Status of the SPLINT package

- Pre-release qcdnum-17-01-80 on

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

- SPLINT package sits in splint sub-directory
- Write-up can be found in splint/doc
- Include file in splint/inc/splint.inc with memory size parameter

```
parameter (nw0=200000)
```

- Splines are created dynamically, provided that memory is large enough
- Adjust `nw0` and recompile if you run out of space (error message)
- Pre-release: few things still missing but otherwise FORTRAN is OK
- C++ interface still being checked, found several typos, don't use yet

2-dim spline

```
ssp_spinit(nuser);                          //initialise
ia = isp_s2make(istepx,istepq);             //new spline object
ssp_s2fill(ia,sfun,rs);                     //sfun → spline
val = dsp_funs2(ia,x,q,ichk);               //spline function
val = dsp_ints2(ia,x1,x2,q1,q2);            //integrate
```

1. Can reserve `nuser` words of user space (use as common block to pass information)
2. Take every nth evolution grid-point as node of the spline (`istep`)
3. The memory address `ia` is an array index and not a C++ pointer
4. Input function `sfun(ix,iq,first)`- see write-up for how to code
5. Can enter limit $\mu^2 \leq xs$ by setting `rs` argument in `s2fill` ($\sqrt{s}$ = 300 at HERA)

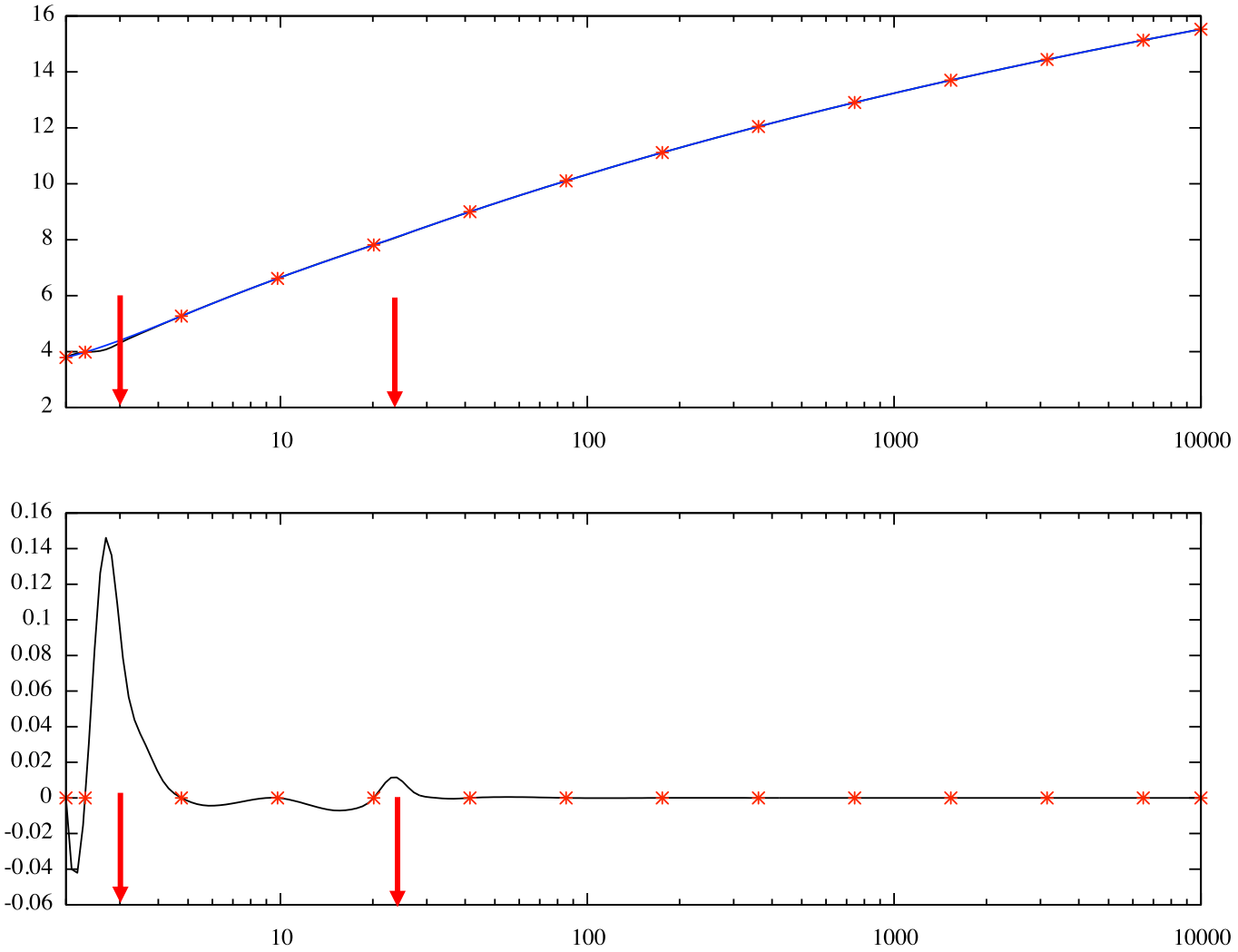
Spline with user nodes

```
ia = isp_s2user(xarray,nx,qarray,nq);
```

1. Input can be un-sorted sparse arrays: routine will turn them into valid nodes
   - Points outside grid are discarded
   - Points are rounded-down to nearest evolution grid-point
   - Sorted in ascending order
   - Discard equal node-points
2. Useful when
   - Auto-nodes cannot be used (create spline in restricted region e.g. between thresholds)
   - Auto-nodes need some fine-tuning

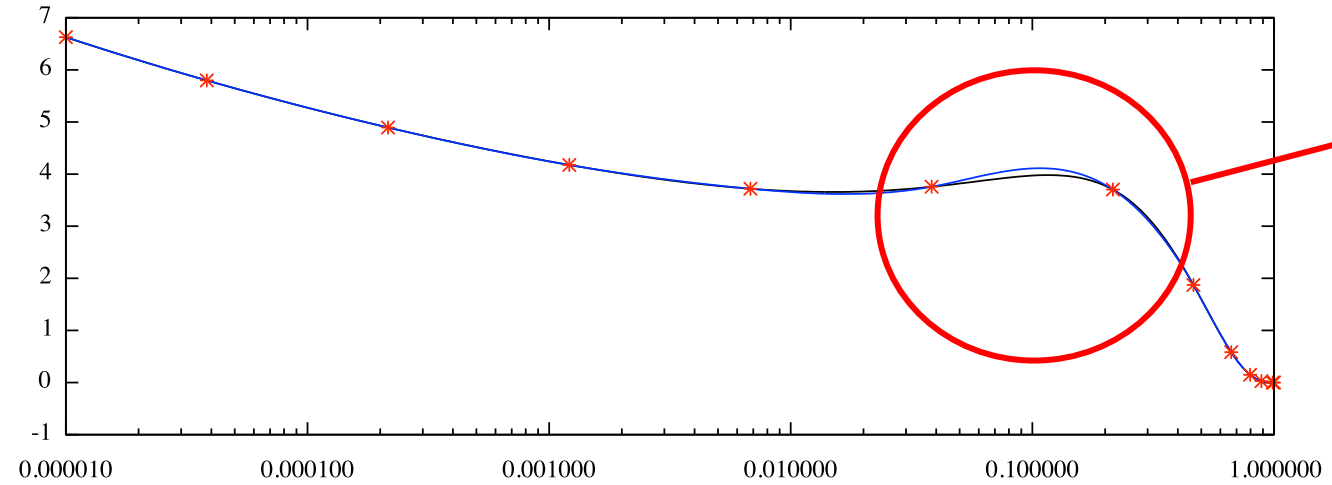
```
ia = isp_s2make(istepx,istepq);          //create spline
ssp_unodes(ia,xarr,n,nx);                //get x-nodes
ssp_vnodes(ia,qarr,m,nq);                //get q-nodes
xarr[n-1] = 0.10;                        //add node point
ja = isp_s2user(xarr,n,qarr,m);          //new spline
```

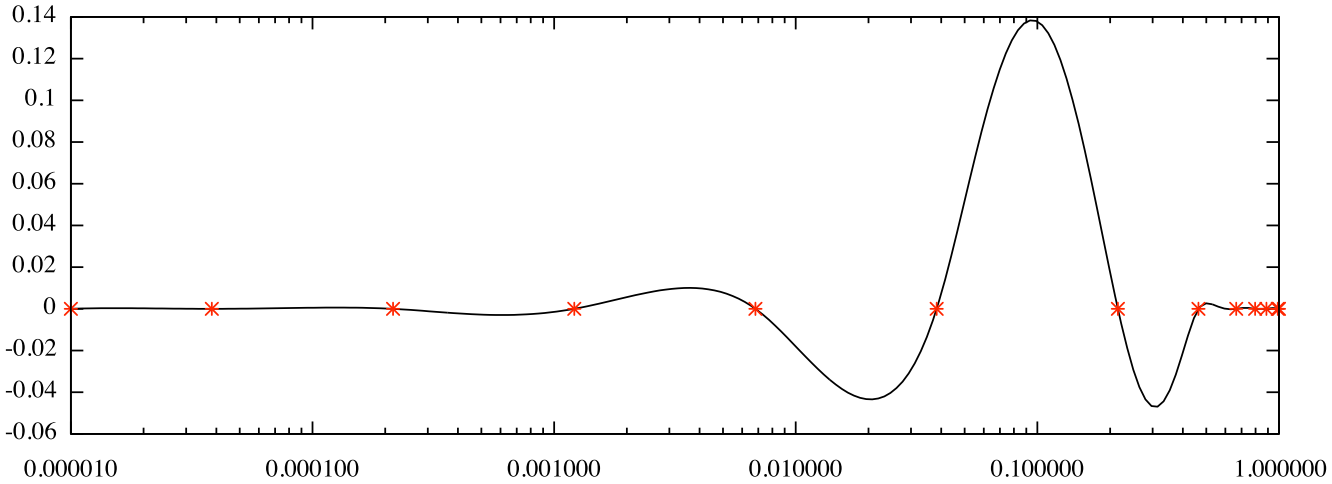
Run QCDNUM in the VFNS



- Spline does not like discontinuities at thresholds
- Should spline each threshold region separately
- Should be non-issue for x-secs since these must be continuous (please check)

Tune proton vs x with grid-step 9

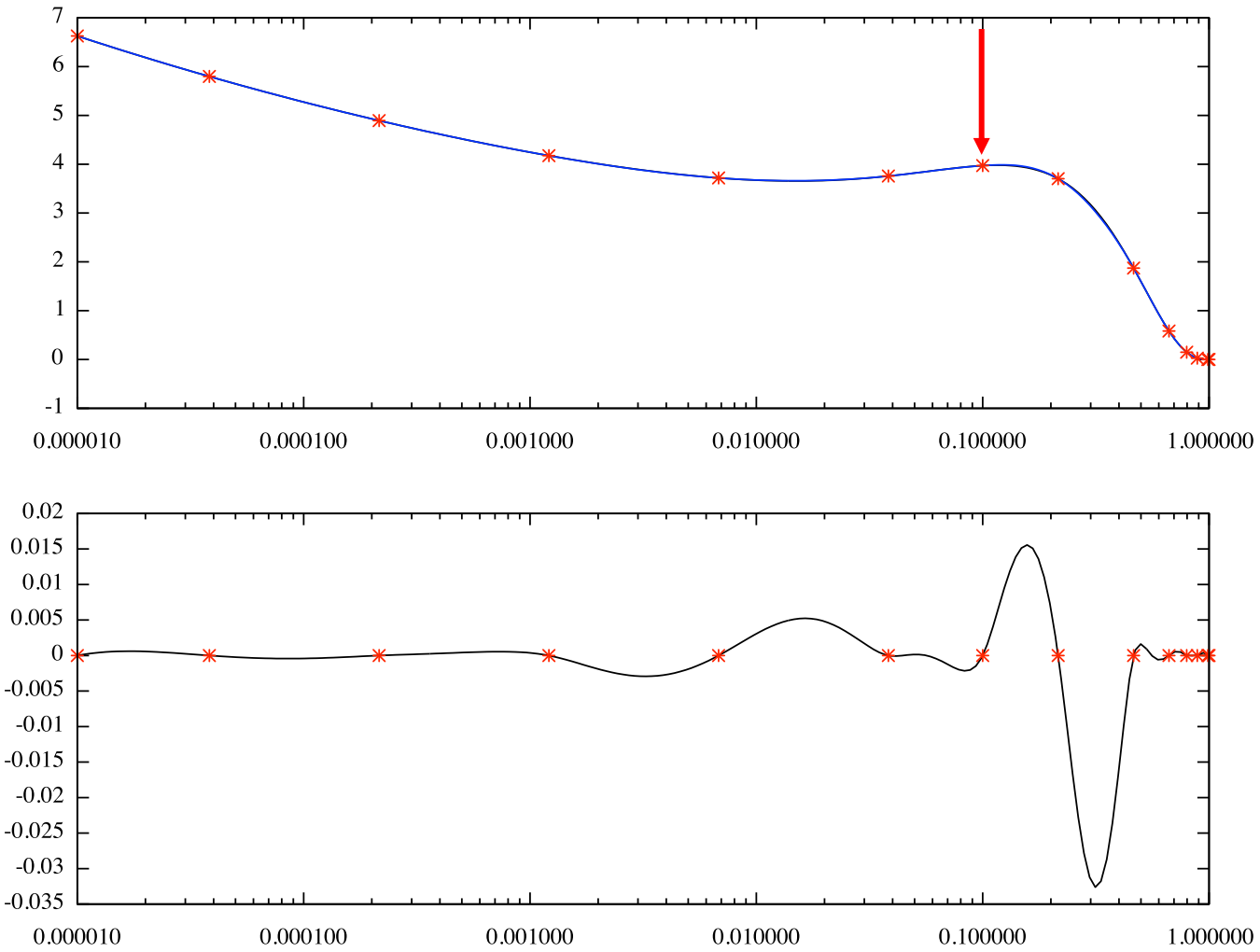




Large deviation of spline with step 9
Add node at $x = 0.1$

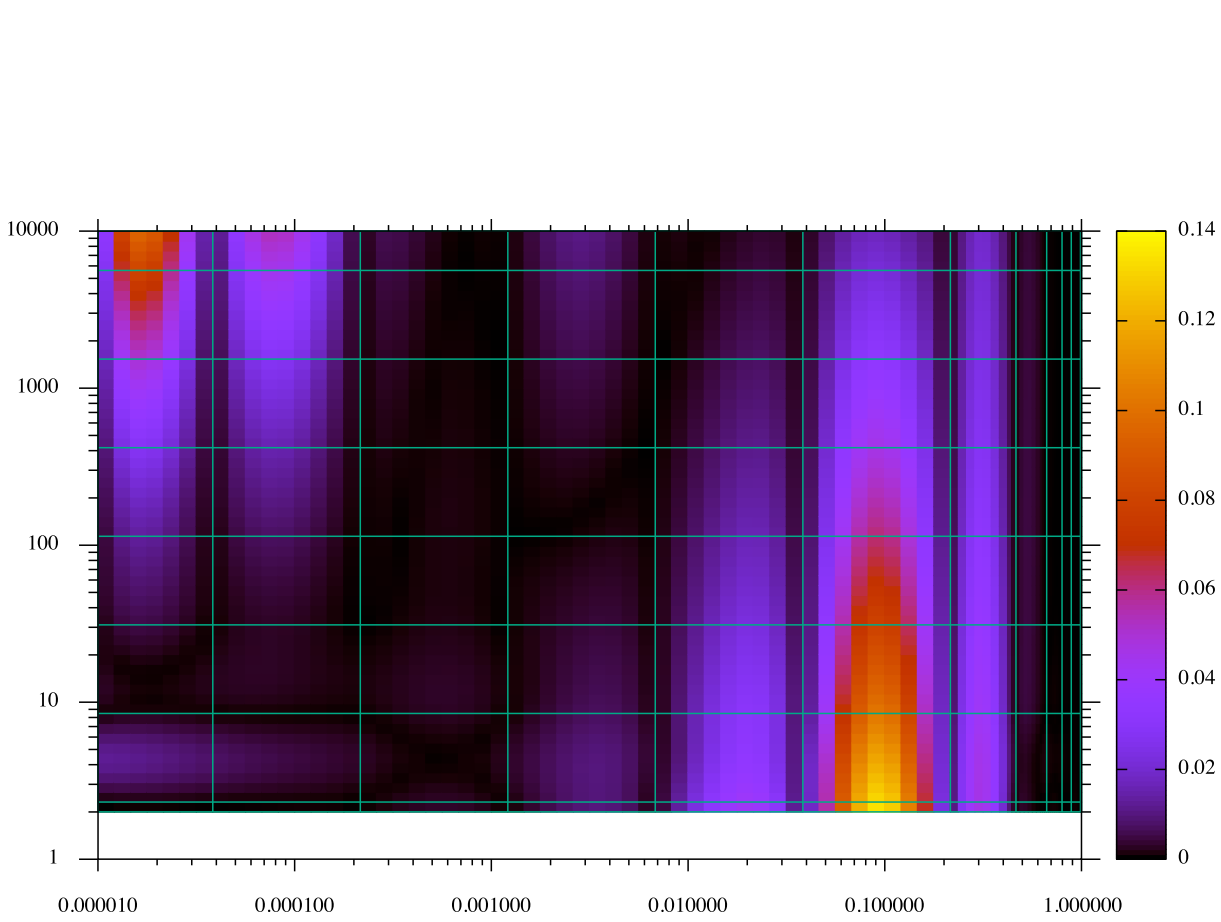
For tuning purposes compare to step-1 reference spline

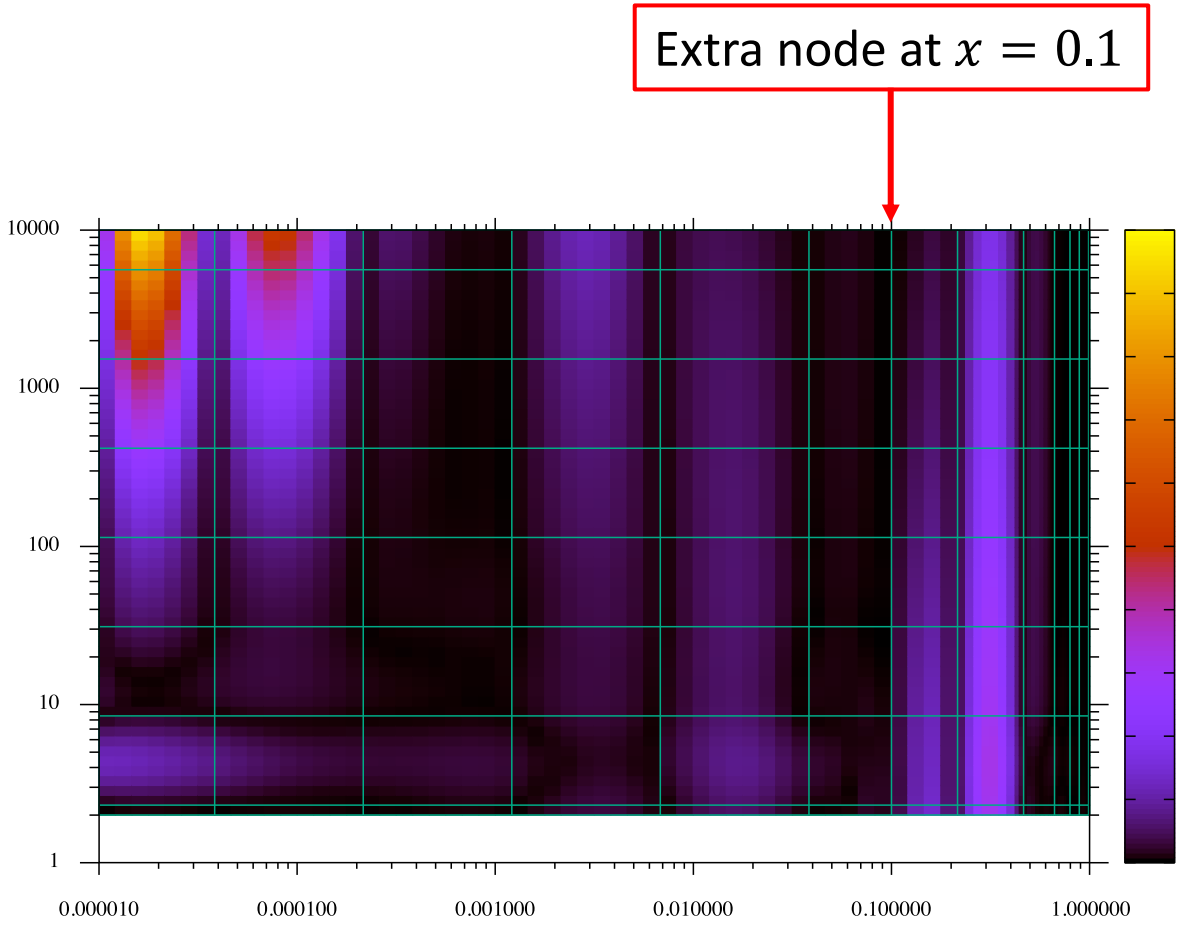
Proton vs x with grid-step 9 + 1 extra node



Just one extra node gives a large improvement in accuracy

2-dim view of tuning the 9-step spline





Can also do 1-dim splines

```
ia = sp_sxmake(istepx);
ia = isp_sxuser(xarray,nx);
ssp_sxfill(ia,fun,iq);

val = dsp_funs1(ia,u,ichk);
```

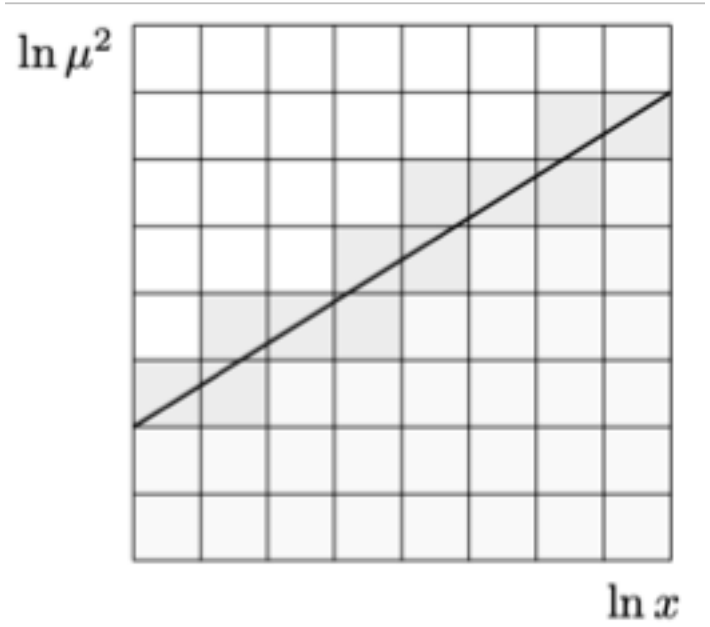
```
ia = isp_sqmake(istepq);
ia = isp_squser(qarray,nq);
ssp_sqfill(ia,fun,ix);

val = dsp_ints1(ia,u1,u2);
```

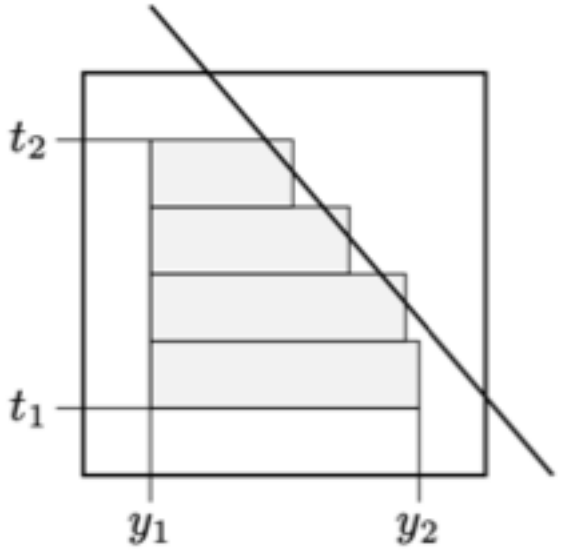
- Input function is the same as for 2-dim with both `ix` and `iq`
- Can fix one coordinate in the `fill` routine
- Thus you can take 1-dim slices in x or μ^2 without re-writing fun
- Can of course also ignore fixed coordinate in the body of fun

Kinematic limit

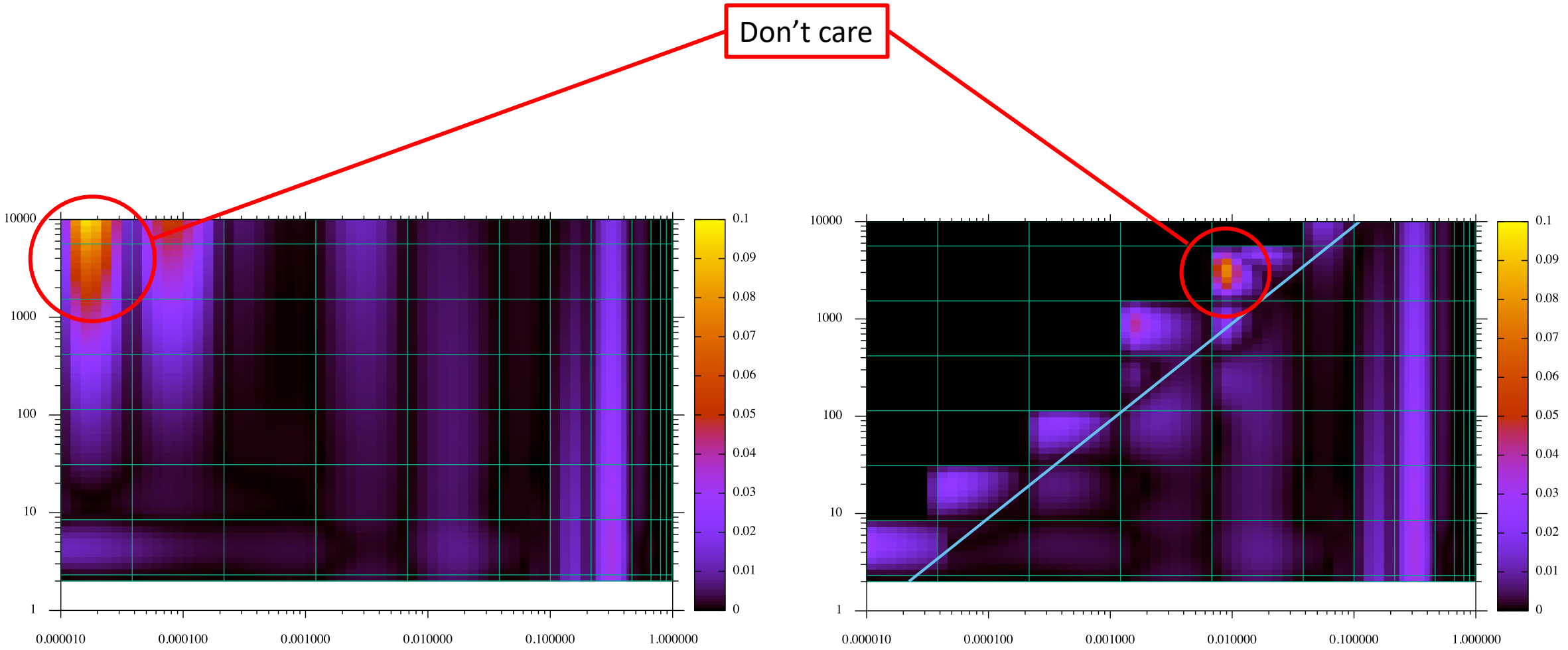
- Enter non-zero value of $\sqrt{s}$ in `ssp_s2fill` routine
- Input x-section undefined above kinematic limit
- Spline needs function defined over entire bin
- User responsibility to provide reasonable extrapolation above the kinematic limit in the dark-shaded bins
- Alternative: spline extrapolation not yet implemented (may be too unreliable anyway)



- Integration over bins crossed by the cut is not yet implemented but it is more or less clear what to do
- Royal pain to find fast algorithm to sub-divide box & integrate



$x\mu^2$ plane with and without kinematic cut



Integration

- Bit complicated since splines are polynomials in $y = -\ln x$ and $t = \ln \mu^2$ and not in x and μ^2
- Introduces Jacobians e^{-y} and e^{t} in the integrals
- Described in appendix A and B of the write-up
- 1- and 2-dim integrals checked against Gauss numerical integration → OK
- 2-dim Gauss much slower than SPLINT integration
- Integral over bin with crossing kinematic limit not yet implemented

So where are we?

- Fortran routines can already be interfaced to JULIA
- Shakedown of C++ interface in progress; you can try but don’t complain
- Integration of bin with crossing kinematic cut still to be done
- Also few bells and whistles are still missing

We are close to getting there