}
q_{ij}^{-}	P_{qq}	P_{-}	P_{-}
$q_{\rm v}$	P_{qq}	P_{-}	$P_{\rm v}$

QCDNUM uses internally a standard singlet/non-singlet set of basis functions

Singlet or Valence

$$\begin{pmatrix} e_1^{\pm} \\ e_2^{\pm} \\ e_3^{\pm} \\ e_4^{\pm} \\ e_5^{\pm} \\ e_6^{\pm} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & & & & \\ 1 & 1 & -2 & & & \\ 1 & 1 & 1 & -3 & & \\ 1 & 1 & 1 & 1 & -4 & \\ 1 & 1 & 1 & 1 & 1 & -5 \end{pmatrix} \begin{pmatrix} u^{\pm} \\ d^{\pm} \\ s^{\pm} \\ c^{\pm} \\ b^{\pm} \\ t^{\pm} \end{pmatrix}$$

Non-singlet

$$q_i^{\pm} = q_i \pm \bar{q}_i$$

Evolution schemes in QCDNUM

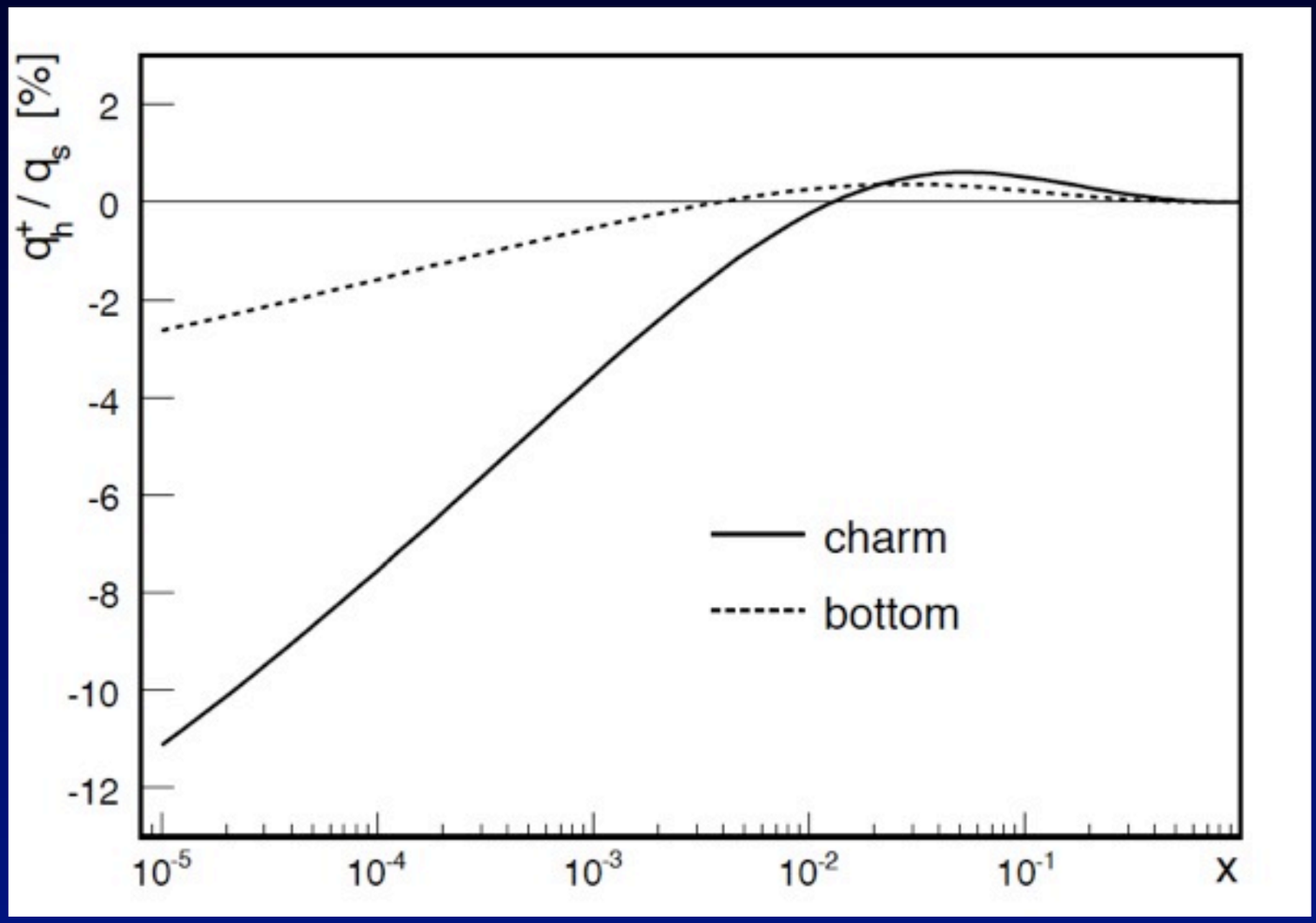
- FFNS: number of active flavours is kept constant $3 < n_f < 6$ for all Q^2
- VFNS: number of flavours changes from n_f to $n_f + 1$ at the thresholds $Q^2_{c,b,t}$

- Input: gluon + 6 light quark PDFs at $Q^2_0 < Q^2_c$
- Heavy quark PDFs are generated by the DGLAP evolution equations
- The PDFs and a_s are continuous at the thresholds in LO and NLO but are discontinuous in NNLO

K.G. Chetyrkin et al., PRL 79 (1997) 2184
M. Buza et al., EPJ C1 (1988) 301

Example: NNLO discontinuity of charm and bottom at their thresholds

- Bottom ~3% of singlet at low x
- Charm ~10%
- PDFs negative below $x = 10^{-2}$
- No problem since PDFs are not observables



Renormalisation scale dependence

- The strong coupling constant evolves on the renormalisation scale and the PDFs evolve on the factorisation scale
- QCDNUM supports a linear relationship between these two scales

$$\mu_{\mathrm{R}}^2 = a\,\mu_{\mathrm{F}}^2 + b$$

- Allows to study renormalisation scale dependence of PDFs (and STFs, Xsecs)

Numerical method in a nutshell

- Solve DGLAP numerically on a log x-Q^2 grid
- Based on linear and quadratic polynomial spline interpolation on multiple equidistant grids
- Convolution integrals become weighted sums with weights calculated at program initialization
- Evolution step becomes a lower triangular n×n matrix equation solved by forward substitution
- This matrix roll-up is the only $O(n^2)$ calculation in the whole program, everything else is $O(n)$

☑ QCDNUM is very fast

QCDNUM: compact user interface

```
call QCINIT(6,' ')
call GXMAKE(xmin,1,1,nxin,nx,iosp)
call GQMAKE(qq,wt,2,nqin,nq)
call FILLWT(0,id1,id2,nwords)
call SETORD(3)
call SETALF(as0,r20)
call SETCBT(0,iqc,iqb,iqt)
call EVOLFF(func,def,iq0,eps)
call PDFSXQ(x,q,pdf,0)
```

Full NNLO evolution in 9 lines

QCDNUM: grids and weights

```
call QCINIT(6,' ')
call GXMAKE(xmin,1,1,nxin,nx,iosp)
call GQMAKE(qq,wt,2,nqin,nq)
call FILLWT(0,id1,id2,nwords)
call SETORD(3)
call SETALF(as0,r20)
call SETCBT(0,iqc,iqb,iqt)
call EVOLFF(func,def,iq0,eps)
call PDFSXQ(x,q,pdf,0)
```

* Initialise QCDNUM
* Define x-Q^2 grid and lin/quad interpolation
* Calculate weight tables

QCDNUM: evolution parameters

```
call QCINIT(6,' ')
call GXMAKE(xmin,1,1,nxin,nx,iosp)
call GQMAKE(qq,wt,2,nqin,nq)
call FILLWT(0,id1,id2,nwords)
call SETORD(3)
call SETALF(as0,r20)
call SETCBT(0,iqc,iqb,iqt)
call EVOLFF(func,def,iq0,eps)
call PDFSXQ(x,q,pdf,0)
```

* Set LO, NLO, NNLO
* Input strong coupling constant
* FFNS, VFNS and thresholds $Q^2_{c,b,t}$

QCDNUM: evolution of all PDFs

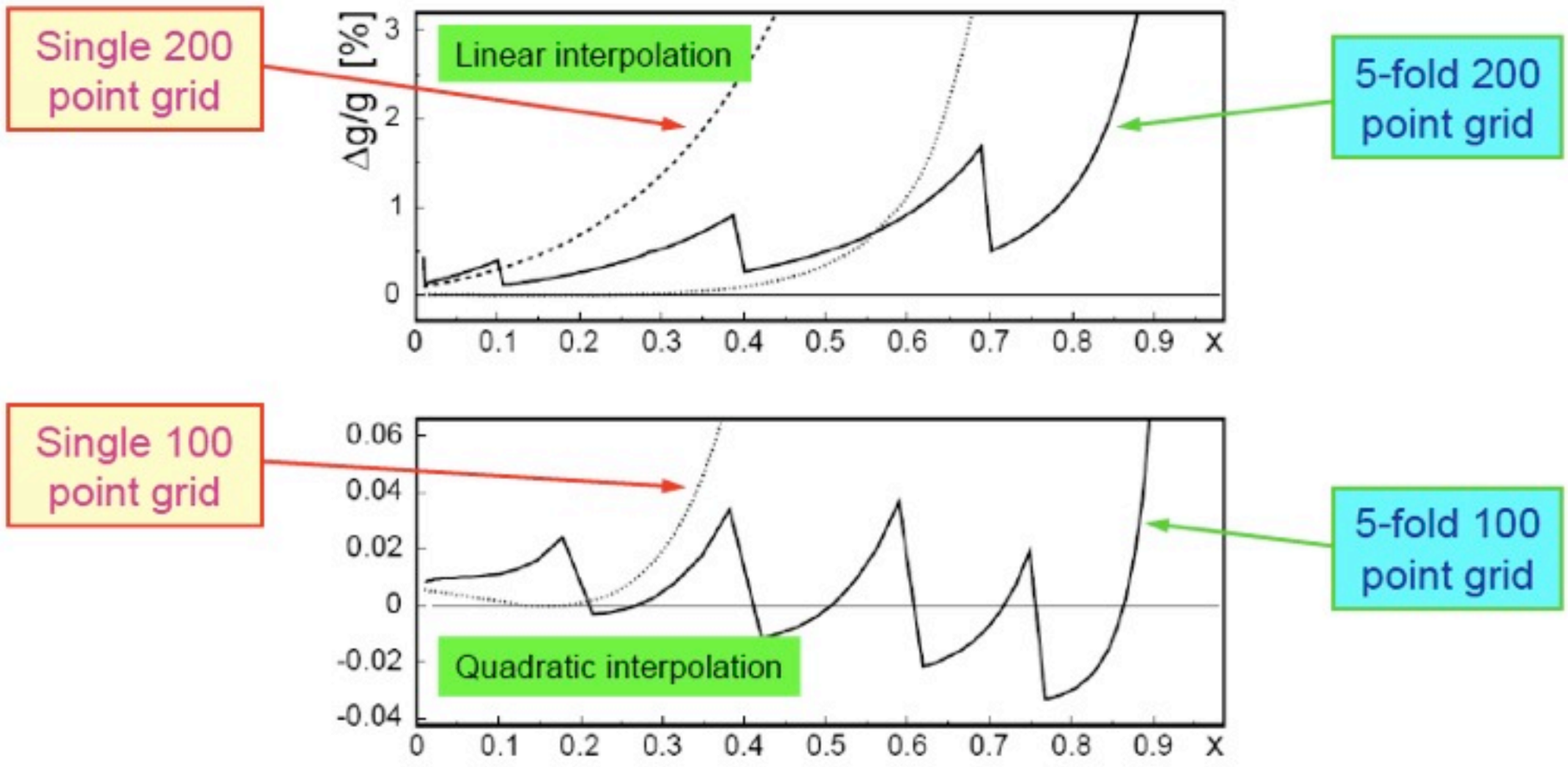
```
call QCINIT(6,' ')
call GXMAKE(xmin,1,1,nxin,nx,iosp)
call GQMAKE(qq,wt,2,nqin,nq)
call FILLWT(0,id1,id2,nwords)
call SETORD(3)
call SETALF(as0,r20)
call SETCBT(0,iqc,iqb,iqt)
call EVOLFF(func,def,iq0,eps)
call PDFSXQ(x,q,pdf,0)
```

* Function **func** provides input PDFs at Q^2_0
* Array **def** describes flavour composition
* Several routines to return PDFs at x, Q^2

How do we know that QCDNUM is correct?

QCDNUM-Pegasus comparison

- NLO gluon evolution from $\mu^2 = 2$ to 10^4 GeV^2

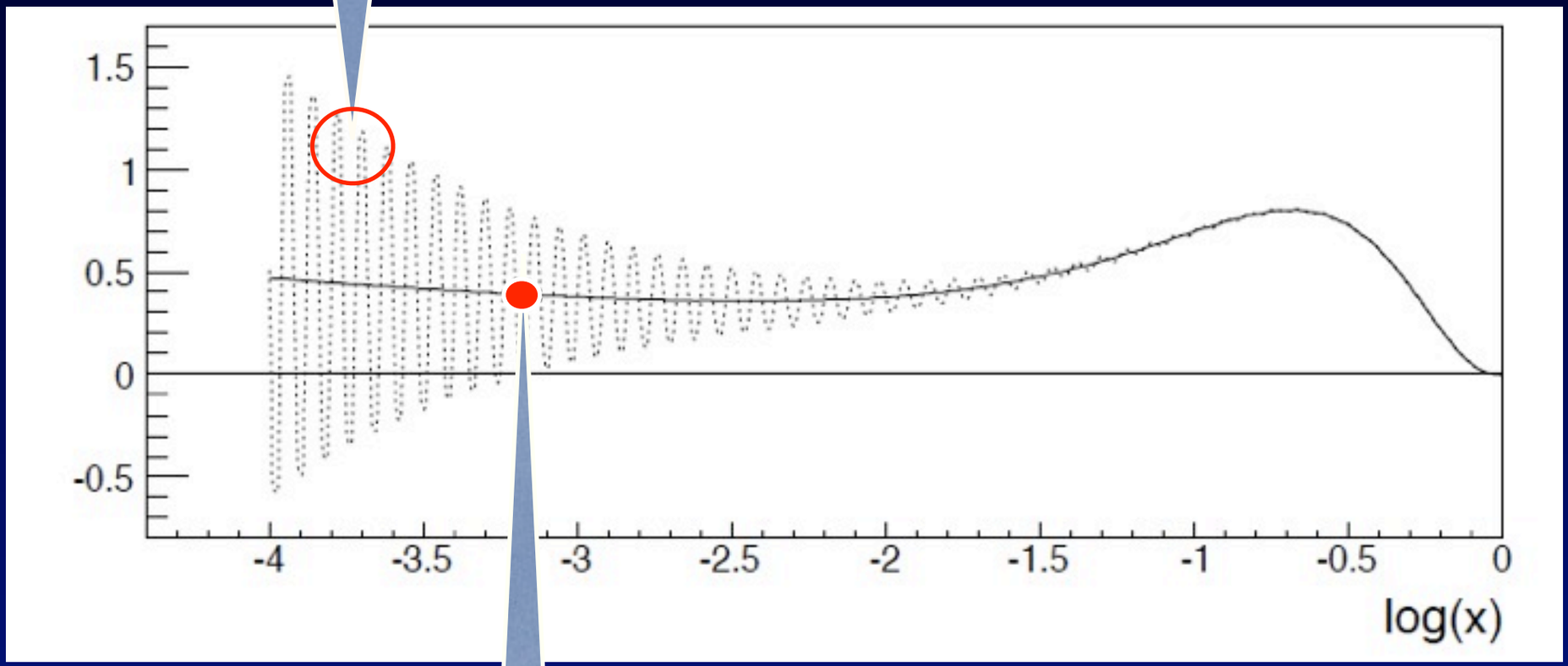


☑ Comparison QCDNUM - PEGASUS O(10^{-4})

A. Vogt, CPC 170 (2005) 65

QCDNUM caveat: backward evolution

Backward evolution with quadratic splines may oscillate
(forward evolution never oscillates neither do linear splines)



QCDNUM can handle this instability but quad backward evolution over a large range in Q^2 is not recommended

Convolution Engine

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

R.S. Thorne and W.K. Tung, arXiv:0903.3861 (2009)

- Uses rescaling variable, like: $\chi \equiv ax = \left(1 + \frac{m_{\mathrm{h}}^2}{Q^2}\right) x$
- C(...) and a(...) must be supplied as Fortran functions
- You can generate weight tables at initialization and then enjoy very fast convolution as weighted sum

- QCDNUM can handle singularities

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

- Each term has its own Makewt routine

Store

Coefficient function

```
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

How to get your Structure Function

- Generate weight tables at initialization
- Write a structure function function, like

```
function stf(ix,iq)
fcc = FCROSSC(w,idw,idf,ix,iq)
stf = GETALFN(iq,n,ierr)*fcc
return
```

- Pass this function to an interpolation routine

```
call STFUNXQ(stf,x,Q2,Fval,1,0)
```

QCDNUM-17-beta-05

- MBUTIL pool of utility routines (incl. write-up)
- QCDNUM evolution program (incl. write-up)
  - ☑ Evolution fully NNLO
  - ☑ Renormalization scale dependence
  - ☑ Convolution engine fully operational
- ZMSTF zero-mass structure function add-on
  - ☑ F_2, F_L, xF_3, F_L' up to NNLO
  - ☑ Factorisation scale dependence
- HQSTF heavy quark stfs in 3-flavour FFNS
  - ☑ F_2, F_L contribution from (c,b,t) up to NLO
  - ☑ Factorisation scale dependence

E. Laenen et al., NP B392 (1993) 162
S. Riemersma et al., PL B347 (1995) 143

Fast?

- 100x50 point 5-fold grid $x > 10^{-5}$ and $Q^2 < 10^4$ GeV^2
- 1000 NNLO evolutions of 11 PDFs (no top) in the VFNS
- For each evolution 1000 NNLO $F_2 + F_L$ in HERA kin range
- Code compiled with gfortran (w/o array boundary chk)
- MacBook 2GHz Intel Core 2 Duo

Routine	Calls	CPU sec	CPU/call
Evolution	1000	42	0.042
F_2, F_L	2.10^6	80	4.10^{-5}

now 8.5 sec

now 18.5 sec

now 10.0 sec

☑ Takes 2 minutes which is pretty fast!

Are we done?

- With this talk, yes
- With QCDNUM, almost:
  - * Timelike evolution (of fragmentation functions)
  - * Evolution of polarised PDFs
  - * Convolution of PDFs (parton luminosities)

On May 8 2010, the non-beta version QCDNUM-17-00 was released, and published in arXiv:1005.1481

☑ Stay tuned on **www.nikhef.nl/~h24/qcdnum**

http://www.nikhef.nl/~h24/qcdnum