**Scoring Algorithm**

1. Read model quantiles and corresponding values from the dataset for a forecast date , target end date and model along withthe corresponding ground truth.
2. Generate 1D grid **V** for **v** such that **V** = [floor(vmin)+0.5 : ceil(vmax)-0.5] and each grid point is separated by dV=1 (I want a small dV and I want to keep it the same across all model forecasts).
3. Calculate **Q** corresponding to **V** based on interpolation (e.g. linear or PCHIP) of **q**.
4. Find the numerical derivative (e.g. Python np.gradient(Q,V)). This is the pdf.
5. Find the grid points Vi and Vi+1 thatG the true value falls.
   1. If both grid points are available, then define p(G) = dV \* .
   2. Else assign p(G) = 0 (although this probability is non-zero, it is significantly small).
6. The score S is then , such that larger score means higher fitness.

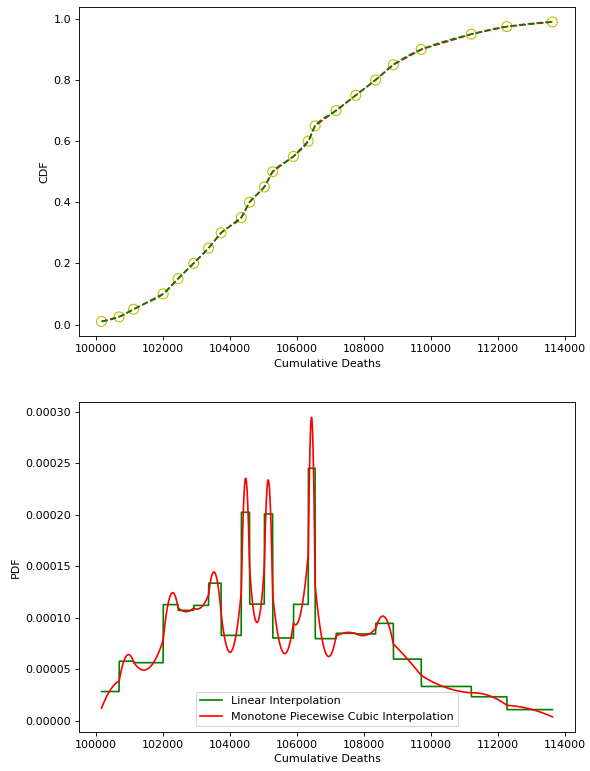


Figure 1: Example interpolation schemes

With this algorithm we penalize models with large prediction intervals. When we integrate the obtained pdf p and compare with the provided cdf, we see the difference between the two is small.



**Caveats:**

1. We assign -Inf for some when the reported prediction quantiles do not encompass the ground truth G. When reporting average scores, we use *nanmean* and do not penalize models for completely missing the ground truth.
2. When reporting average forward score for a model, we give equal weights to forecasts made earlier in time. For example, if a model m made a forecast on di,di+1, and di+2 for a particular target date t, our average forward score is the average of .