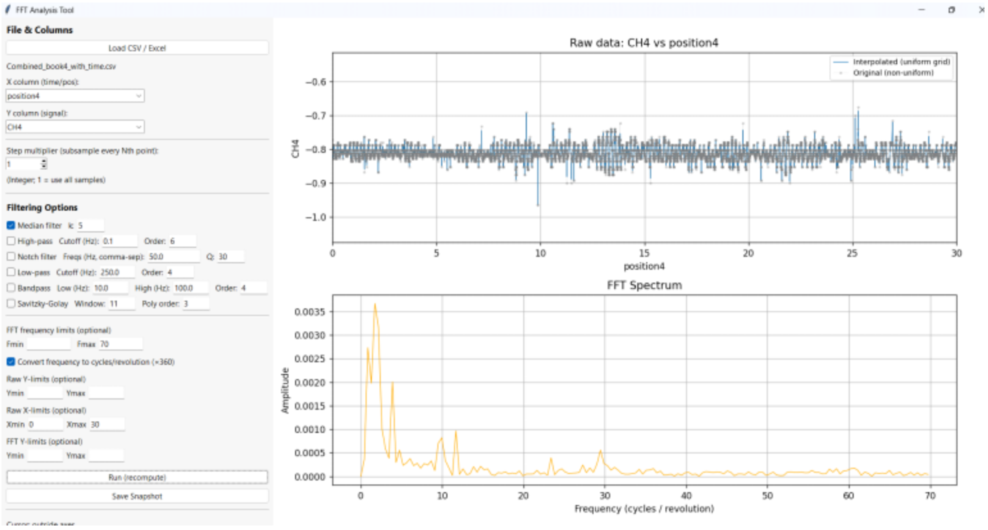
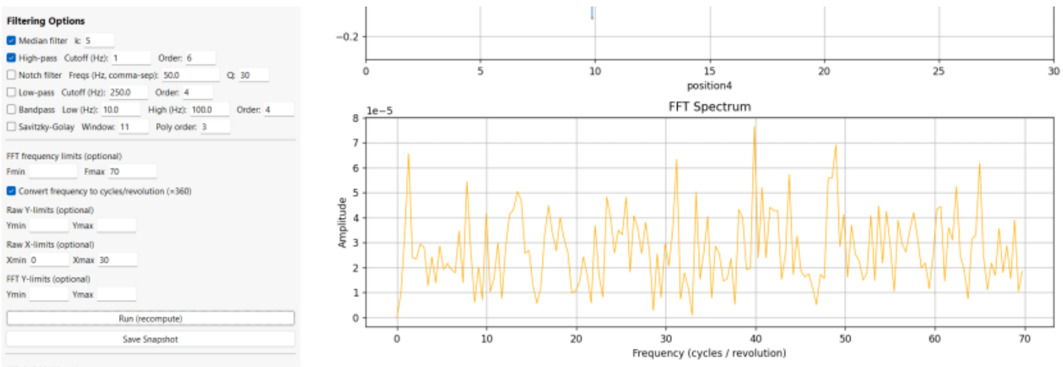
FFT - algorithm checks

10 November 2025 21:46

Problem 1: **<u>Key note</u> --> Need to evaluate the High pass filter:**



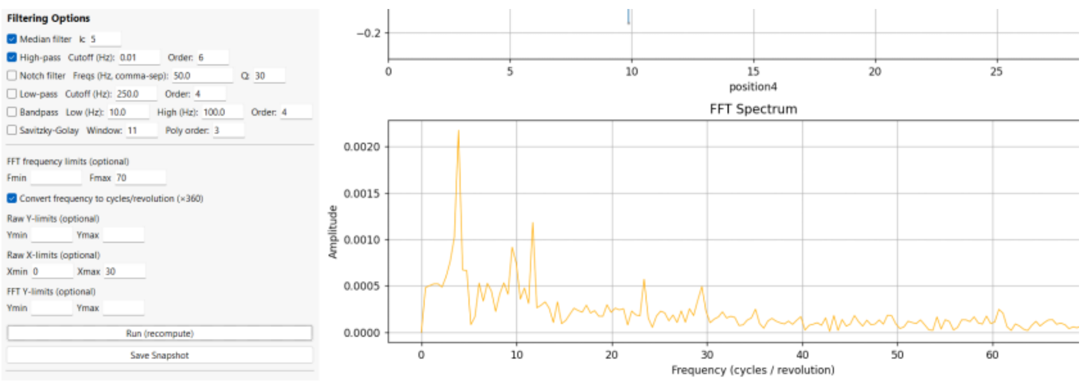
Actually it is working fine:



On initial inspection, a high pass at 1 Hz frequency appears to produce a whole load of nonsense at the output. It seems to remove even some of the more important salient peaks, like the one that appears 4-5Hz, 10 Hz, 12 Hz…

The key thing to remember, the FFT is in the spatial domain: but the frequency filter is in the temporal domain. Therefore, if we actually want to filter out that approx. 1 Hz - consider: 1 (cycle/rev) /360 == 0.00278

Creates quite a sharp peak - which is an artefact of the filter rather than a true point. But notice it does result in a fairly flattened spectrum towards the very low end.



Problem 2: **Checking the Zero Crossing script - things to check:**

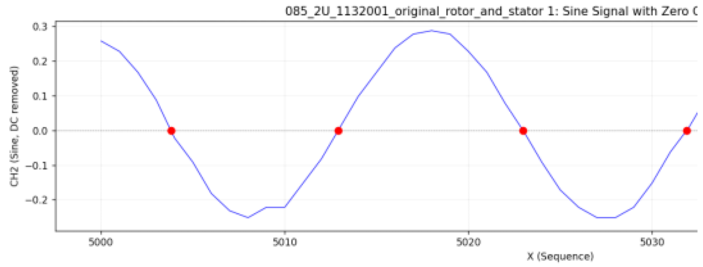
1. Which graph actually gets used? Sine or Cosine sensor

```
# Find interpolated zero crossings for sine (CH2) and cosine (CH3)
zc_x_sine, zc_idx_sine = find_zero_crossings_interpolated(X, CH2_centered)
zc_x_cosine, zc_idx_cosine = find_zero_crossings_interpolated(X, CH3_centered)

print(f"\nZero crossings detected (interpolated to exact y=0 positions):")
print(f"  CH2 (sine) zero crossings: {len(zc_x_sine)}")
print(f"  CH3 (cosine) zero crossings: {len(zc_x_cosine)}")

# Combine zero crossings from both signals and sort by X position
all_zero_crossings_x = np.unique(np.concatenate([zc_x_sine, zc_x_cosine]))
all_zero_crossings_x = np.sort(all_zero_crossings_x)
```

The current algorithm merges them, then sorts into a list: so the zero crossings are collated from both. a 90° phase-shifted encoder you get four zero crossings per electrical cycle (two from sine, two from cosine), so by throwing them all into all_zero_crossings_x we halve the angle step size and skewed the mechanical angle reconstruction.

2. Is there a 0.036 angle in between?

Sine zero crossings:

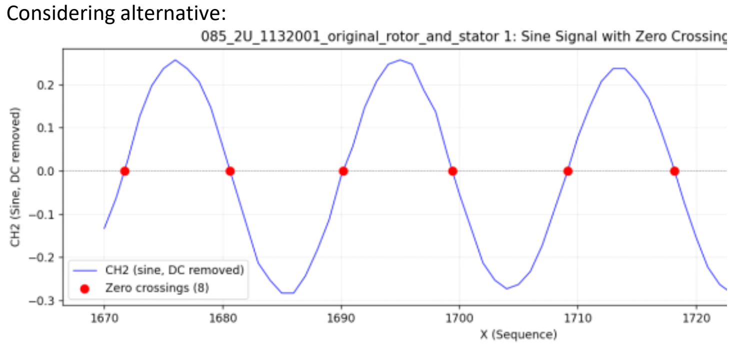


Point	5004	5013	5023	5032
Angle	37.90799	37.98	38.052	38.12399
Increment	-	0.07201	0.072	0.07199

From corresponding section in the .csv file:

```
 Col 1: position4 ,1.18,5007
37.943999999999996,1.18,5008
37.95119999999999,1.06,5009
37.9584,1.06,5010
37.965599999999995,1.18,5011
37.9728,1.26,5012
37.98,1.14,5013
37.987199999999994,1.26,5014
37.9944,1.3,5015
38.001599999999996,1.26,5016
38.0088,1.38,5017
38.016,1.3,5018
38.023199999999996,1.42,5019
38.0304,1.3,5020
38.0376,1.26,5021
38.0448,1.42,5022
38.052,1.3,5023
38.0592,1.26,5024
38.066399999999994,1.38,5025
38.0736,1.3,5026
38.080799999999996,1.26,5027
38.087999999999994,1.26,5028
```

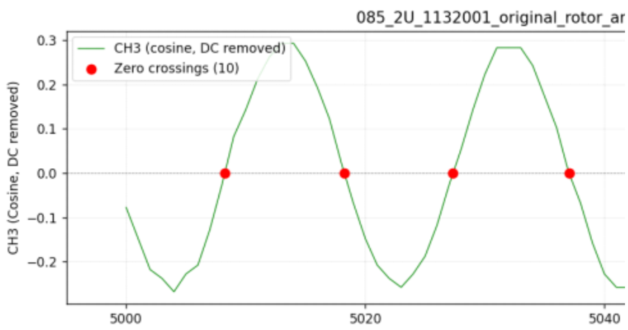
Considering alternative:



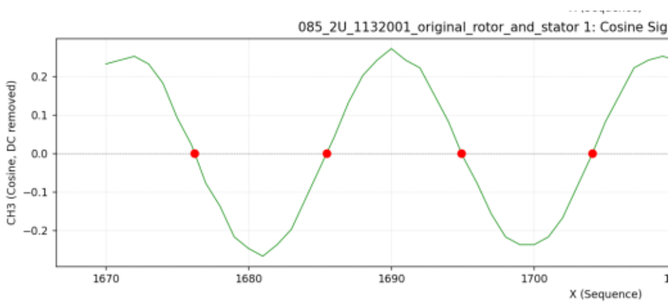
Point	1672	1681	1690	1699	1700
Angle	12.636	12.70799	12.78	12.8448	12.85199
Increment	-	0.07199	0.07201	0.0648	Difference to 1690: 0.7199

Perhaps for plotting and interpolation: given the zero crossing points to do precisely lie at integer X increments, small variations would be expected.

Cosine zero crossings:



Point	5008	5018	5027	5037
Angle	37.94399	38.016	38.080799	38.16
Increment	-	0.072	0.0647	0.0792



Point	1676	1685.5	1695	1704
Angle	12.67199	12.736799 to 12.744	12.81599	12.888
Increment	-	Zero crossing is precisely between 1685 and 1686th point: 0.0648 to 0.07201	0.079191 to 0.07199	0.07201

Even using the cosine zero crossings, similar angles close to the expected 2 x 0.036 deg. is found to occur. However, the key thing to note is that we have often rounded to the nearest integer when hovering over the red dots on the graphs. Assuming a fairly constant speed, which it does seem to be over the rotation: this would indeed give small variations around 0.072 deg.

Compare this to the arctan method used for resolving the angle: 085_2U_1132001_original_combined.csv

Point	5004	5013	5023	5032
Angle	18.9736518	19.007538	19.0448842	19.0795214
Increment	-	0.03381	0.0373	0.03463

Some clear results arise: within about ±1 point, the angular step would indeed agree: at 0.036 deg. per encoder half cycle (of sine/ cosine).

3. **Is linear Interpolation being applied? For angles between zero crossings? Or is it using discrete points to the nearest X point?**
As checked above - this, and this is also evident in the code - interpolation is used to figure out angles on either side. The code does NOT anchor to the nearest X point.