Optimising Likelihood Calculation

And some testing ….

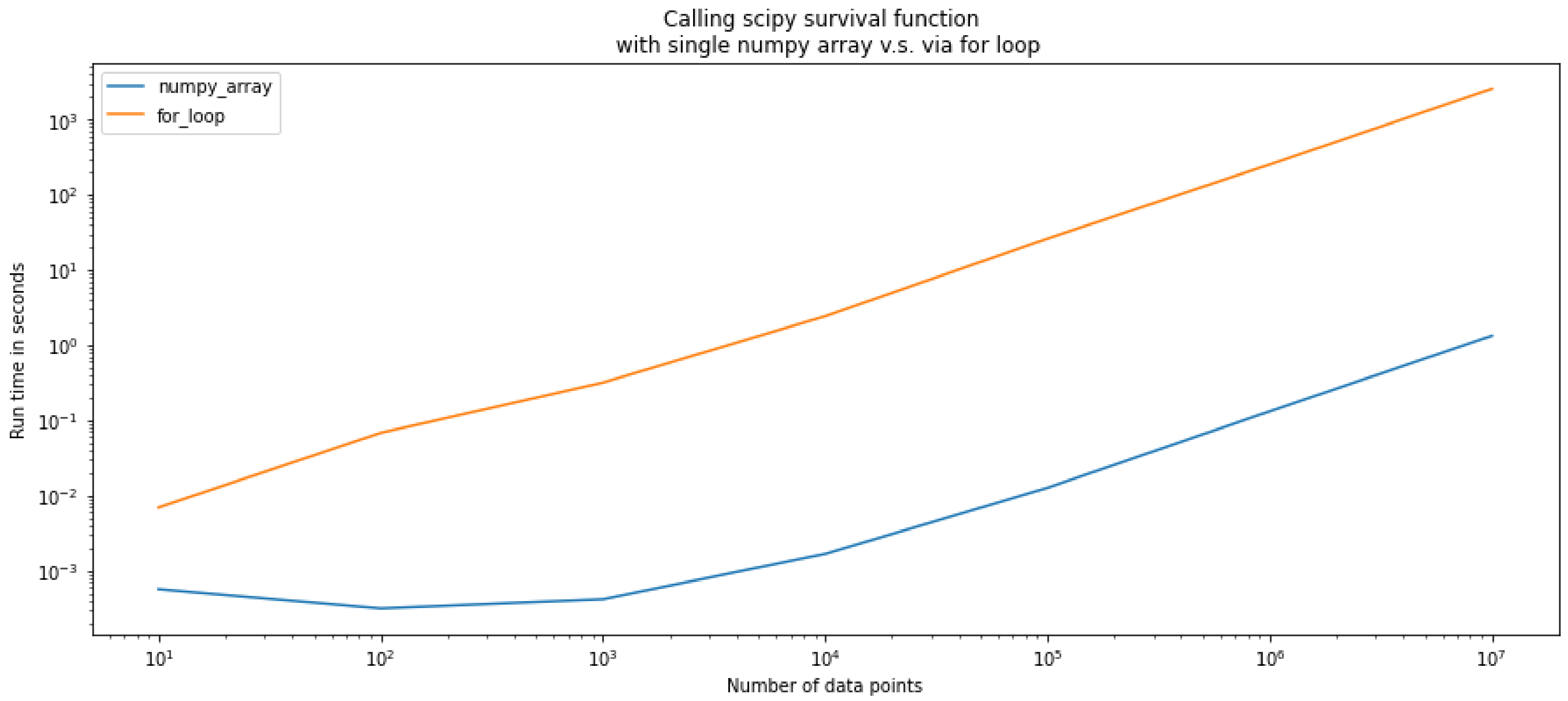
Scipy Survival Method – Profile (1)

- Most of the time spent computing confidence levels currently comes from calls to the scipy survival method via loops.

- The plot on the next slide shows that we can improve code efficiency by calling scipy survival on a numpy array as opposed to looping through values.

- The code used to produce the plot on the next slide can be found here https://github.com/SeanBrayUCL/contur_profiles/tree/main/Profiling_Code .

Scipy Survival Method – Profile (2)



Current Contur

- We currently call scipy survival via two avenues:
  1. Within the __pval method of likelihood object, one call is to compute p values from test statistics, and another call is to compute a confidence level from test statistics. __pval is called once per analysis object in a yoda file, so in a full grid of n yoda files we will get $\sum_{i=i}^{n} 2m_i$ calls where m_i is the number of analysis objects in yoda file i.
  2. Within the __chisq method of the likelihood object, we call scipy survival when self._covBuilt returns False. So through this avenue we get between 0 and $\sum_{i=i}^{n} m_i$ calls.

- If we simplify a bit and assume each yoda file has m analysis objects and self._covBuilt is always False, with current contur if we have n yoda files we will make **n*m*3** calls to scipy survival.

Proposed Changes (1)

1. **Introduce _pval_to_cls method in likelihood object** which computes a cls direct from a p value without a call to scipy survival. Reduces calls to scipy survival to **n*m*2**. See commit here https://gitlab.com/hepcedar/contur/-/commit/a1131583803e9cefdab51b21a781dfd0b17fed2c .
2. **Introduce likelihood_blocks_ts_to_cls method in yoda_factories**. With this change we no longer compute cls upon instantiation of likelihood object, instead we gather test statistics of all likelihood blocks into a single numpy array and then pass this to scipy survival. This reduces calls to **n*m +n** (n*m from chi, n for calling likelihood_blocks_ts_to_cls once per yoda). See commit here https://gitlab.com/hepcedar/contur/-/commit/9fac4dd425f96935e030c7967bc202bc4a2c32b8 .

Proposed Changes (2)

3. **Introduce method likelihood_blocks_find_dominant_ts method in yoda factories.** With this change we no longer complete the dominant chi test statistic (only done when self._covBuilt is False) upon instantiation of likelihood block, instead the new method does this all once for the whole yoda file. Reduces calls to **2n**. Commit for the change can be found here https://gitlab.com/hepcedar/contur/-/commit/49740f6f0f99f6eb02e7f18a82cfa1111cc17917 .

Impact of changes, a practical example

- Consider a grid with 100 yoda files, and 2,000 analysis objects per yoda file, currently this will give 600,000 calls to scipy survival. From examining the profile plot of scipy survival we see this will translate into a runtime of between **10^1-10^2 seconds**.

- After the change we would only have 200 calls to scipy survival, translating into a runtime of less than **10^-1 seconds**.

- The value of the change is more evident when we scale up the n, the number of yoda files. If we used n=1000, current run time would rise to between **10^2-10^3 seconds**, while with the changes run time would still be around **10^-1 seconds**.

Downside of the changes

- Currently the likelihood object can instantiate fully (i.e. with cls calculation) independently of the rest of contur.

- So in theory the likelihood object could be instantiated outside of contur and used. These changes will break that ability by tying the full calculation of cls etc… to the yoda factory object.

- Another thing to be aware of is that some internal methods of the likelihood object were needed in the new yoda factory methods, thus making them no longer internal methods (this still needs to be cleaned a bit)

Adding to Contur’s Tests – Added

- We have added two regression tests to Contur’s tests folder:
  1. **Single yoda file run**: Compares that the results .txt file from running Contur on a single yoda file is unchanged from a base .txt file. We can just read the two txt files and check they are equal after excluding the first line of the txt file because the first line of the txt file gives the location where is being run from.
  2. **Grid run:** We compare the DataFrame arrived at using the Depot objects _build_frame method. We use the pandas function assert_frame_equal with it’s default setting (has a tolerance for differences in floats of less than 10^-8) to check that our target and base DataFrames are the same.

Adding to Contur’s Tests – To Add

- Include a theory run with the regression tests.
- Unit tests of individual objects. A natural place to start here would be the likelihood object considering the optimisation work currently ongoing.
- Simple test to check contur plot runs fine.