Grid-tune of QCDNUM and SPLINT

Michiel Botje

BAT meeting 31-08-2021

New SPLINT routines

- Added a few routines to manage spline objects in memory , see section 7 write-up

`nw = isp_SpSize(ia)`	Get object size
`call ssp_Erase(ia)`	Clear memory
`call ssp_SpDump(ia, 'filename')`	Dump spline to disk
`ia = isp_SpRead('filename')`	Read spline from disk
`call ssp_SpSetVal(ia, i, dd)`	Write info into a spline
`dd = dsp_SpGetVal(ia, i)`	Read info from a spline

- Can write splines to disk and read them back in another run to use as a reference
- Can store extra info in such splines like evolution parameters, particle code, *etc.*
- New version QCDNUM-17-01-82 at `www.nikhef.nl/user/h24/download`

Structure function splines

- Fill 2-dim 100×50 spline with NNLO F_2
- With a $\sqrt{s}$ cut of 300 GeV this takes 4.9 CPU secs (stf computed with `ZMSTFUN`)
- Added a new structure function grid-point routine to `ZMSTF`

```
F = ZMSTFIJ( istf, def, ix, iq, ichk )
```

- It now takes 1.4 CPU secs → grid routine is factor 3 faster
- Still quite slow: introduce fast structure function spline filling routine in `SPLINT` that makes use of the list processing capability of `ZMSTFUN`

```
call ssp_S2F123( ia, iset, def, istf, rs )
```

- Filling now takes 0.013 CPU secs: factor 100 faster!

Timing study of `zmstf`: CPU per structure function

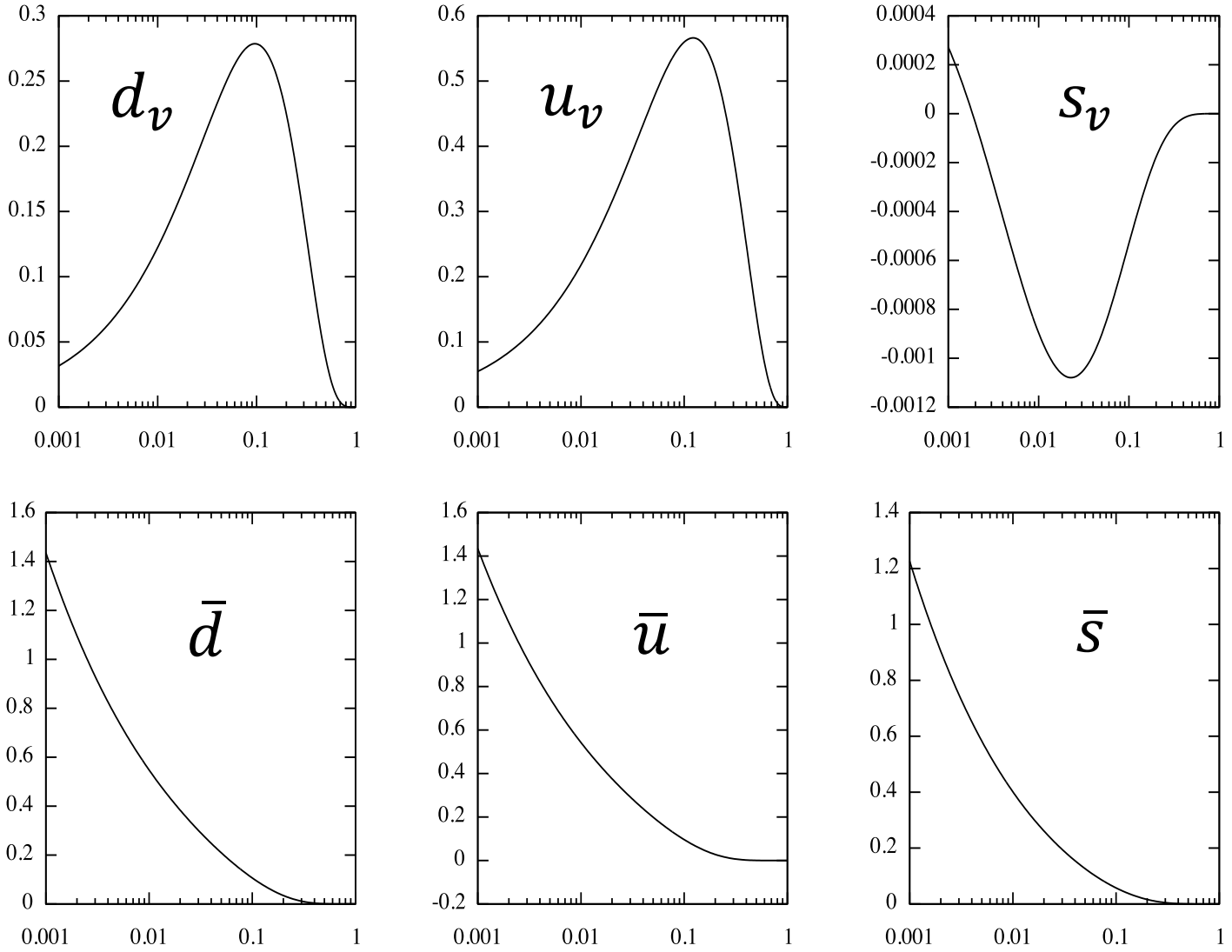
- Random set of grid points or x-Q^2 points
- Compute F_2 in loop or as a list

- `ZMSTFIJ` factor 3 faster than `ZMSTFUN`
- `ZMSTFUN` little difference in on/off-grid loop
- `ZMSTFUN` list is factor 10-100 faster than loop

- `SPLINE` faster than `ZMSTFIJ` for $n > 10$
- Faster than `ZMSTFUN` list for $n > 100$
- Factor 4 faster than `ZMSTFUN` for large n

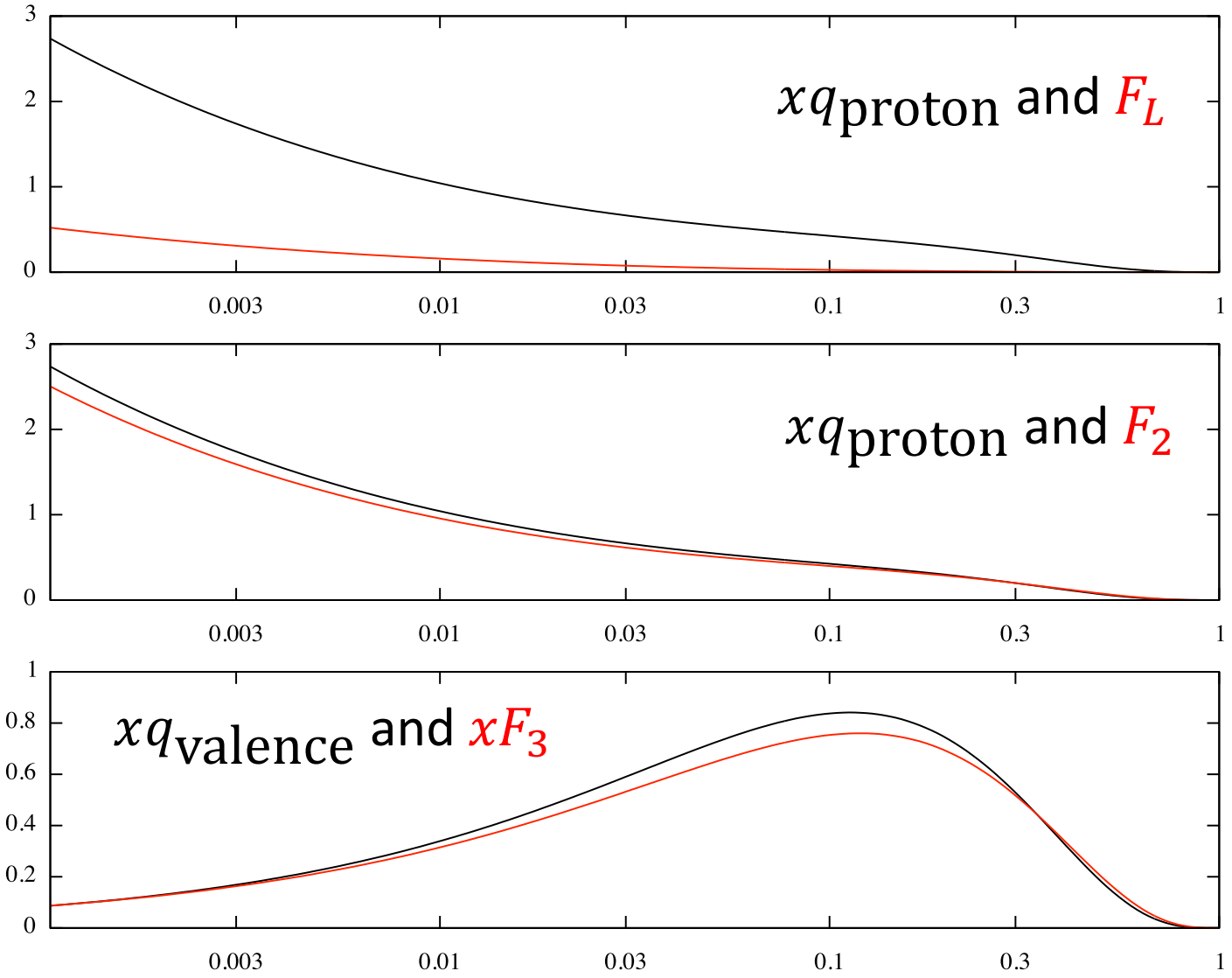
Routine	On/off-grid	Loop/List	n	t/n [ms]
ZMSTFIJ	on-grid	loop	10	0.117
ZMSTFUN	on-grid	loop	10	0.417
ZMSTFUN	off-grid	loop	10	0.437
ZMSTFUN	on-grid	list	10	0.050
ZMSTFUN	on-grid	list	100	0.009
ZMSTFUN	on-grid	list	1000	0.003
ZMSTFUN	off-grid	list	10	0.065
ZMSTFUN	off-grid	list	100	0.014
ZMSTFUN	off-grid	list	1000	0.004
SPLINE	off-grid	loop	10	0.108
SPLINE	off-grid	loop	100	0.014
SPLINE	off-grid	loop	1000	0.001

BATUNE00: Evolve test-job pdfs in VFNS NNLO from 1-100 GeV^2



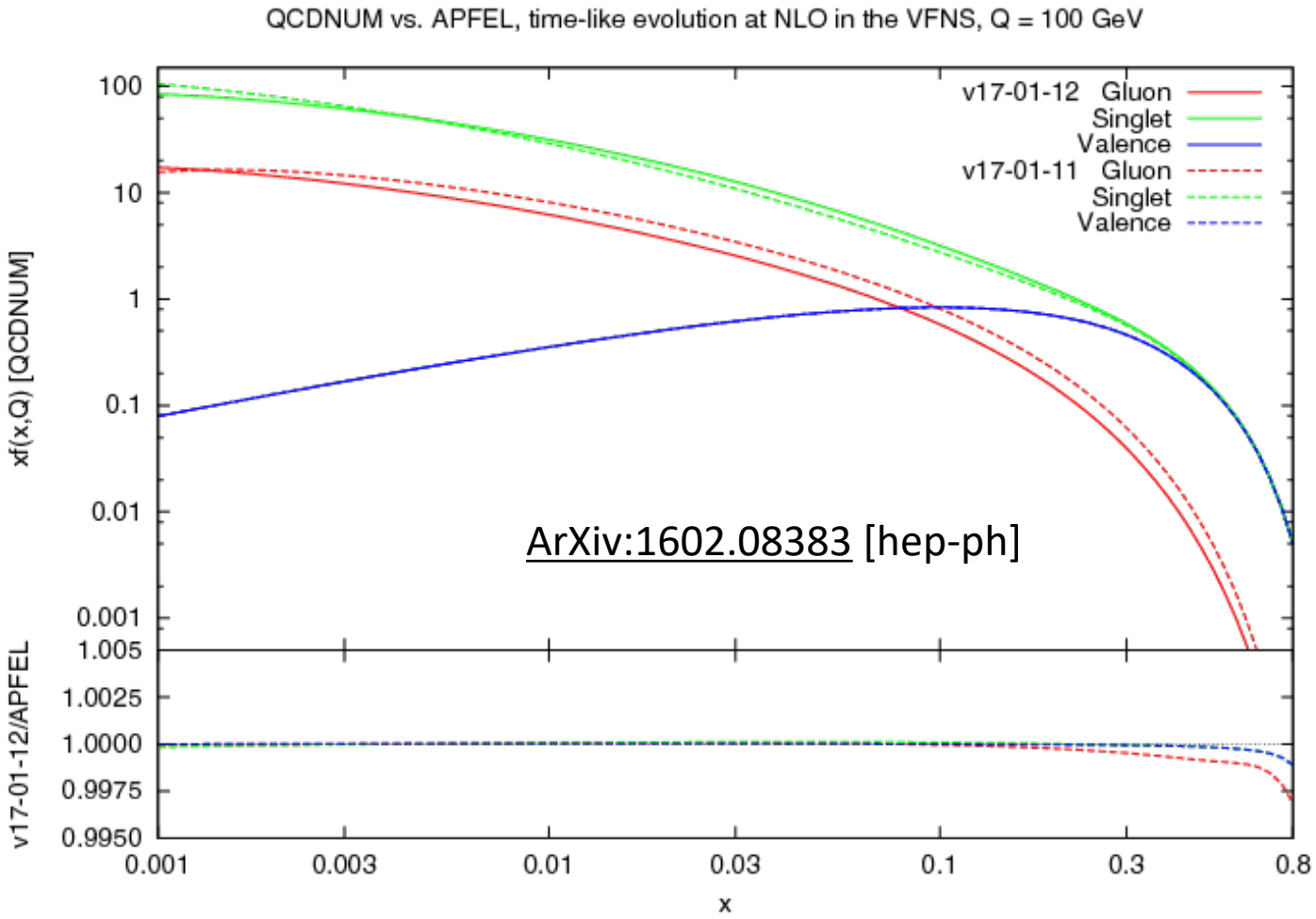
- u, d, s evolved to 100 GeV^2
- c, b not shown
- s, c, b NNLO evolution generates a small valence component with $\bar{q} > q$ at intermediate x
- Starting pdfs for BATUNE01

Pdfs and structure functions versus x at $\mu^2 = 100$ GeV^2



BATUNE01: write high density reference splines

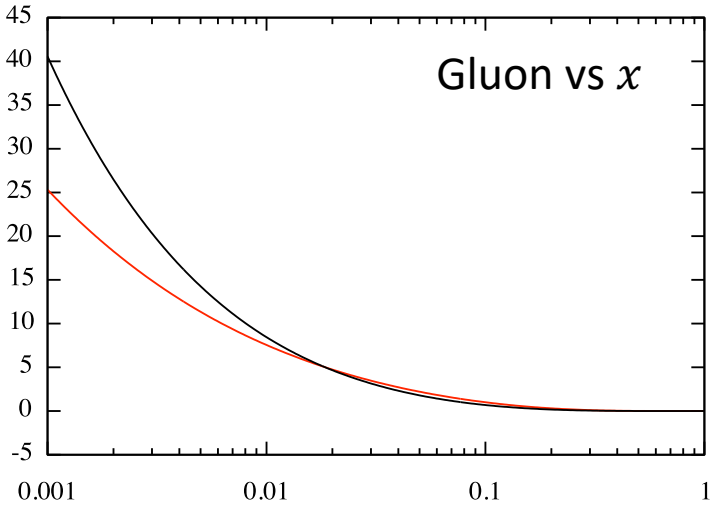
- Plot shows QCDNUM/APFEL comparison with pdfs evolved on a 200 point QCDNUM x grid
- We assume that pdfs evolved on a 300x150 grid are accurate enough for benchmarking
- Write high resolution (100×50 nodes) 2-dim reference splines of
  - Gluon
  - Singlet
  - F_2 proton
  - F_L proton
  - xF_3 valence

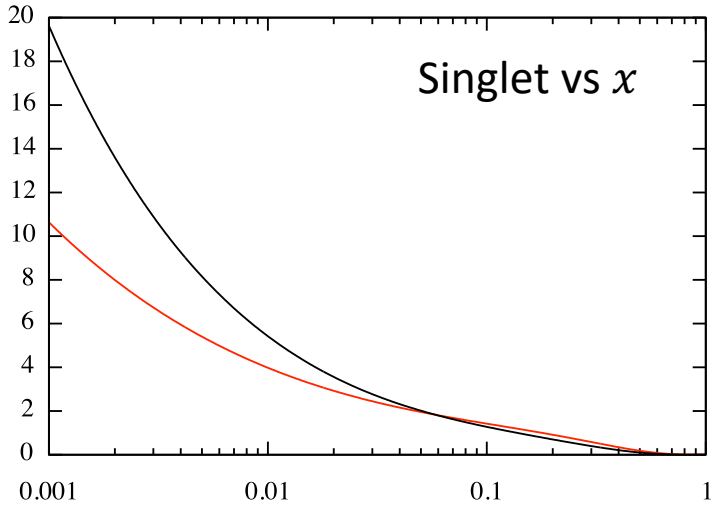


BATUNE01: Evolve from 100-30000 GeV^2 at NNLO

$\mu^2 = 100$ GeV^2

$\mu^2 = 3\times10^4$ GeV^2





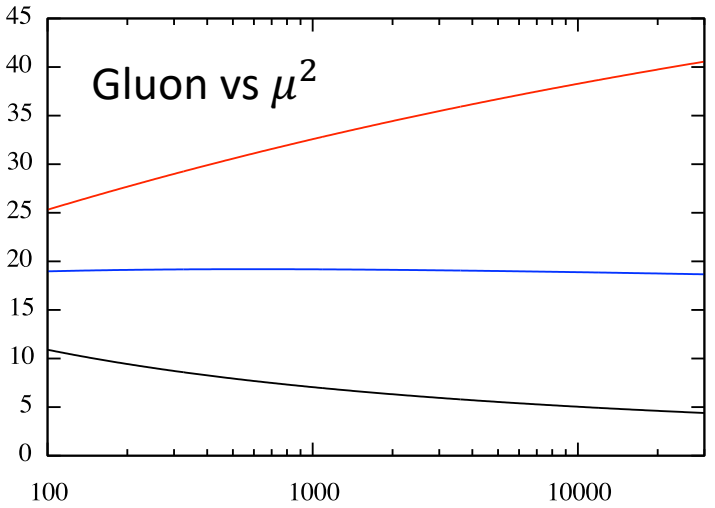
$\mu^2 = 100$ GeV^2

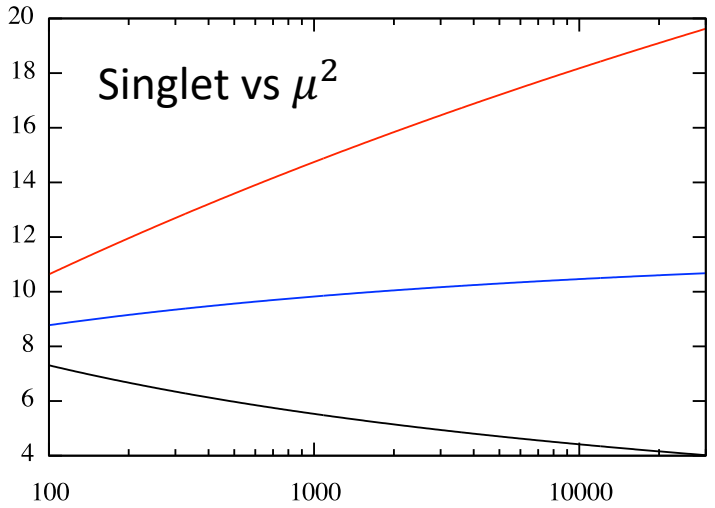
$\mu^2 = 3\times10^4$ GeV^2

$x = 10^{-3}$

$x = 0.02$ $(\times4)$

$x = 0.5$ $(\times1000)$





$x = 10^{-3}$

$x = 0.02$ $(\times3)$

$x = 0.5$ $(\times40)$

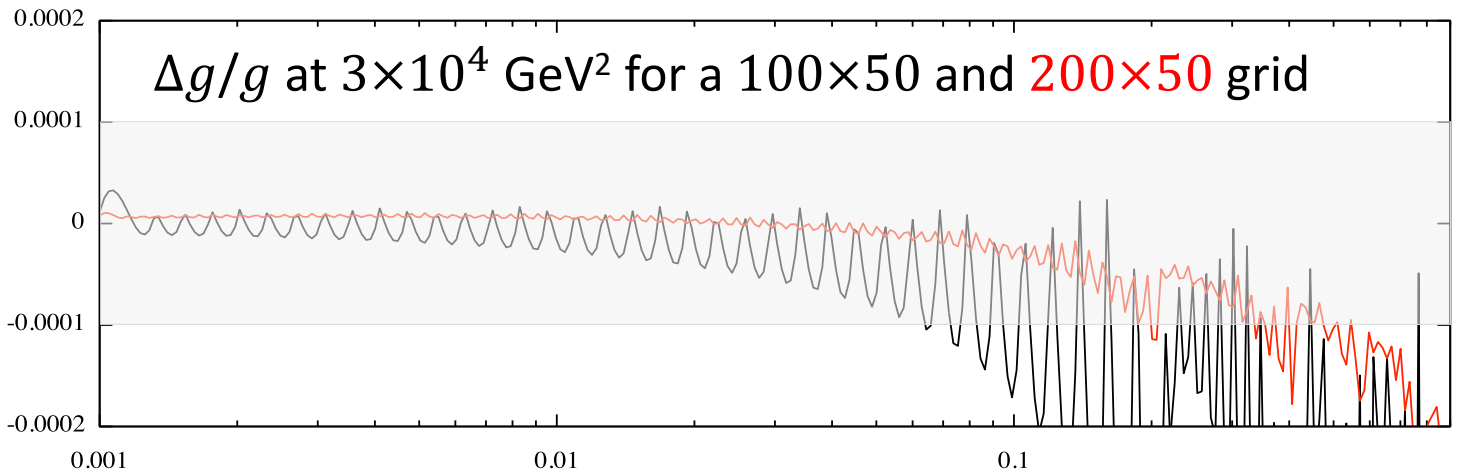
Almost pure logarithmic μ^2 dependence → do not need many grid/node points

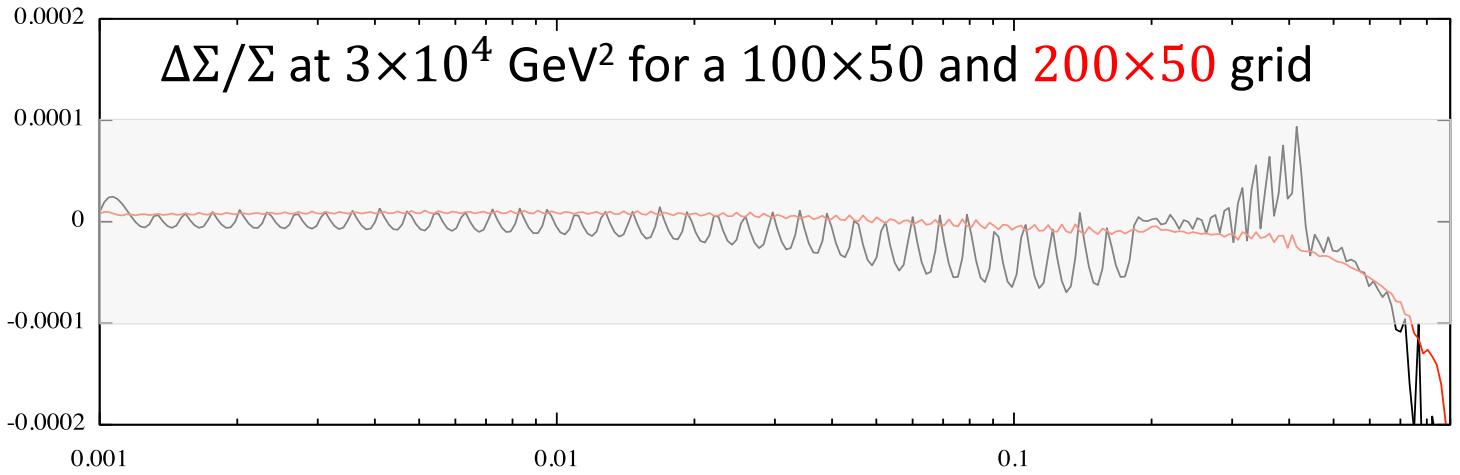
BATUNE02: Tune x-μ^2 grid with respect to reference splines

Subgrids in x

Lower Limit	Weight
10^{-3}	1
0.2	2
0.4	4
0.6	8
0.8	16

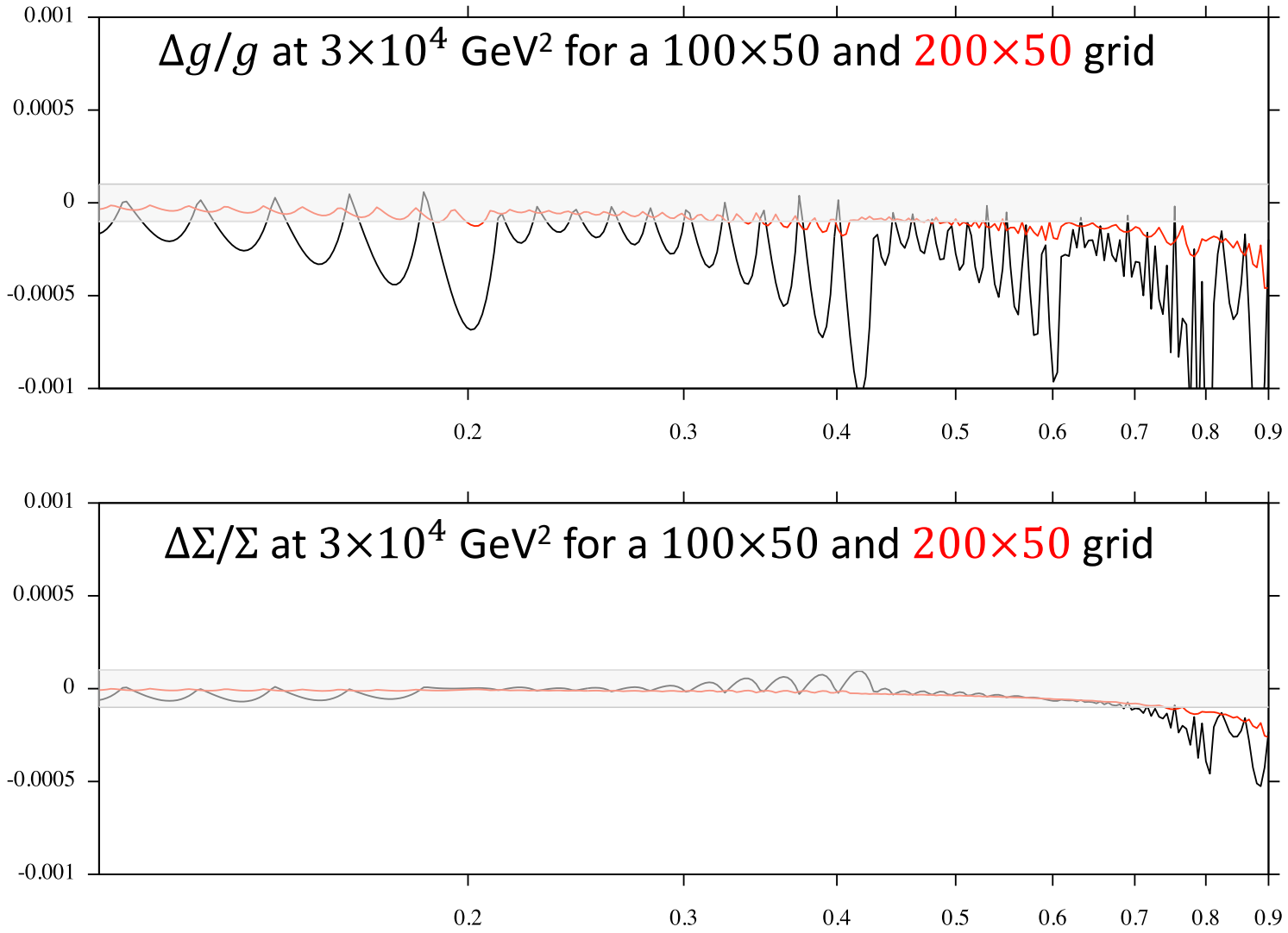
- 200x50 grid evolve in 12.8 ms
- 100x50 grid evolve in 5.1 ms
- $\Delta f/f < 10^{-4}$ for $x < 0.1$
- Bit larger deviation at large x where pdfs are very small





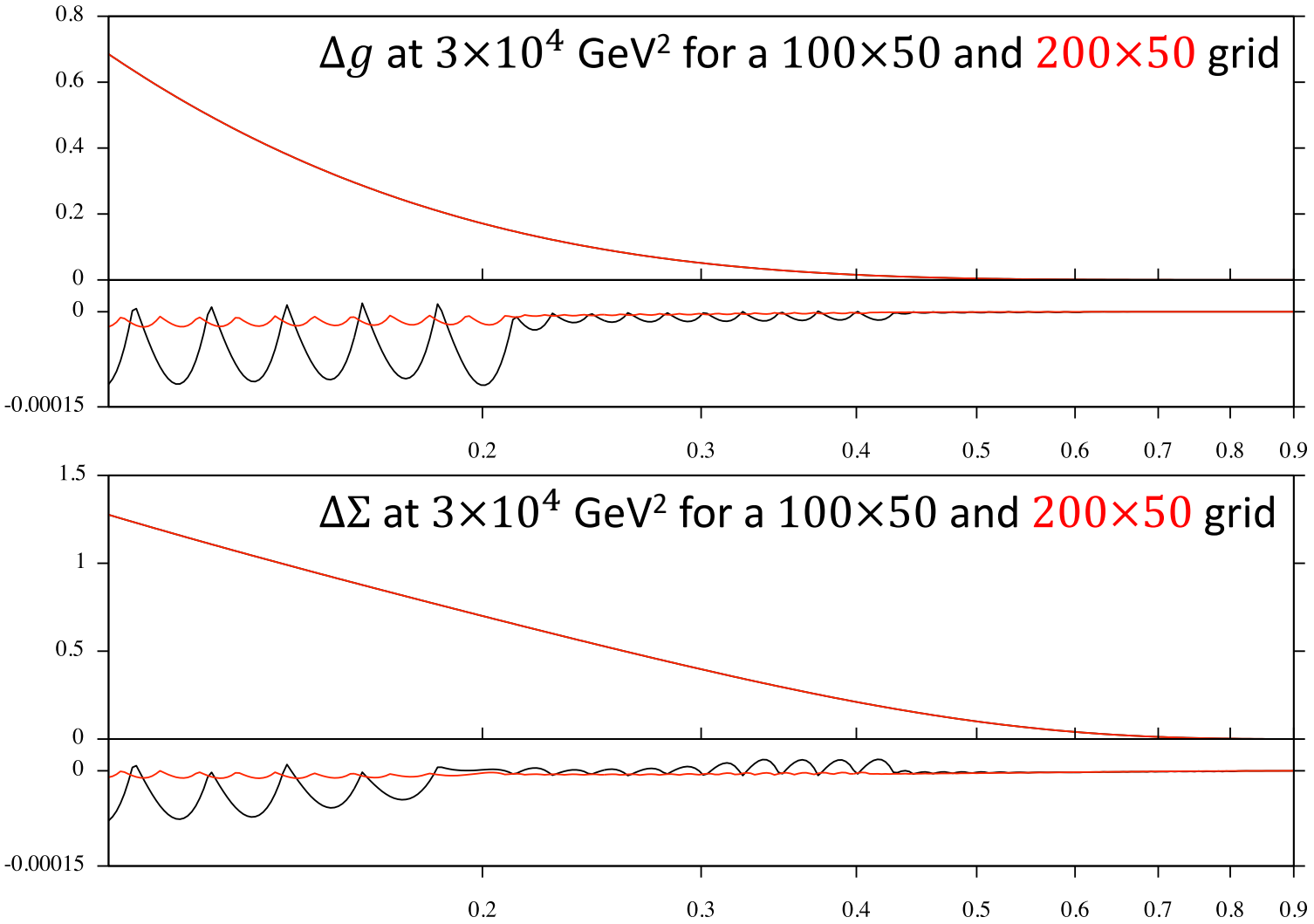
BATUNE02: $\Delta g/g$ and $\Delta\Sigma/\Sigma$ at large x

- 200x50 evolution accurate to 10^{-4} up to large x
- Also true for 100x50 singlet
- 100x50 gluon less accurate but still within 10^{-3}
- No need for high accuracy since gluon vanishes quickly at large x



BATUNE02: Absolute deviation Δg and Δs at large x

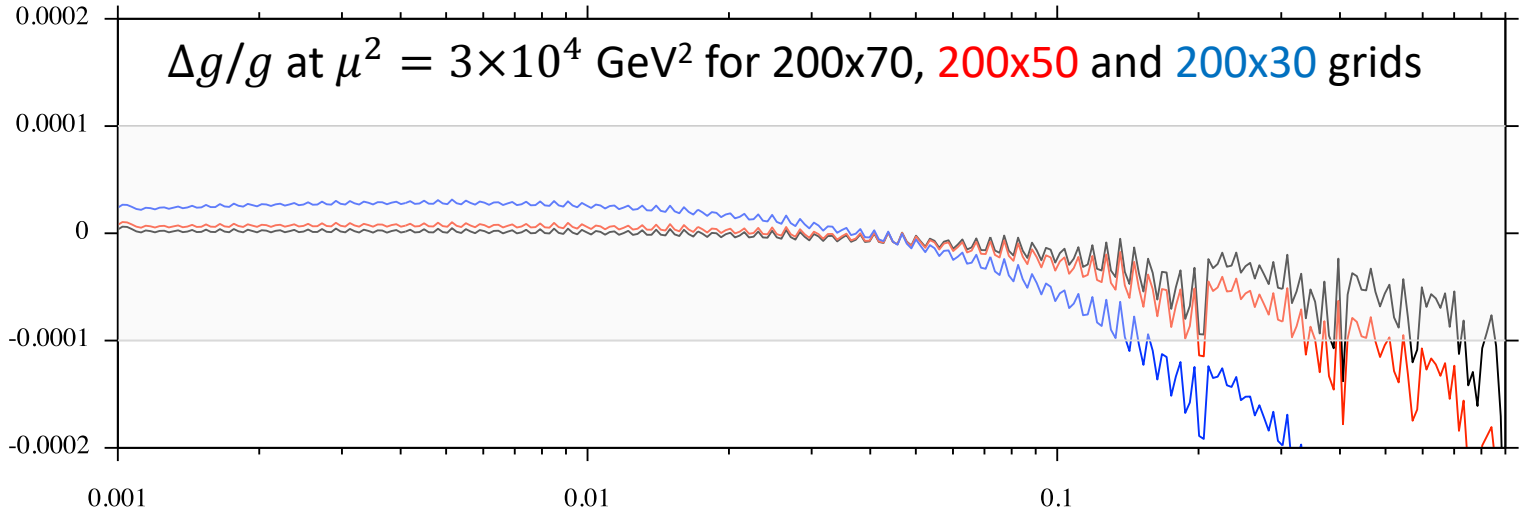
- Gluon vanishes faster than singlet at large x
- Tuning for large relative gluon accuracy is not useful here
- 100x50 is my preferred grid

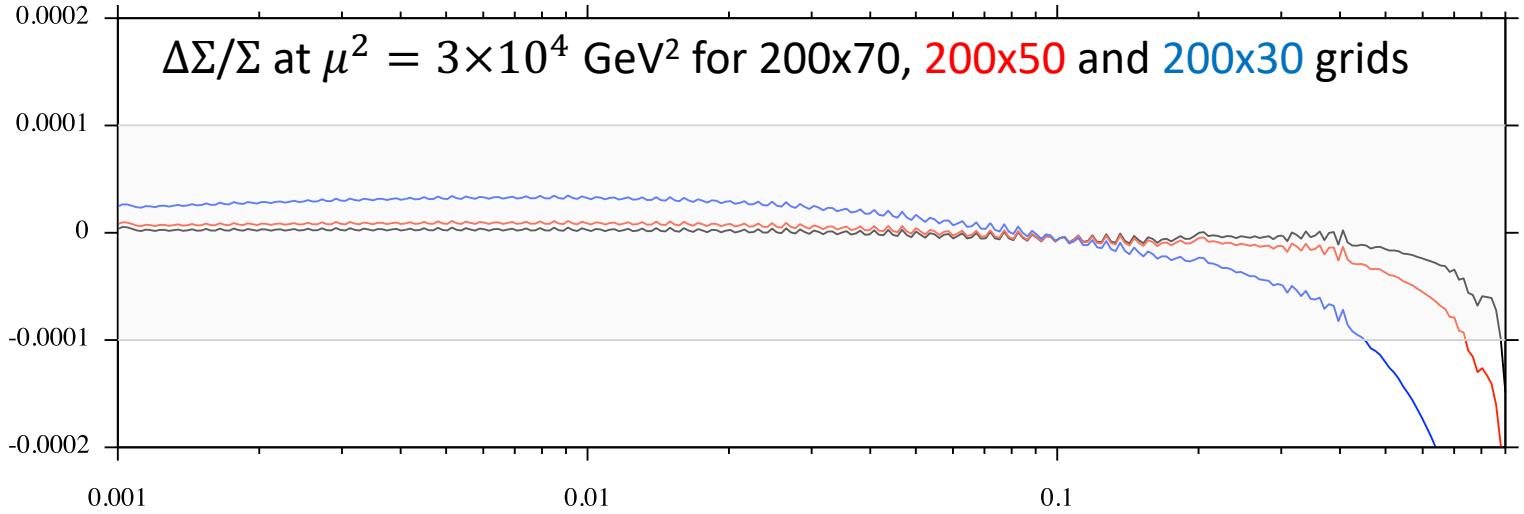


BATUNE02: Vary the number of μ^2 grid points

- Take $n_x = 200$ to have clear plots of $\Delta f / f$
- Small bias develops for $n_q < 50 \rightarrow$ I prefer $n_q = 50$
- Here are the CPU times of evolution with various grid settings

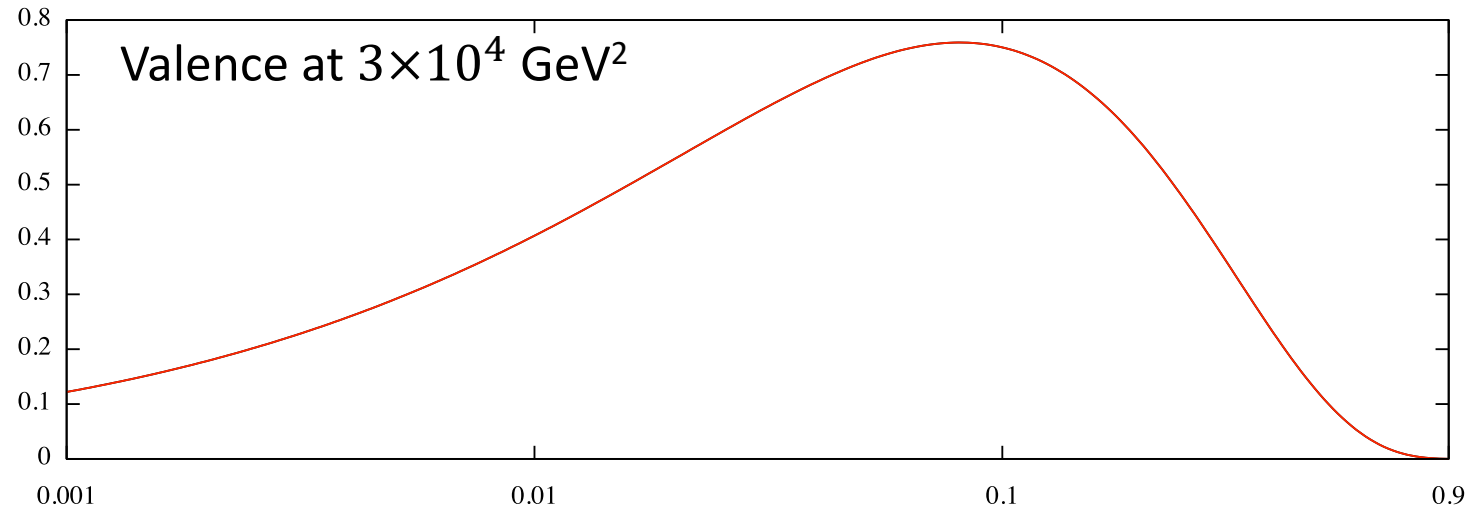
n_x	n_q	t [ms]	factor
300	150	56.8	11.1
200	90	22.3	4.4
200	70	17.6	3.5
200	50	12.8	2.5
200	30	8.1	1.6
100	50	5.1	1.0

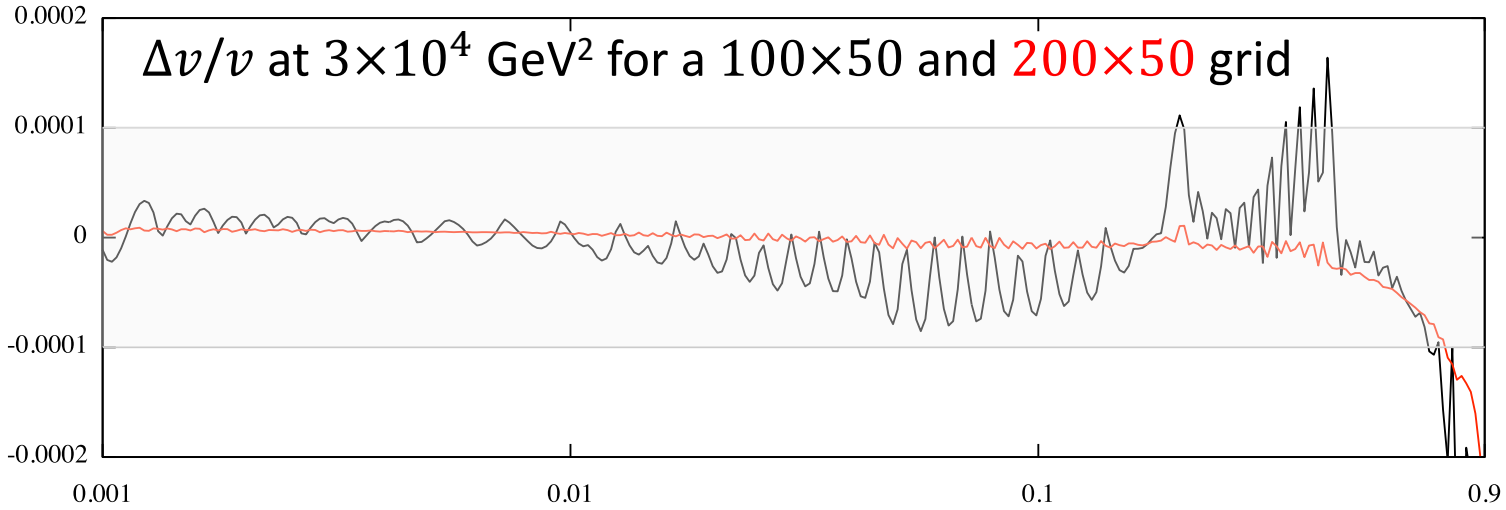




BATUNE02: Valence pdf and $\Delta v/v$

- Tuning also OK for the valence pdf





QCDNUM grid tuning conclusion

Sub-grids in x

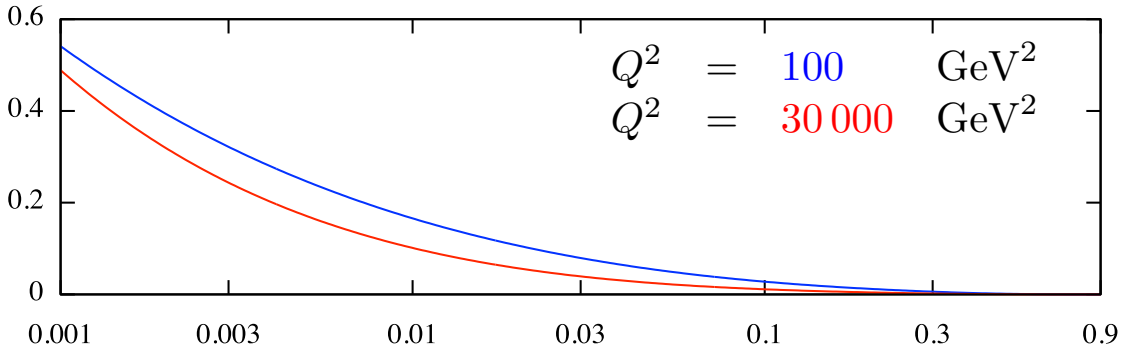
Lower Limit	Weight
10^{-3}	1
0.2	2
0.4	4
0.6	8
0.8	16

- My preferred grid is 100 points in x and 50 points in μ^2 with 5 sub-grids as given in the table
- NNLO evolution on this grid costs about 5 msec in CPU
- Numerical error $\Delta f/f < 10^{-4}$ for $x < 0.1$
- Bit larger relative deviation at large x where pdfs are very small

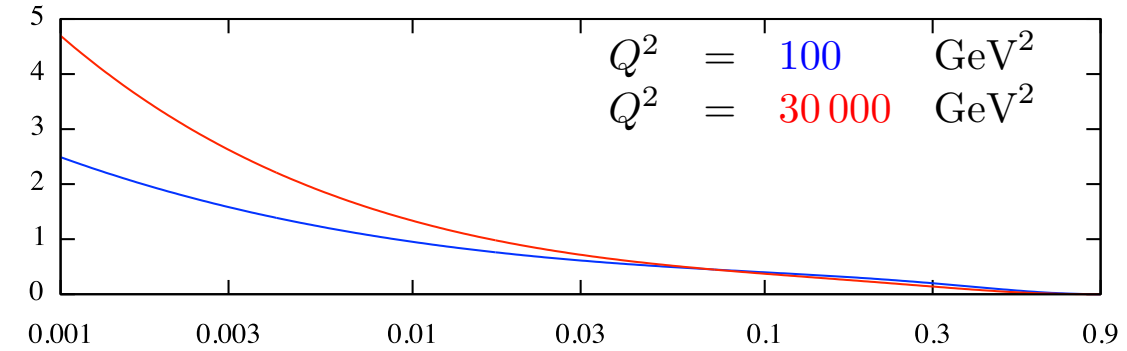
Increasing to $n_x = 200$ gives ~ 10 times better accuracy with double the cost in CPU
Increasing n_q does not give much gain in precision

BATUNE03: Read structure function reference splines

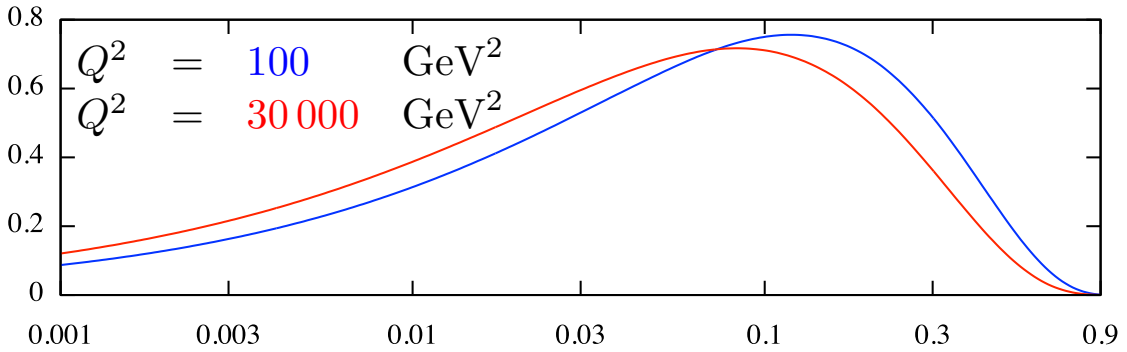
F_L proton



F_2 proton

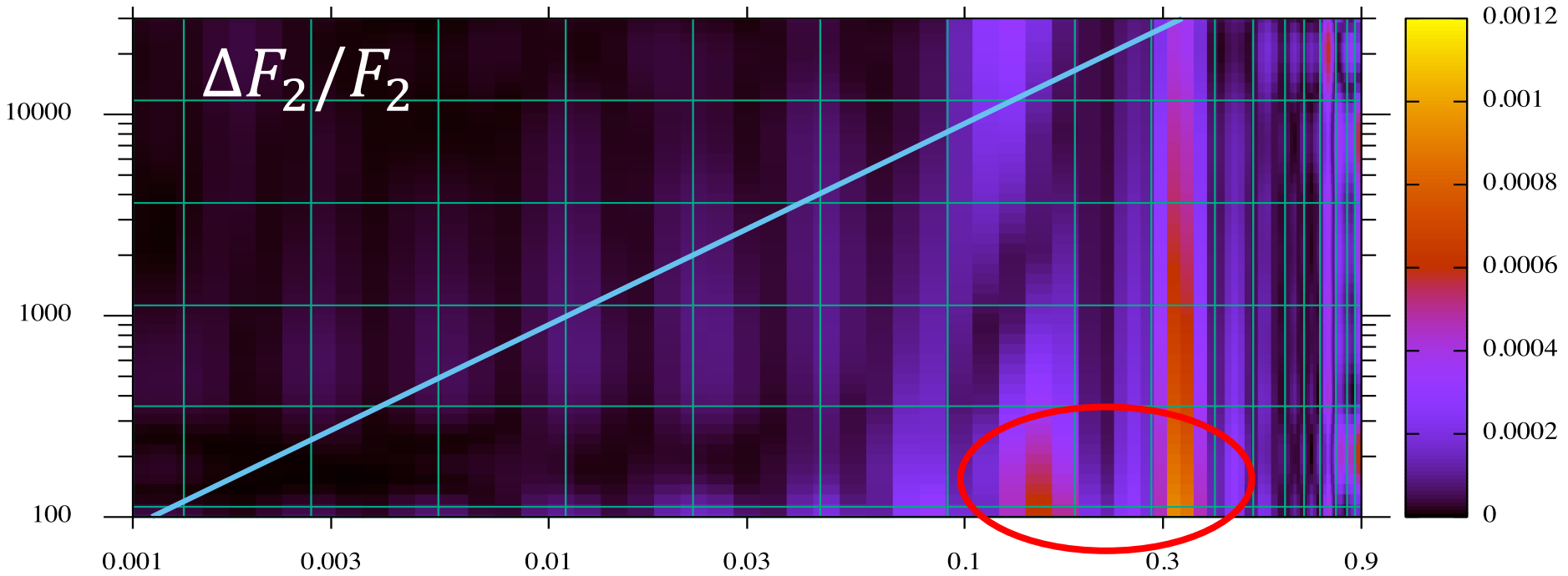


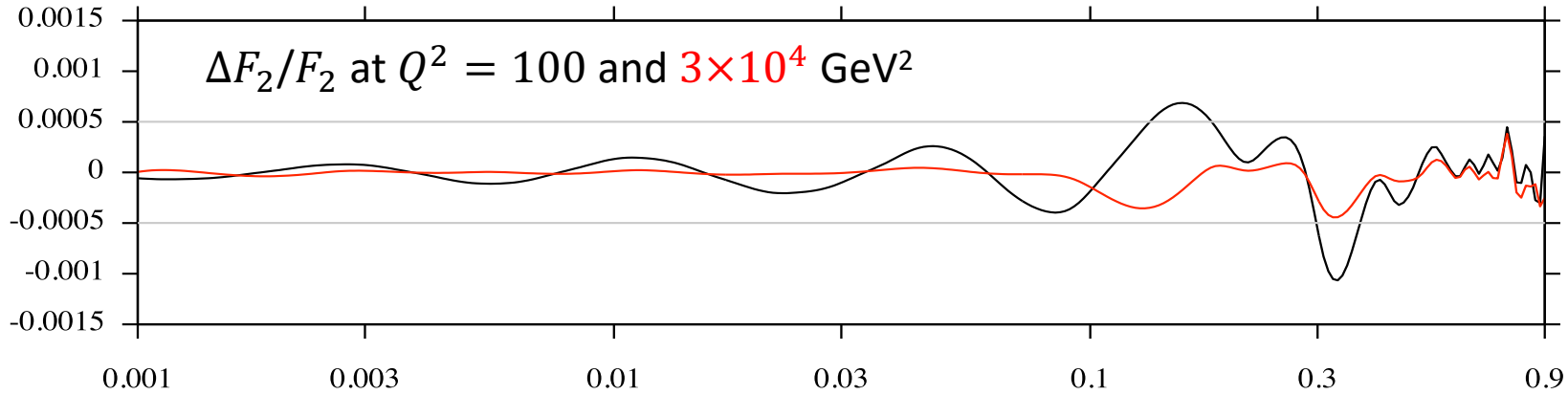
xF_3 valence



BATUNE03: tune proton F_2 spline

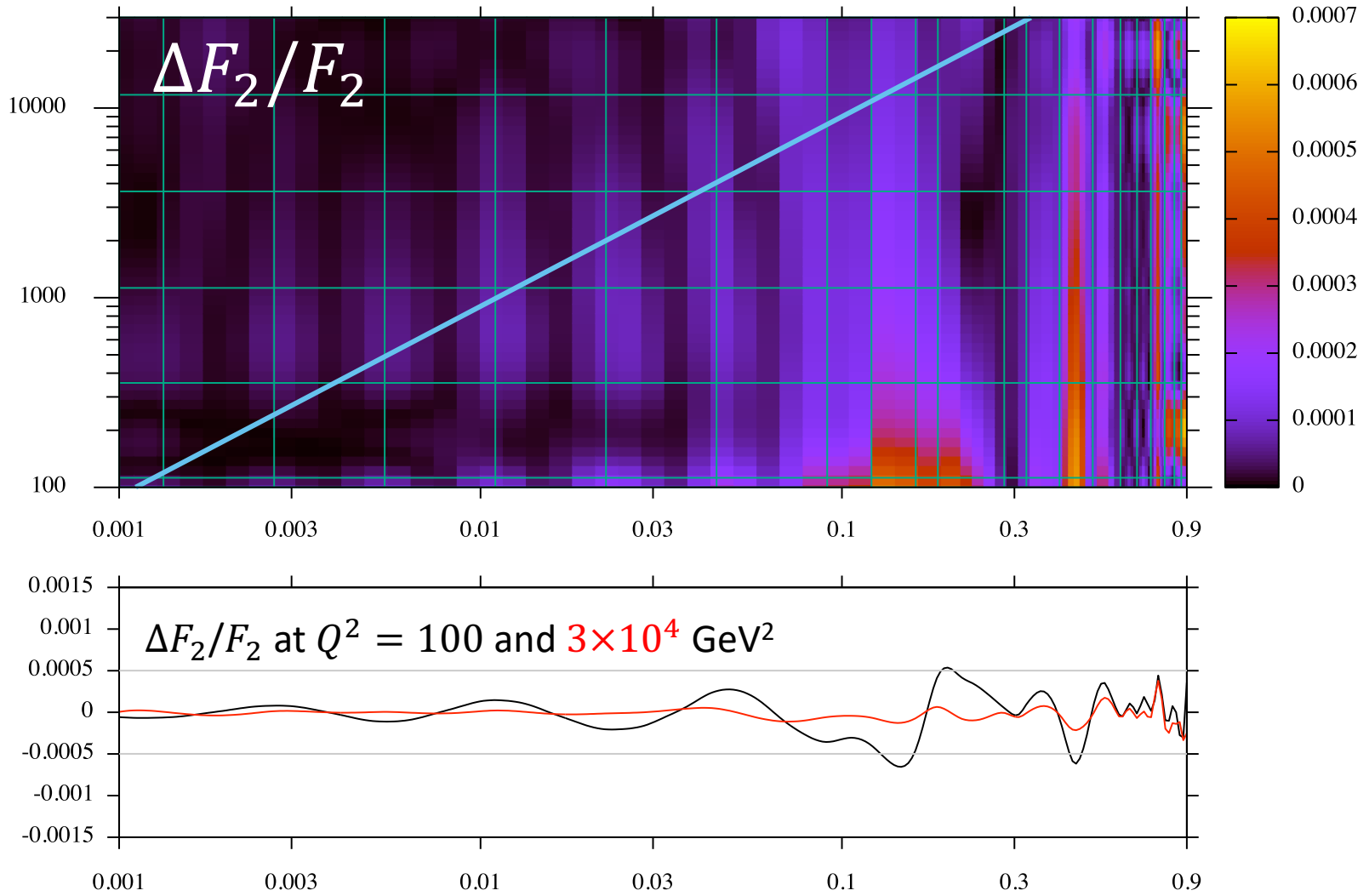
- Step-x = 5 and step-q = 10 gives 22×7 nodes
- Add 3 nodes to suppress hot spots at $x \sim 0.2$



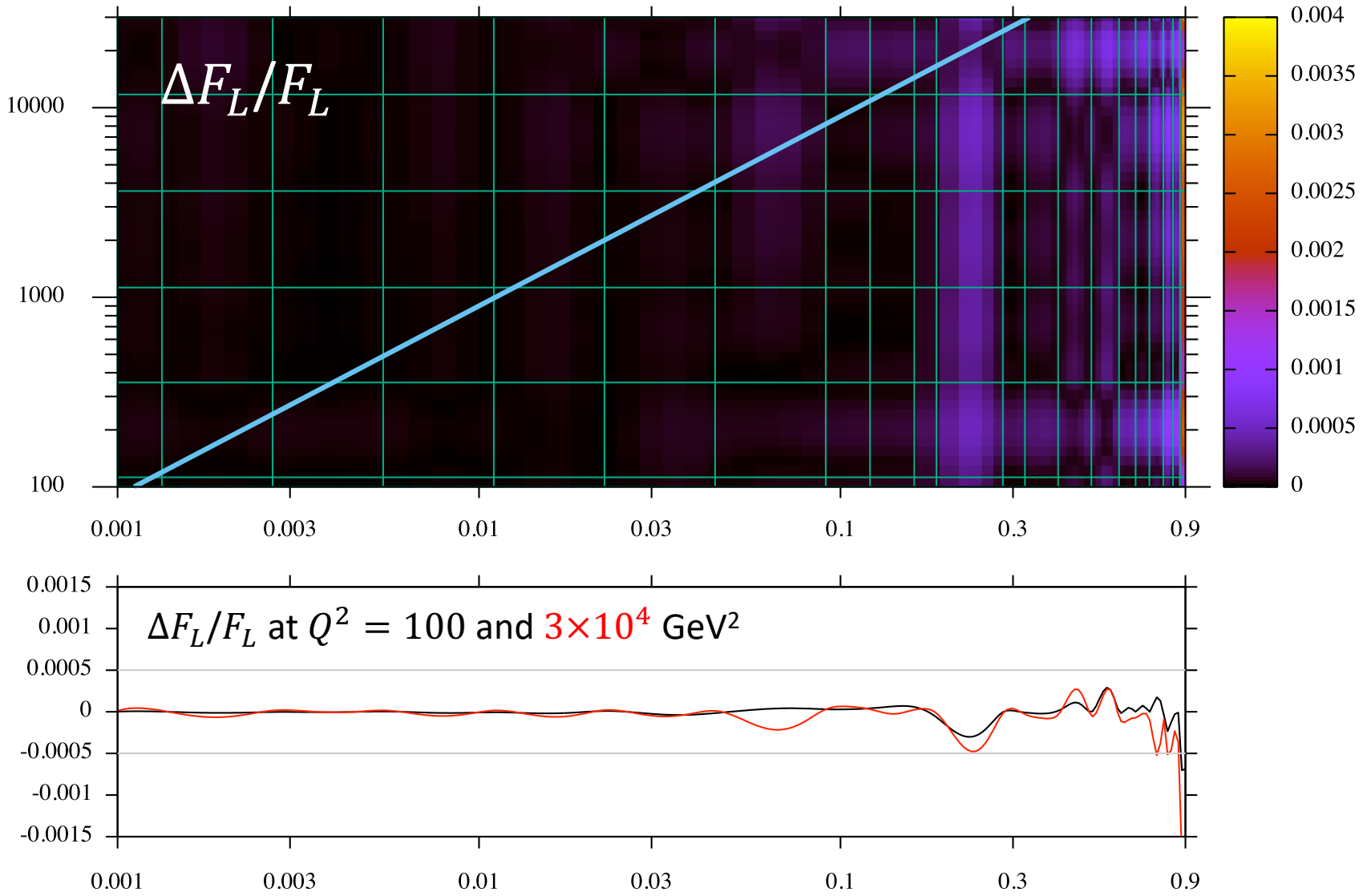


BATUNE04: proton $\Delta F_2/F_2$ on a 25×7 node grid

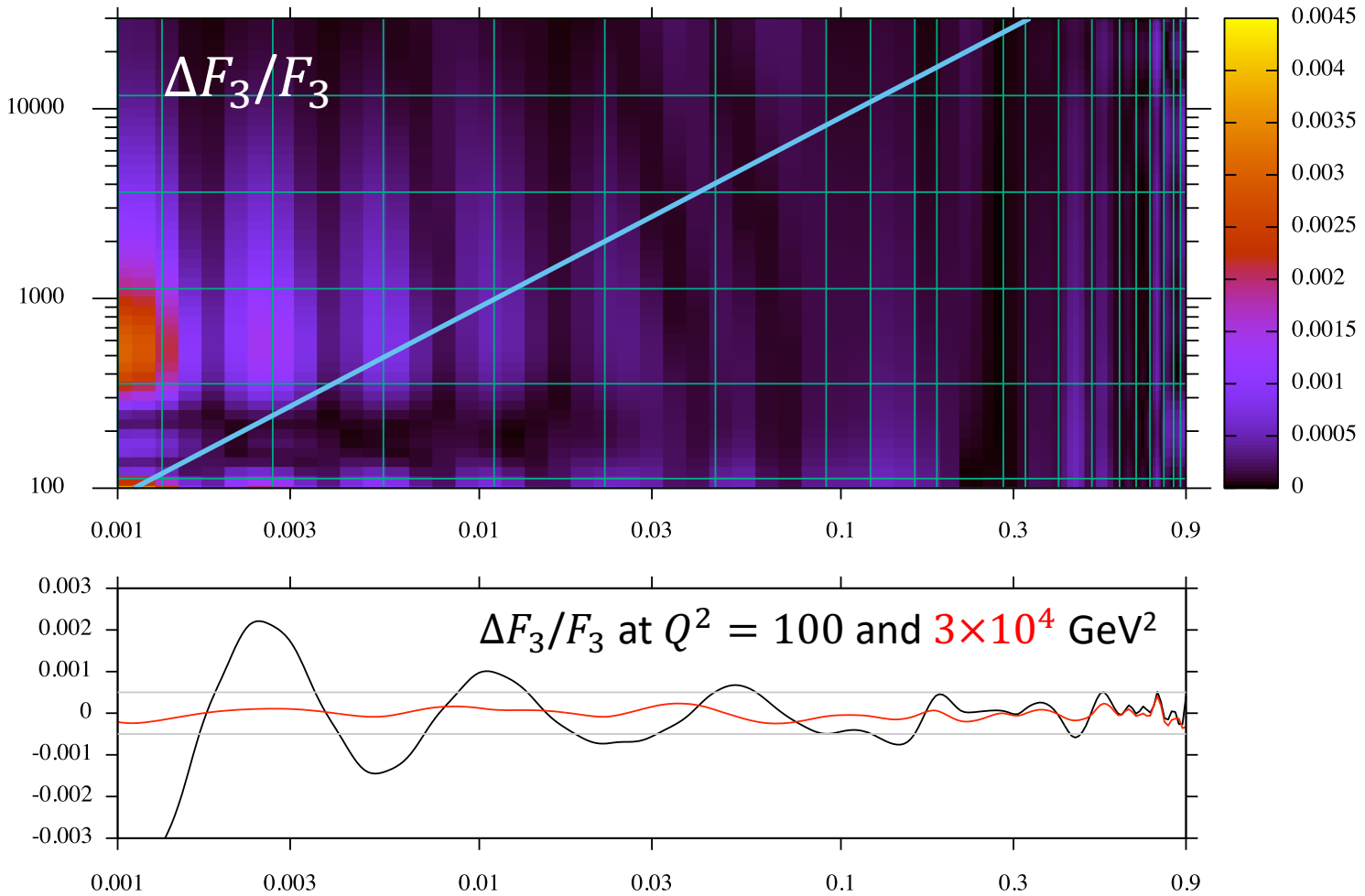
- Add nodes at $x = 0.13$, 0.15, 0.33 to suppress some minor hot-spots
- Slight improvement
- $\Delta F_2/F_2 < 5\times10^{-4}$ almost everywhere



Node-tuning also fine for proton F_L …



… and acceptable for valence xF_3



NB: there is only a small contribution from xF_3 to the cross-section at high Q^2

BATUNE04: Structure function splines for xsec (25×7 nodes)

- F_L for uptype and downtype $\sum(q + \bar{q})$
- F_2 for uptype and downtype $\sum(q + \bar{q})$
- xF_3 for uptype and downtype $\sum(q - \bar{q})$

3 msec for 6
25×7 splines

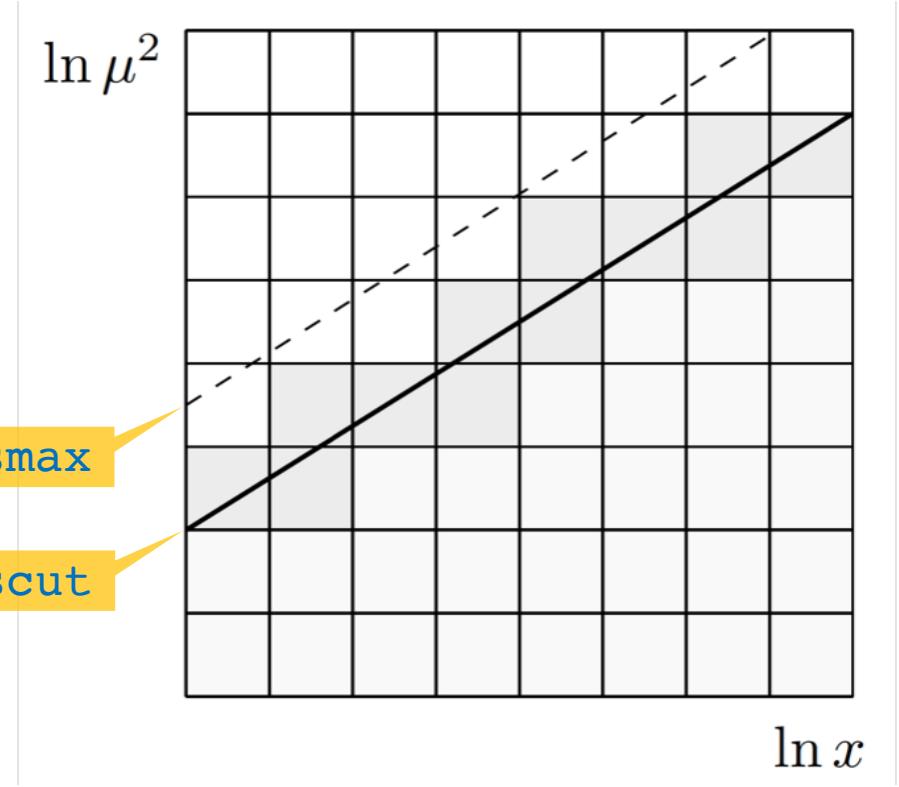
Simplified cross-section

$$\begin{aligned} F_{2,\mathrm{L}} &= \tfrac{1}{9}F_{\mathrm{dtype}} + \tfrac{4}{9}F_{\mathrm{utype}} & y &= Q^2/xs \\ xF_3 &= 0 & Y_\pm &= 1 \pm (1-y)^2 \end{aligned}$$

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}x\mathrm{d}Q^2} = \frac{2\pi\alpha^2}{xQ^4}\left[Y_+F_2 - y^2F_\mathrm{L} + Y_-\,xF_3\right]$$

$\sqrt{s}$ cut revisited

- In the transition region between `rscut` and `rsmax` the spline is not available everywhere
- This region should be avoided when reading a spline to fill another spline
- Cure: set the $\sqrt{s}$ cut of the source spline above the `rsmax` of the target spline

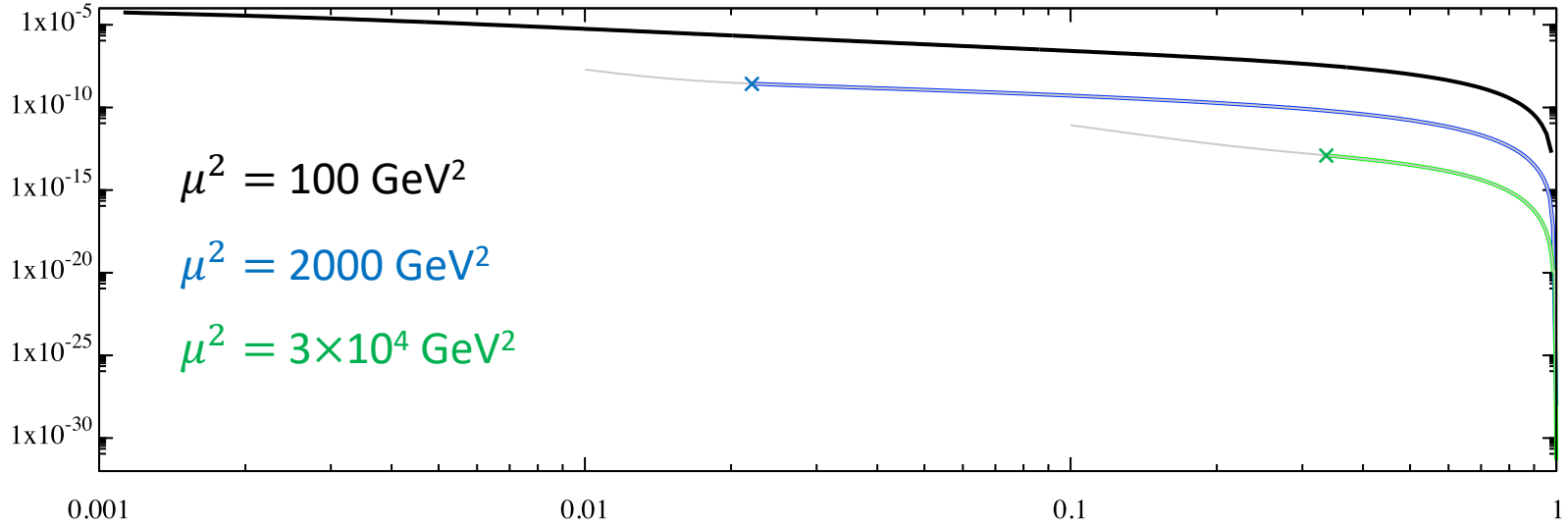


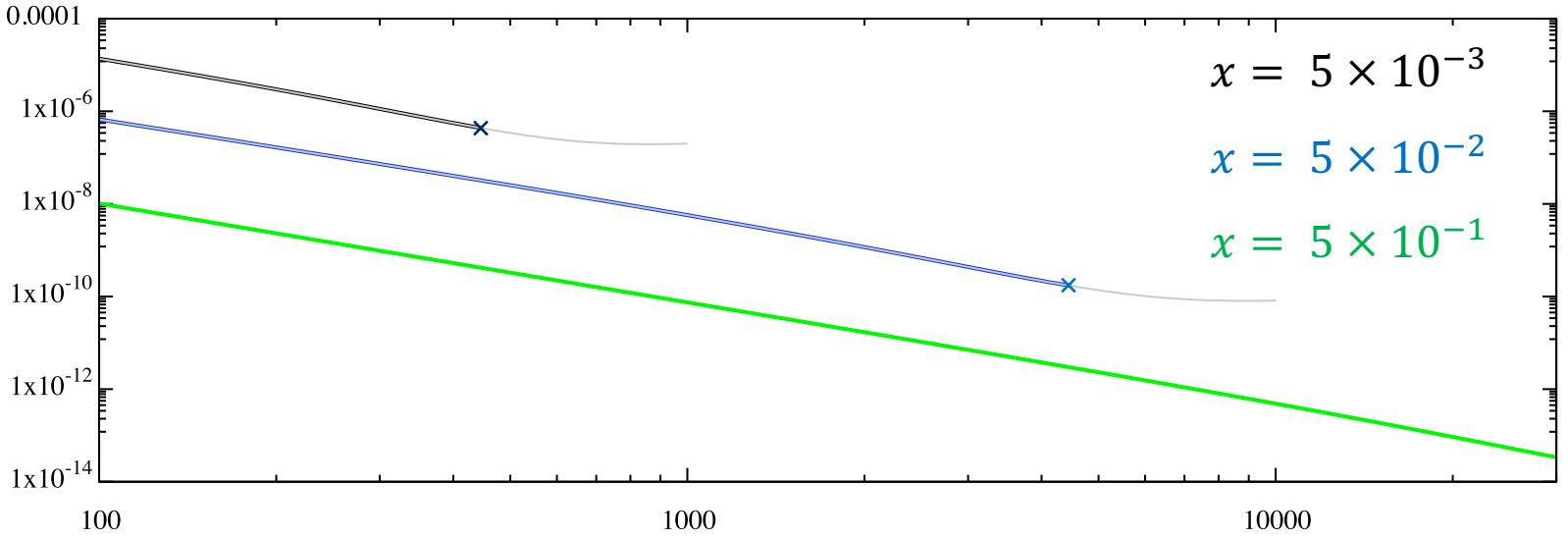
Add two new `SPLINT` routines

```
rsc = dsp_RsCut(ia)        rsm = dsp_RsMax(ia)
```

BATUNE04: compute (simplified) cross-section

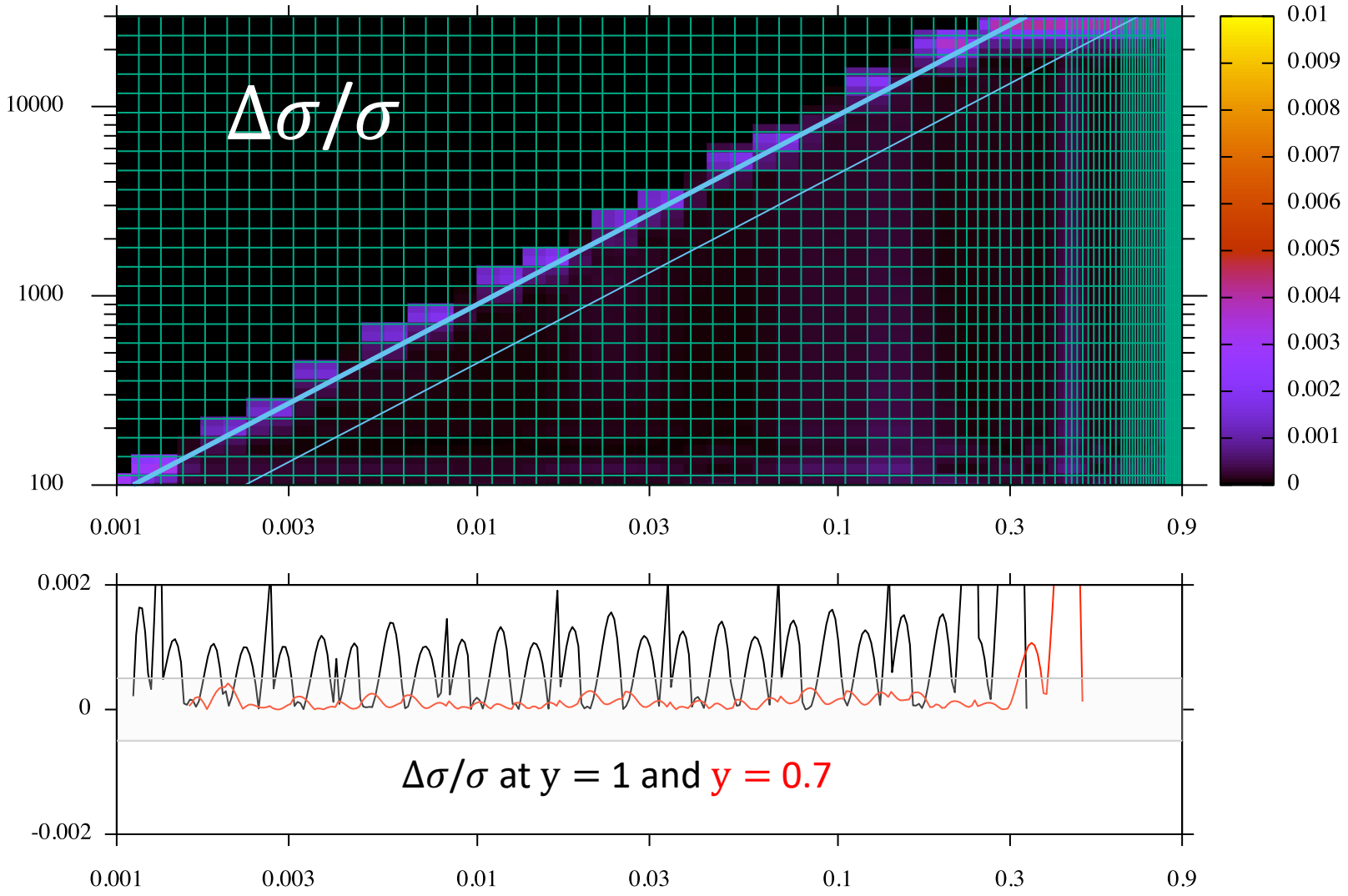
- Very steep fall-off over several orders of magnitude
- Such steep dependence can not be splined accurately on a coarse node-grid
- The grey curves show that the cross-section extrapolates smoothly beyond the kinematic cut → no further action required
- Next: find optimal nodes for cross-section spline





Cross-section spline with 100×25 nodes

- No $\sqrt{s}$ cut at input structure function splines (not worth it)
- Set step-x = 1 and step-q = 2
- $\sqrt{s}$ cut at 300 GeV
- $\Delta\sigma/\sigma < 5{\times}10^{-4}$ except along the kinematic cut ($y = 1$)
- Since cross-section smoothly extrapolates we can raise the $\sqrt{s}$ cut to improve at $y = 1$

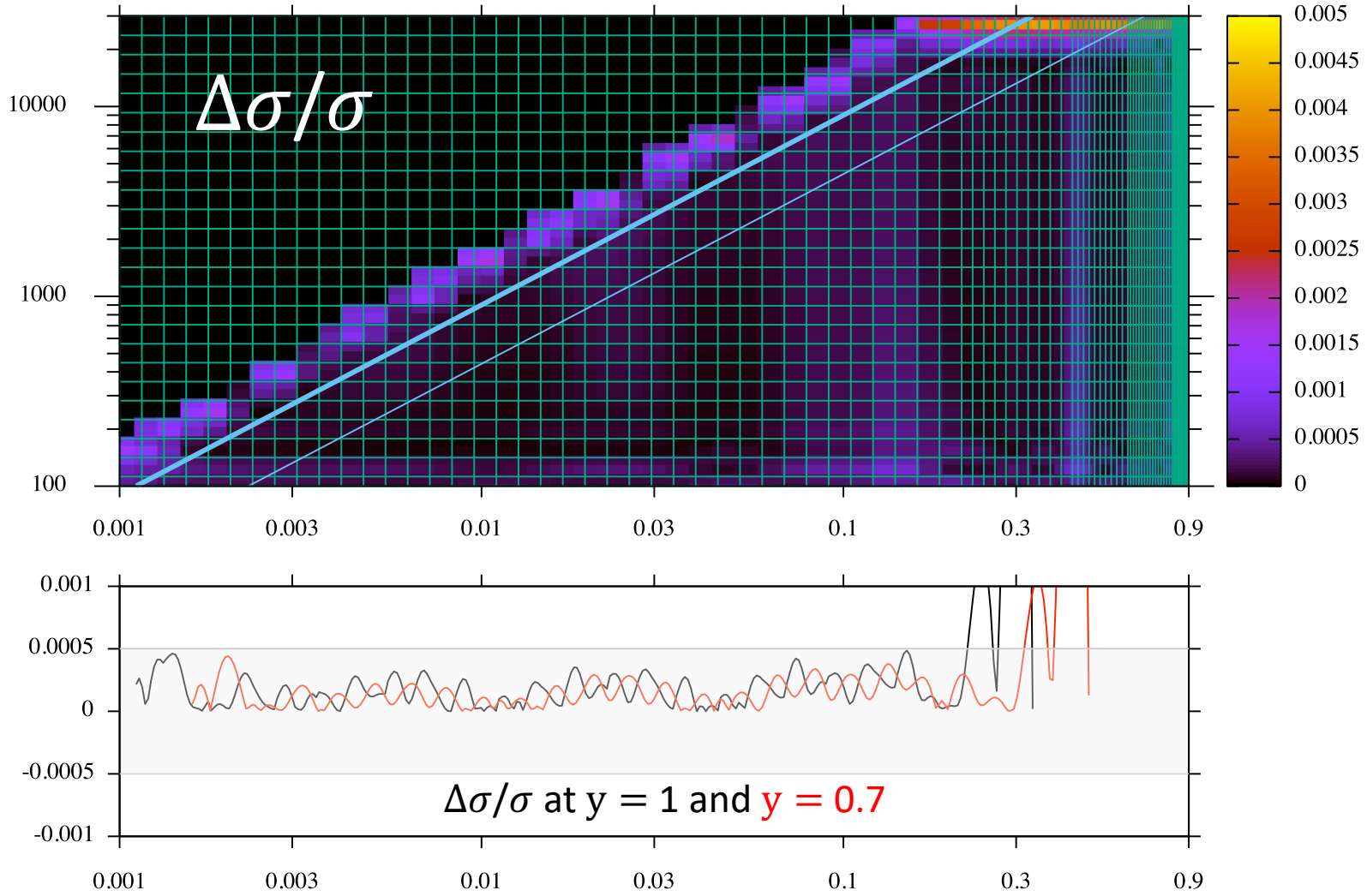


Cross-section spline with 100×25 nodes

- Set step-x = 1 and step-q = 2
- $\sqrt{s}$ cut at 370 GeV
- $\Delta\sigma/\sigma \lesssim 5\times10^{-4}$ along the kinematic cut ($y = 1$)

Timing (optimised code)

	n_x	n_q	t [ms]
Evolution	100	50	3.6
6 Stf splines	22	7	2.9
	100	50	4.5
Xsec spline	100	25	2.2
	50	25	1.2



Changes in the QCDNUM distribution

- Stand-alone write-up for `ZMSTF` (and `HQSTF` in the make)
- All write-ups are now collected in the `/doc` directory
- C++ test jobs are now in `/testjobs` (no more `/testjobsCxx`)
- Running jobs from the `/run` directory:
  - Must specify extension `.f` or `.cc`
  - Can specify source code directory (default `../testjobs`)

```
bash> ./runtest example.f
bash> ./runtest example.cc
bash> ./runtest mydir/myjob.cc
```

BATUNE goodies in the QCDNUM-17-01-82 distribution

- `BAT` directory
  - `batune00` – `batune04.f` tuning jobs
  - `batxxpyy.plt` selection of gnuplot plotting macros
  - `btxxyyyy.dat` selection of gnuplot ASCII input files

- Can do straight-forward check of QCDNUM (without having to plot)

```
bash> cd qcdnum-17-01-82/run
bash> ./runtest ../BAT/batune00.f
bash> diff ../BAT/batune00.dat ../plots/batune00.dat
```

- The unix diff should give no differences

- My suggestion is to convert `batune00.f` to JULIA and do the same test to check for oscillations

Tuning OK so what's next

- Timing of bin-integrations
- For this I need an ASCII file with bin limits $x_1, x_2, \mu_1^2, \mu_2^2$
- Some streamlining of `SPLINT` code
- Bin integration at kinematic limit