Validation CV_CF plots for HIGG-2016-22 (Blue dots are the Exp. data from *Figure 8b*):

https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PAPERS/HIGG-2016-22/

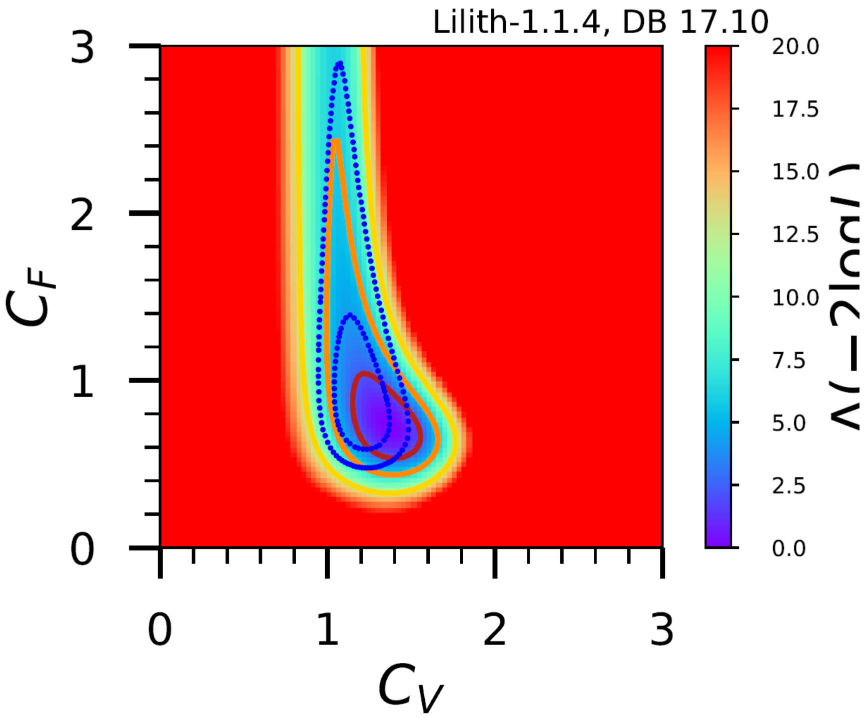

Figure 1

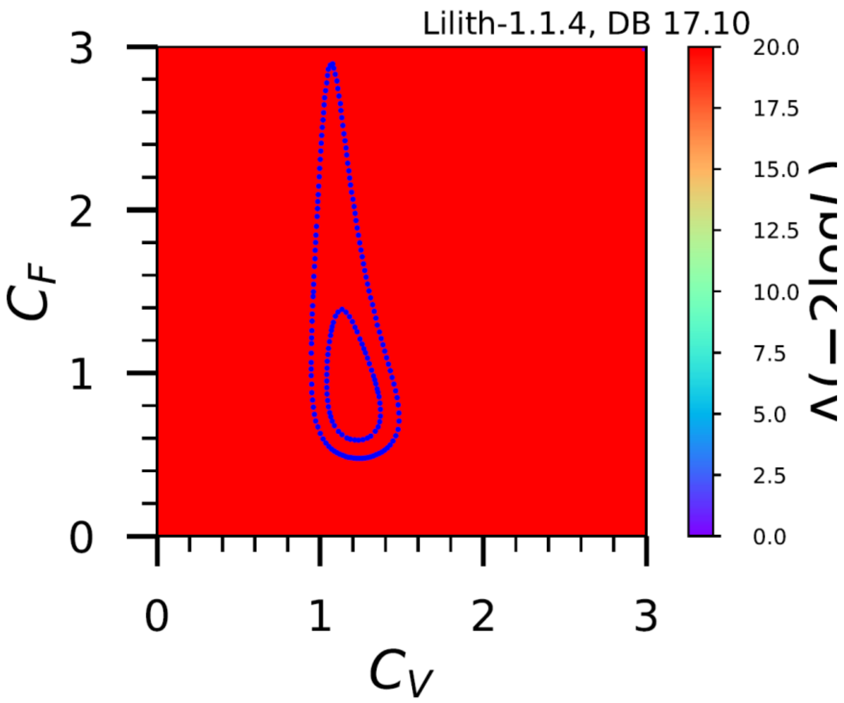

Figure 2

- Pure 2D data (Fig.1):

ATLAS\Run2\36fb\HIGG-2016-22_VBF-ggH_ZZ_n68.xml

- Pure full 1D data (Fig. 2):

ATLAS\Run2\36fb\HIGG-2016-22_VBF_ZZ_f.xml

ATLAS\Run2\36fb\HIGG-2016-22_ggH_ZZ_f.xml

- 1D + 2D data (still Fig. 2):

ATLAS\Run2\36fb\HIGG-2016-22_VBF-ggH_ZZ_n68.xml

ATLAS\Run2\36fb\HIGG-2016-22_VBF_ZZ_f.xml

ATLAS\Run2\36fb\HIGG-2016-22_ggH_ZZ_f.xml

Thanks to Ms. Nhung, who suggested the problem might lie at extrapolation of Log Likelihood for the full 1D data as she'd noticed some inflations there.

I have checked "readexpinput.py" and found the problem was at line 553:

```
Lxy = interpolate.UnivariateSpline(grid["x"], grid["L"], k = 3, s = 0)
```

Here Lilith use "scipy.interpolate.UnivariateSpline" as mentioned in:

https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.UnivariateSpline.html

I extracted with the grid (blue dot) from HIGG-2016-22_VBF_ZZ_f.xml (HIGG-2016-22_ggH_ZZ_f.xml yields the same results - checked) and saw the inflation for the extrapolation as Fig. 3 (interpolation is still fine).

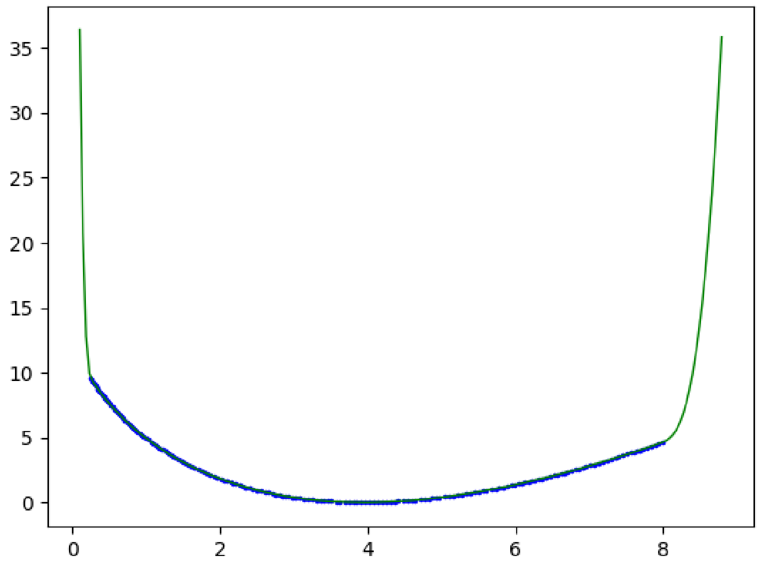

Figure 3

I restricted all the plots below in range -1 to 11 to make comparisons:

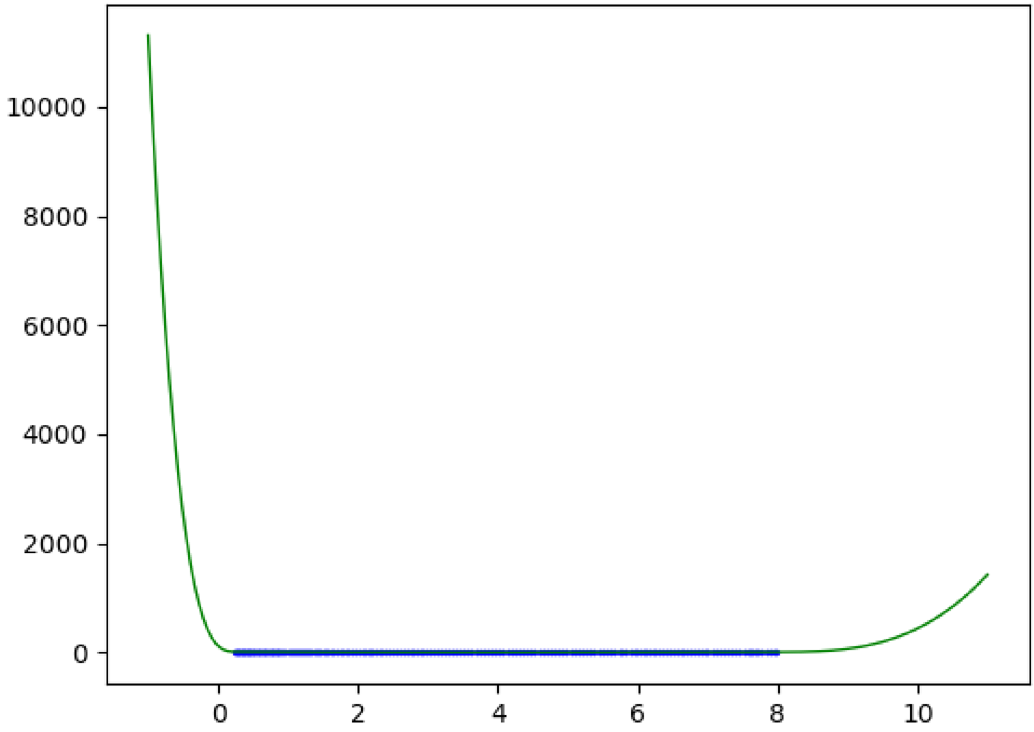

Figure 4: The original `k = 3, s = 0 (for range -1 to 11)`

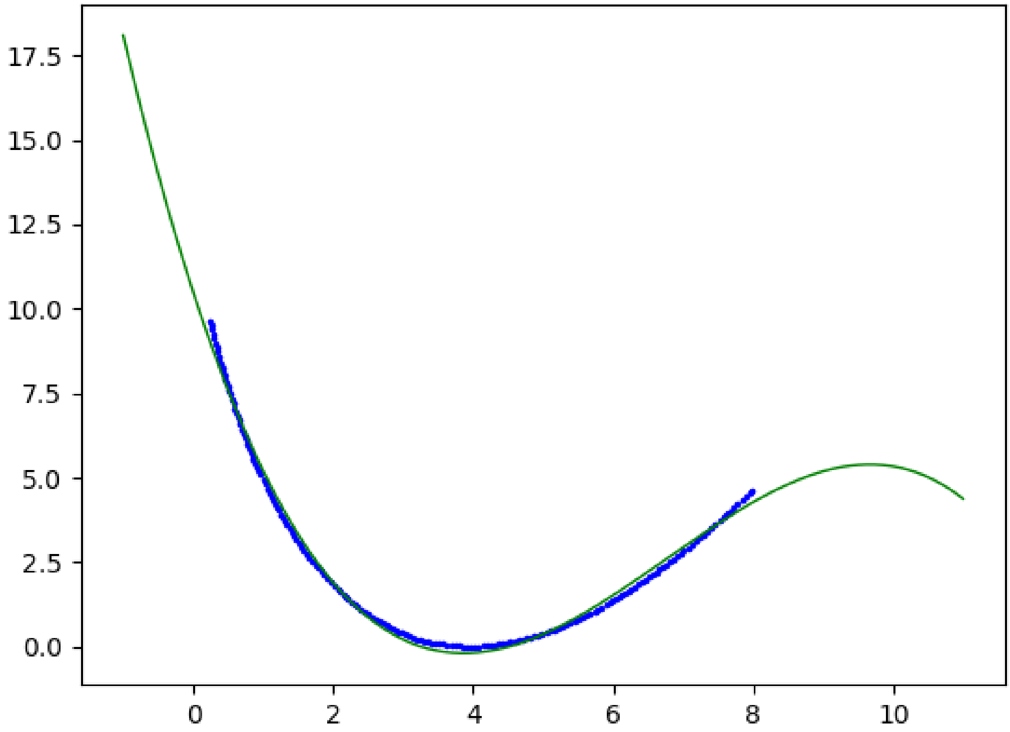

Figure 5: `k = 3, s = None`

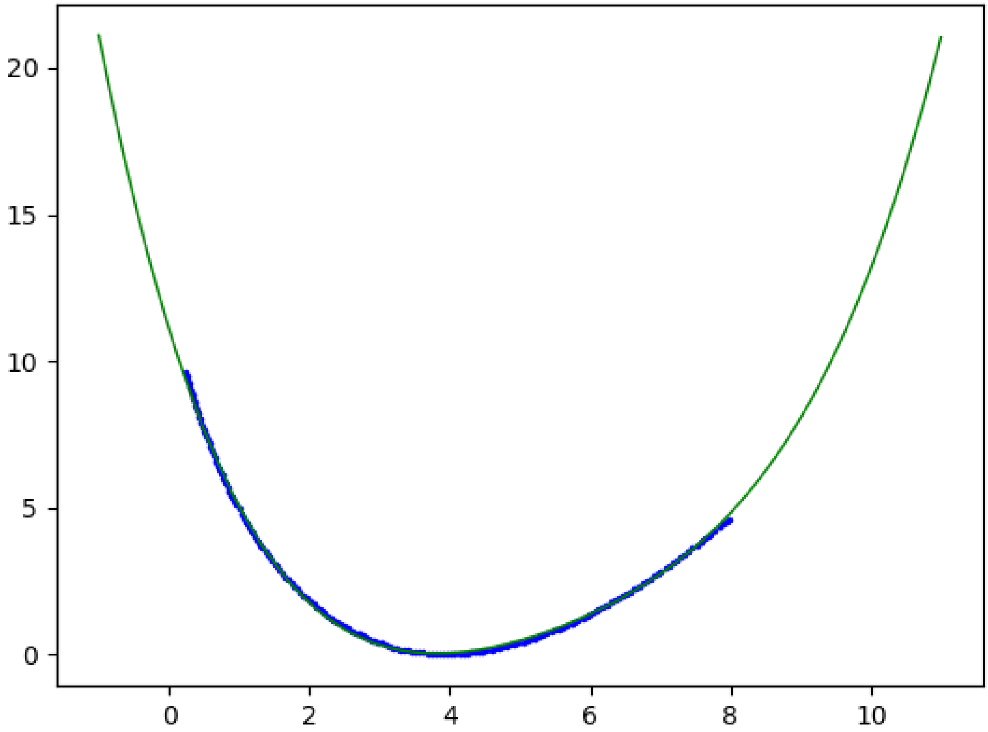

Figure 6: `k = 4, s = None`

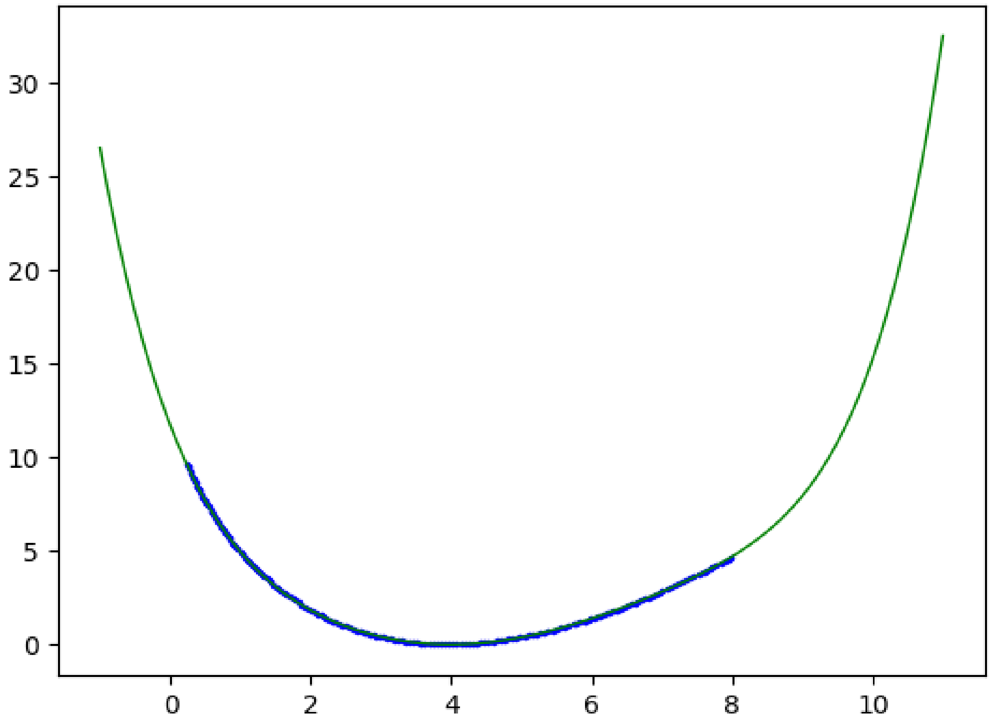

Figure 7: `np.poly1d(np.polyfit(x,y,6))` (same as `k = 6, s = None`)

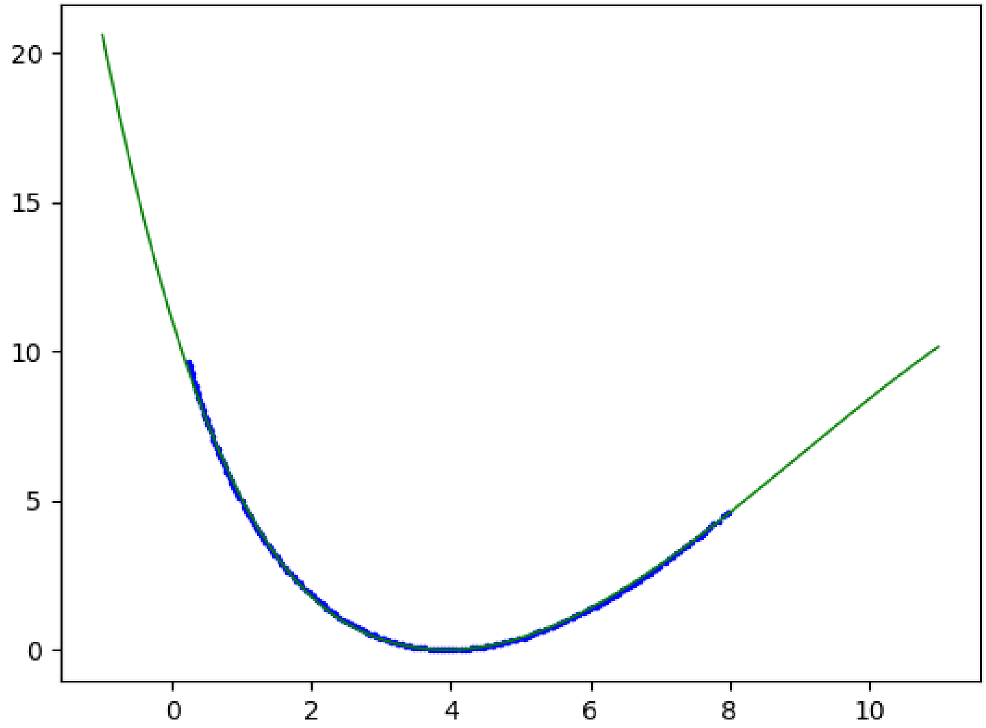

Figure 8.1: `k = 3, s = 1`

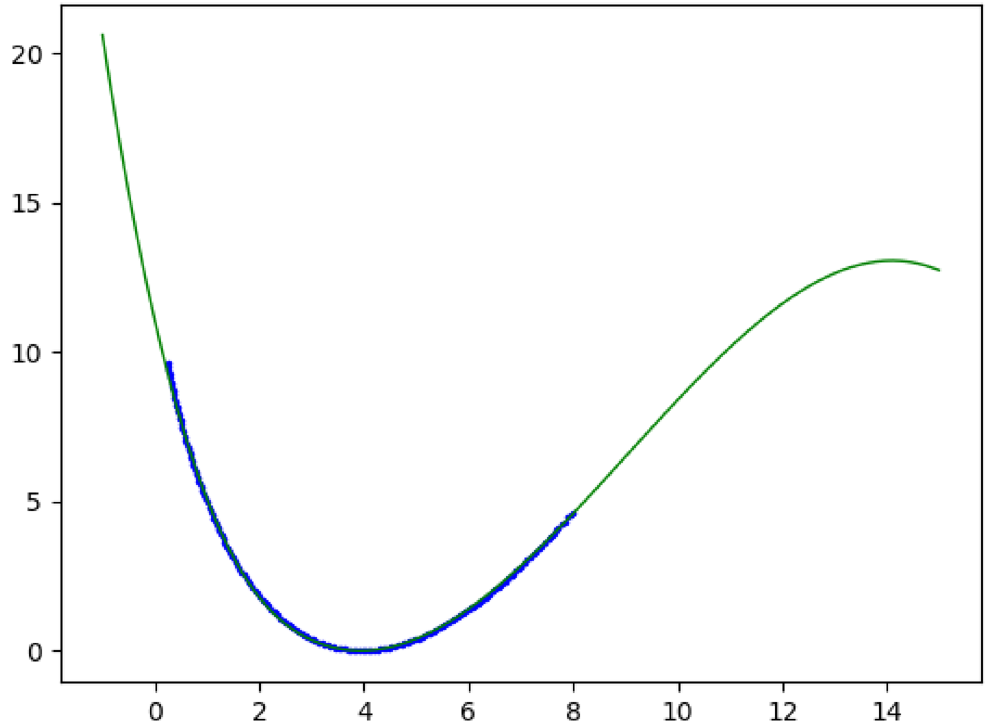

Figure 8.2: `k = 3, s = 1`

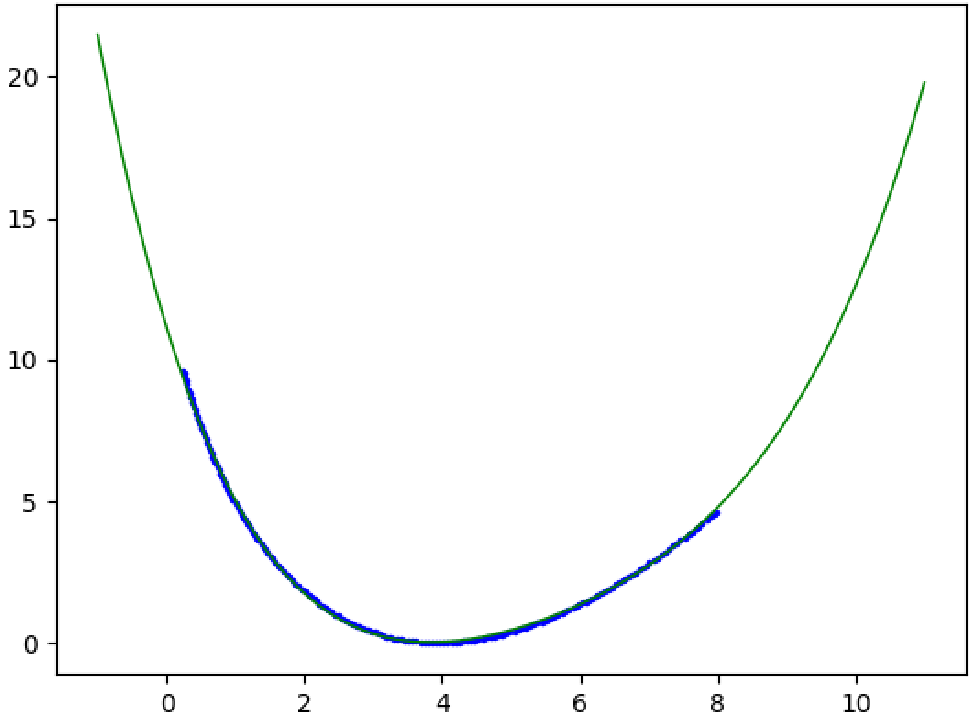

Figure 9: `k = 4, s = 1`

Comment:

1. “s = None” means exactly polynomial to the order k.

   ```
   Lxy = interpolate.UnivariateSpline(grid["x"], grid["L"], k = 6, s =
   None)
   ```

   The same as using numpy to the same order (here is 6):

   ```
   Lxy = np.poly1d(np.polyfit(grid["x"], grid["L"], 6))
   ```

   (Links: https://plot.ly/python/interpolation-and-extrapolation-in-1d/)

   The scipy.interpolate limits k <= 5, for s = None, we can use the above numpy for higher orders of k, as Fig.7.

2. I tried with several values of s and k (you can also try them with my attached python file), the best fits are k = 4, s = 1 (Fig.9) and k = 3, s = 1 (Fig 8.1). The k = 3 looks a bit better. However, for odd k, the graph will turn down at some extrema near there (like Fig.8.2 when I extended the range of Fig 8.1), while for even k, there’s no other extremum(!). Actually, putting k = 3, s = 1 yields back the wrong validation of Fig.2. To the best approximation we have now, I will choose k = 4, s = 1. It actually solved the full 1D validation problem (Fig.10, Fig.12)

3. From Fig. 10, 11, 12, we see that the 2D data is good for explaining the upper tails of the plots.

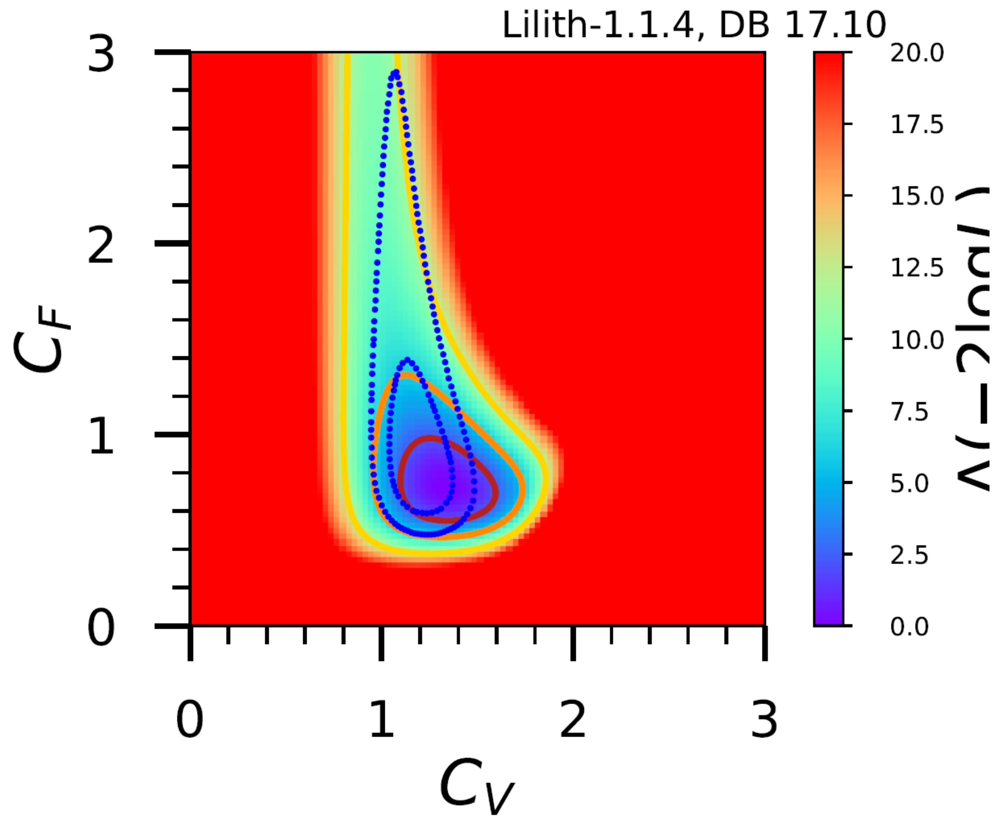

Figure 10: Fixed pure full 1D data

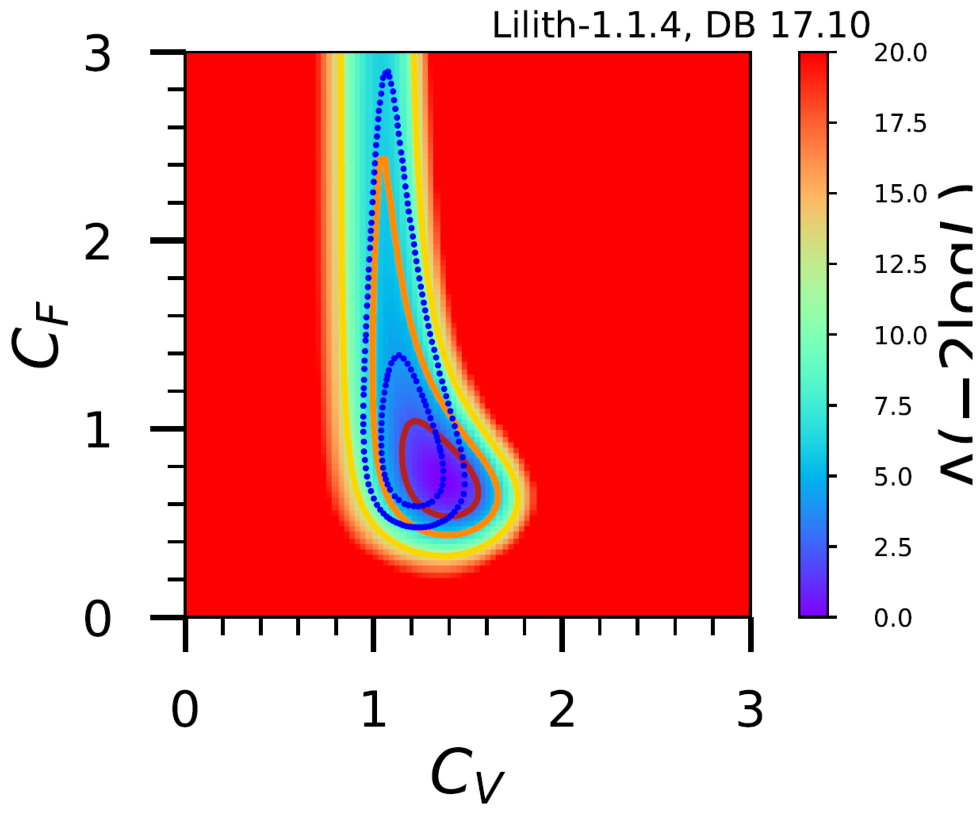

Figure 11: Pure 2D data, same as Fig. 1

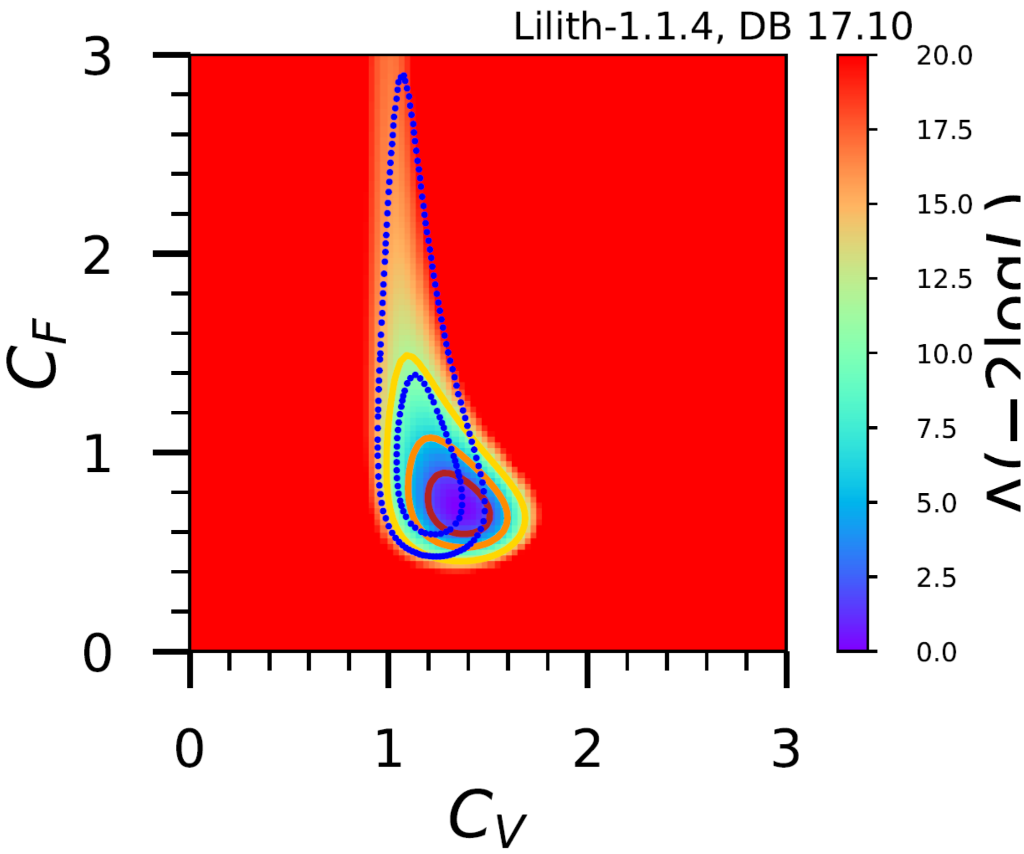

Figure 12: Fixed 1D + 2D data